FinGEAR: Financial Mapping-Guided Enhanced Answer Retrieval

Ying Li[1] **Mengyu Wang**[1] **Miguel de Carvalho**[1,2]
Sotirios Sabanis[1,3,4] **Tiejun Ma**[1,5]
[1]The University of Edinburgh, United Kingdom
[2]University of Aveiro, Portugal
[3]National Technical University of Athens, Greece
[4]Archimedes/Athena Research Centre, Greece
[5]The Artificial Intelligence Applications Institute, The University of Edinburgh, United Kingdom
{sunnie.y.li, mengyu.wang, miguel.decarvalho, s.sabanis, tiejun.ma}@ed.ac.uk

Abstract

Financial disclosures such as 10-K filings pose challenging retrieval problems because of their length, regulatory section hierarchy, and domain-specific language, which standard retrieval-augmented generation (RAG) models underuse. We present **FinGEAR** (Financial Mapping-Guided Enhanced Answer Retrieval), a retrieval framework tailored to financial documents. FinGEAR combines a finance lexicon for Item-level guidance (FLAM), dual hierarchical indices for within-Item search (Summary Tree and Question Tree), and a two-stage cross-encoder reranker. This design aligns retrieval with disclosure structure and terminology, enabling fine-grained, query-aware context selection. Evaluated on full 10-Ks with the FinQA dataset, FinGEAR delivers consistent gains in precision, recall, F1, and relevancy, improving F1 by up to 56.7% over flat RAG, 12.5% over graph-based RAGs, and 217.6% over prior tree-based systems, while also increasing downstream answer accuracy with a fixed reader. By jointly modeling section hierarchy and domain lexicon signals, FinGEAR improves retrieval fidelity and provides a practical foundation for high-stakes financial analysis.

1 Introduction

Financial disclosures such as 10-K filings are key for investment analysis, regulatory monitoring, and risk assessment. They are long (often 100+ pages) and organized by SEC-mandated Items, for example, Item 1 (Business), Item 1A (Risk Factors), Item 7 (Management's Discussion and Analysis), and Item 8 (Financial Statements). These sections mix narrative text, tables, and footnotes. Many financial NLP tasks, including sentiment analysis, trend detection, entity extraction, risk detection, and question answering, depend on first retrieving the right passages from these filings. Retrieval is difficult because relevant evidence may be spread across multiple Items or years, and domain synonyms (e.g., "sales" vs. "revenue") and cross-references are common. Therefore, retrieval remains a major bottleneck for current work (Reddy et al., 2024; Edge et al., 2024; Asai et al., 2023; Guo et al., 2024).

Recent efforts such as DocFinQA (Reddy et al., 2024) underscore the difficulty of applying question answering to full-length financial filings. Yet these systems typically treat retrieval as a separate external step and rely on fixed-size chunks or off-the-shelf retrievers, without aligning it to the SEC Item hierarchy and the terminology used in financial reports. This reveals a broader limitation: current retrieval methods struggle with hierarchical organization, domain terms, and the need for precise evidence in financial analysis. FinGEAR directly addresses this gap by redefining retrieval as a first-class objective, tailored to the realities of regulatory filings and their analytical use cases.

Recent advances in Large Language Models (LLMs) (Achiam et al., 2023; Dubey et al., 2024; Wu et al., 2023) and Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) have enabled progress in financial document analysis by grounding outputs in retrieved evidence. Our analysis identifies three core limitations in current retrieval pipelines that constrain downstream performance across financial NLP tasks: (1) *Lack of structure awareness*: fixed-size segmentation discards the logical hierarchy of disclosures, leading to misaligned context retrieval; (2) *Lack of financial specificity*: generic retrievers fail to distinguish nuanced but crucial concepts (e.g., "net income" vs. "operating income"); (3) *Dense-only retrieval is hard to control and explain*: pure vector similarity offers limited interpretability in evidence-heavy settings.

To address these issues, we present **FinGEAR**, **Fin**ancial Mapping-**G**uided **E**nhanced **A**nswer **R**etrieval, a retrieval-centric framework designed for long, professionally authored, semi-structured disclosures. FinGEAR treats retrieval as the core

problem, aiming to surface content that is structurally coherent, financially grounded, and useful across tasks.

FinGEAR introduces three key contributions: (1) *Document–Query hierarchical alignment*, which captures the structural layout of financial documents via a Summary Tree and enables query-sensitive retrieval through a structurally mirrored Question Tree; (2) *Financial Lexicon-Aware Mapping (FLAM)*, which steers retrieval using domain-specific term clusters and lexicon-weighted scoring; (3) *Hybrid dense–sparse retrieval*, which integrates sparse keyword anchoring with dense embedding similarity to balance interpretability and relevance.

Evaluated on full 10-K filings, FinGEAR achieves up to 138% higher retrieval F1 than flat RAG, up to 28% over graph-based baselines (e.g., LightRAG), and up to 263% over prior tree-based systems. Ablation studies confirm that these gains derive from the combined design of its structural and domain-aware modules. While FinGEAR does not directly optimize for reasoning tasks, downstream experiments confirm that enhanced retrieval leads to better answer accuracy, reinforcing retrieval quality as the foundation of financial document understanding.

To our knowledge, FinGEAR is the first retrieval-first system tailored to financial disclosures. It offers a principled and modular foundation for structured, explainable, and task-flexible financial NLP.

2 Related Work

2.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) augments language models by fetching relevant context from external corpora, reducing the need for full-model fine-tuning (Guu et al., 2020; Ram et al., 2023). Advanced variants such as Self-RAG (Asai et al., 2023) and Adaptive RAG (Jeong et al., 2024) improve coordination between retrievers and generators but still use fixed-size chunks. This makes it hard to preserve document structure and can introduce drift in long documents, as seen in long-form QA benchmarks like ELI5 (Fan et al., 2019). Recent work addresses context-length limits with longer-context models (e.g., Transformer-XL (Dai et al., 2019)), retrieval-aware chunking (Zhong et al., 2025), and studies of position bias (Liu et al., 2023). These efforts mainly target input length and do not solve structured retrieval in financial domains.

2.2 Hierarchical and Graph-Based Retrieval

Hierarchical methods such as RAPTOR (Sarthi et al., 2024) and HiQA (Chen et al., 2024) represent documents as trees and retrieve recursively from higher-level summaries. Graph-based systems, including GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2024), model relations between entities and sections to support multi-hop reasoning. In particular, GraphRAG builds local–global graphs over LLM-extracted entities and community summaries and retrieves via community-level traversal, while LightRAG performs dual-level query decomposition with lightweight neighborhood expansion over section-aligned segments. Longtriever (Yang et al., 2023b) targets long-context retrieval by combining local and global semantics at the block level. These graph approaches differ from retrieval that uses section hierarchies (e.g., SEC Items), so we include GraphRAG as a representative graph baseline for community-level traversal. Many graph/tree systems also depend on LLM-generated summaries, which may hallucinate content (Maynez et al., 2020; Li et al., 2023).

2.3 Financial NLP and Domain-Specific Retrieval

Financial NLP supports applications such as sentiment analysis (Zhang et al., 2023; Wang and Ma, 2024), event prediction (Li et al., 2024; Wang et al., 2024), and hybrid QA (Chen et al., 2021; Zhu et al., 2021), often using domain-adapted models like FinBERT-QA (Yuan et al., 2021) and FinGPT (Yang et al., 2023a). These models improve semantic understanding but usually assume that relevant context is provided and therefore lack retrieval. DocFinQA (Reddy et al., 2024) evaluates QA over full filings but relies on an oracle retriever, leaving retrieval design unaddressed. As a result, prior work does not fully model retrieval architectures that reflect hierarchical layouts, domain terminology, and section-specific semantics. FinGEAR addresses this gap by treating structure-aware retrieval as a core objective in financial document understanding.

2.4 Guided and Interpretable Retrieval for Financial Documents

In high-stakes settings like finance, retrieval should be both relevant and interpretable because results support regulatory or analytical decisions (Yu et al.,

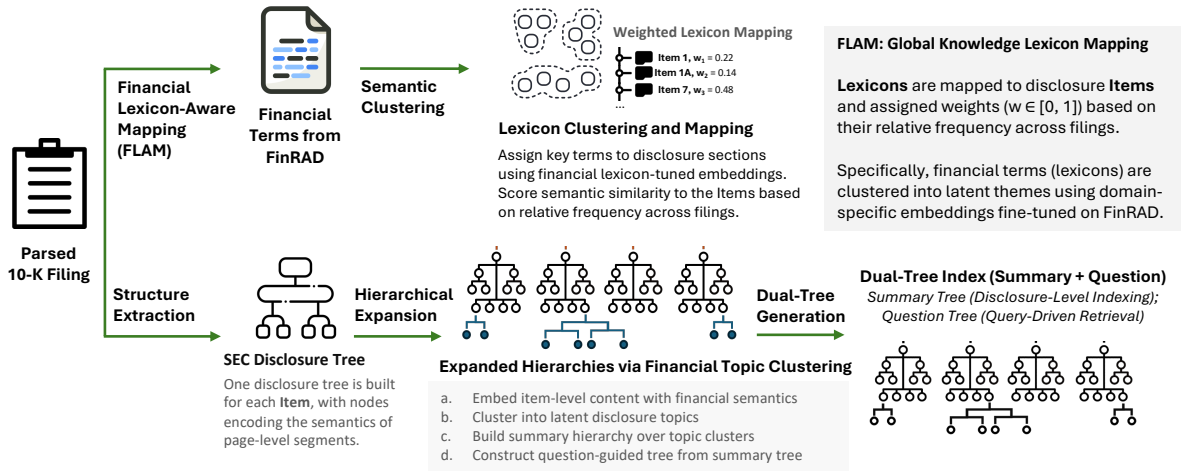

Figure 1: Pre-retrieval. From parsed 10-K filings, FinGEAR performs structure extraction and lexicon mapping (FLAM). FLAM clusters domain terms and assigns Item weights; topic clustering builds a Summary Tree and a mirrored Question Tree for each Item.

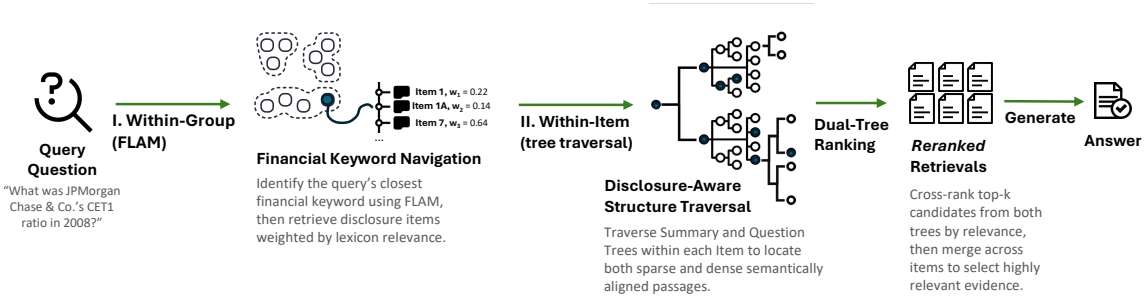

Figure 2: In-retrieval. FLAM allocates the budget across Items (Within-Group). Within each Item, the Summary and Question Trees are traversed (Within-Item). Candidates are jointly reranked and merged across Items. Example query: CET1 ratio in 2008.

2024). Flat, dense-only pipelines can obscure why passages were selected. Structured methods (hierarchical and graph-based) reviewed above improve traceability by encoding document structure and relations, and they typically outperform flat chunking in both quality and transparency. However, most remain domain-agnostic. For 10-Ks, where a standardized section layout and stable terminology are available, integrating domain signals (e.g., a finance lexicon and disclosure Item hierarchy) can further align retrieval with analyst intent. FinGEAR follows this principle by combining lexicon-guided global navigation with Item-aligned hierarchical indexing, providing interpretable, section-aware evidence selection tailored to financial disclosures.

3 Methodology

FinGEAR is a modular retrieval framework designed to align with the structure and terminology of long financial filings. It is motivated by the observation that 10-K reports embed rich domain-specific signals, such as SEC Item headings, disclosure hierarchy, and financial terms. These signals can be extracted and used for improved retrieval. Rather than relying on flat chunking or dense-only similarity, FinGEAR builds hierarchical representations and uses a financial lexicon to guide search with hybrid matching, enabling retrieval with structural fidelity and domain specificity.

3.1 Pre-Retrieval Pipeline

Before retrieval, FinGEAR builds indexing structures for context-aware navigation: (1) *structure extraction* to model disclosure hierarchy, and (2) *Financial Lexicon-Aware Mapping (FLAM)* to steer search toward financially salient content. Figure 1 summarizes this pre-retrieval stage, showing dual-tree construction (Summary/Question Trees) alongside FLAM's lexicon-to-Item weighting that will drive per-Item budgets used later (Section 3.2).

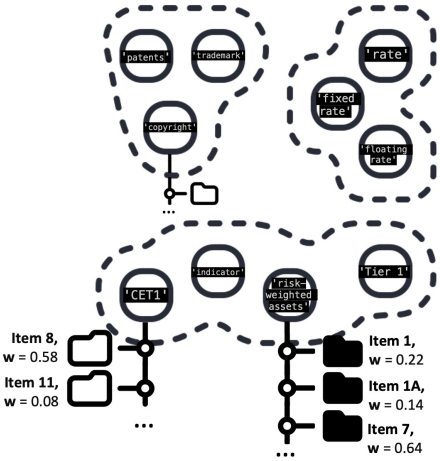

Figure 3: Global navigation with FLAM. Lexicon clusters map query terms to disclosure Items and assign weights $w \in [0, 1]$ based on relative frequency across filings; these weights determine per-Item budgets k_i^* used for traversal.

3.1.1 Structure Extraction

FinGEAR extracts fine-grained structure within each 10-K Item. While SEC-mandated itemization provides a high-level layout (Alberg, 1970), each Item can span dozens of pages and include heterogeneous content. To support retrieval at multiple granularities, FinGEAR builds semantic trees within each Item using topic-based clustering. This hierarchical organization is depicted in Figure 1 as the Summary/Question Tree index built during pre-retrieval.

Chunking and Encoding Each Item is segmented into approximately 2,000-token chunks with a 100-token overlap (see Appendix F, Table 6). We encode each chunk with sentence embeddings fine-tuned on financial data, yielding a domain-aligned representation space.

Hierarchical Expansion via Topic Clustering We then build the hierarchy bottom–up: UMAP is first applied for dimensionality reduction, followed by Gaussian Mixture Models (GMM) for soft clustering. Leaves correspond to the original chunks; each internal node represents a soft cluster and is summarized to guide top-down traversal. The result is a content hierarchy that captures themes from coarse to fine granularity (hyperparameters in Section 5.1; schematic in Figure 1). An example of the resulting hierarchy is shown in Figure 4.

Summary and Question Trees (Outputs) Structure extraction yields two structurally identical hierarchical indices. The *Summary Tree* stores node summaries at internal nodes and the original text chunks at leaves. The *Question Tree* mirrors the same topology but stores LLM-generated sub-questions (embedded in the same query space) at internal nodes; its leaves point to the same chunk IDs as the Summary Tree. We derive the Question Tree by generating sub-questions for each Summary-Tree node and embedding them with a FinQA-aligned encoder. Sharing topology while differing in node content enables hybrid sparse–dense traversal during retrieval. Fine-tuning and further construction details appear in Appendix D.

3.1.2 Financial Lexicon-Aware Mapping (FLAM)

FLAM provides domain-aware guidance by weighting Items before traversal. Candidate terms are extracted using a rule-based method (Singh, 2017) from a curated subset of the FinRAD lexicon (Ghosh et al., 2021), and then clustered with sentence embeddings fine-tuned on financial language (Appendix D). Each relevant term is assigned a weight using *Relative Frequency*:

$$\text{weight}(k_i) = \frac{\text{count}(k_i)}{\sum_j \text{count}(k_j)},$$

chosen for its interpretability and robustness across heterogeneous filings. (Ablations appear in Section 5.2 and Section 5.2.4.)

FLAM vs. structure extraction. FLAM operates at the corpus level: it clusters *lexical terms* (FinRAD-tuned embeddings) and converts them into per-Item *weights* that allocate the global retrieval budget. Structure extraction operates within each 10-K Item: it clusters *text chunks* (FinQA-tuned embeddings) to build *local trees* used for traversal. In short, FLAM decides *where* to look across Items, while structure extraction decides *how* to search within an Item.

3.2 In-Retrieval Pipeline

At query time, FLAM first identifies Items that are likely to contain relevant content. Within these Items, dual-tree traversal retrieves candidate passages: the *Summary Tree* provides sparse, high-level routing; the *Question Tree* provides dense, query-specific refinement. Figure 2 illustrates this process for an example FinQA query: *"What was JPMorgan Chase & Co.'s CET1 ratio in 2008?"*.

```
Key Performance (#docs=0, #qn=5)
├── Revenue Trends (#docs=0, #qn=5)
│   ├── YoY growth (#docs=3, #qn=5)
│   └── Segment revenue (#docs=4, #qn=5)
└── Capital Ratios (#docs=0, #qn=5)
├── CET1 ratio (#docs=2, #qn=5)
└── Tier 1 capital (#docs=3, #qn=5)
```

Figure 4: Example tree printout. Internal nodes are summaries; leaves are retrievable chunks. "#qn" is the number of sub-questions stored at the corresponding Question Tree node.

3.2.1 Global Navigation

Global Navigation selects which SEC Items to search before any tree traversal. Using FLAM, we (i) expand the query into clusters of related financial terms, (ii) measure how strongly each Item is associated with those clusters to obtain normalized weights w_i, and (iii) convert the weights into per-Item retrieval budgets k_i^* such that $\sum_i k_i^* = k$. Items with $k_i^* = 0$ are skipped; the rest proceed to within-Item traversal.

Procedure.

1. *Term detection and expansion.* Extract salient financial terms from the query (e.g., *CET1*, *capital ratio*) and expand them using FLAM's lexicon clusters (FinRAD-tuned embeddings) to capture close variants.

2. *Map terms to Items.* For each expanded term cluster, count its occurrences in each SEC Item across the recovered filings and aggregate these counts per Item. Convert counts to normalized Item weights

$$w_i = \frac{\text{freq}_i}{\sum_j \text{freq}_j}, \qquad \sum_i w_i = 1,\ w_i \in [0,1].$$

3. *Allocate the budget.* Given a total retrieval budget k, assign an Item-level budget

$$k_i^* = \text{round}\big(k \cdot w_i\big)$$

(with a final adjustment so that $\sum_i k_i^* = k$; ties are broken by larger w_i).

4. *Handoff to traversal.* For each Item with $k_i^* > 0$, pass the Item and its budget to the within-Item search over the Summary and Question Trees.

Item weights w_i (e.g., w_1, w_{1A}, w_7) are illustrated in Figure 3 to match the notation used in Section 3.2. For example, with k=10: Item 7 receives k^*=6, Item 1 k^*=2, and Item 1A k^*=2.

3.2.2 Within-Item Search

Within each selected Item, we traverse two trees with identical topology, the *Summary Tree* and the *Question Tree*. The structure (parent–child layout and leaf set) is the same; what differs is the node content used for scoring during traversal: Summary-Tree nodes carry summaries, while Question-Tree nodes carry LLM-generated sub-questions embedded in the query space. Leaf nodes in both trees reference the same chunk IDs, so leaf hits are merged and deduplicated into a single candidate pool.

- *Summary-based retrieval (sparse):* nodes are matched to the query using a bag-of-words scorer over node *summaries*. This favors sections dense in relevant financial terms and headings.

- *Question-based retrieval (dense):* nodes are matched using cosine similarity between the query embedding and LLM-generated *sub-questions* stored at each node (financial QA–tuned embeddings).

Scoring and stopping. The two trees are traversed independently. In the Summary Tree, nodes are scored with BM25 over node summaries (sparse signal). In the Question Tree, nodes are scored by cosine similarity between the query and each node's sub-questions (dense signal). At each internal node we expand only the top b children by that node's score (default b=3 in all experiments). This per-node child limit is separate from the per-Item budget k_i^*. We continue descending until leaves. Because the trees are built with depth at most 2 (Section 5.1), traversal reaches leaves in at most two steps. All leaves visited across both trees form the Item's candidate pool, from which we select up to k_i^* evidence chunks.

Reranking. Stage 1 (cross-tree): merge and rerank candidates from the Summary and Question traversals with a cross-encoder (*BAAI/bge-reranker-large*, XLM-R-Large, 560M).
Stage 2 (cross-Item): rerank the top spans across different Items to prioritize globally informative, coherent answers. This two-stage reranking sharpens precision while preserving coverage for downstream QA.

Finally, we select up to k_i^* highest-scoring unique chunks for that Item and pass them to reranking.

4 Dataset and Evaluation

4.1 Datasets

To evaluate our retrieval capabilities, we use the FinQA dataset (Chen et al., 2021), a benchmark designed for financial question answering. FinQA contains 8,281 question–answer pairs derived from 10-K filings, requiring numerical reasoning and domain expertise. The dataset is split into training (6,251), validation (883), and testing (1,147) sets.

While the FinQA dataset provides ground-truth question–answer pairs, it only includes *pre-extracted* context passages. To enable full-document retrieval evaluation, we recover the original 10-K filings corresponding to each FinQA instance. Using SEC EDGAR records[1], we match company names to ticker symbols and Central Index Keys (CIKs), resulting in a corpus of 720 full-length 10-K filings (we refer to these as our *recovered 10-K filings*).

These documents span a diverse set of industries—including technology, healthcare, financial services, consumer goods, and energy—and cover companies listed in the S&P 500 index between 1999 and 2019. Filings are converted from PDF to structured Markdown format, preserving original SEC itemization. This setup ensures FinGEAR retrieves from complete, uncurated documents, reflecting real-world retrieval challenges.

Dataset breadth. FinQA dataset includes both numerical and categorical questions and covers single-step and multi-hop reasoning. We use this typology in our evaluations (see Section 5.2 and Table 10), underscoring that FinQA dataset is a multifaceted dataset rather than a narrowly scoped benchmark.

4.2 Evaluation Framework

We evaluate FinGEAR's retrieval performance using the RAGAS framework (Es et al., 2023), which provides *component-level RAG evaluation metrics*. Specifically, we report:

Precision. The proportion of retrieved passages that are relevant to the query.

Recall. The proportion of all relevant passages that are successfully retrieved.

F1 Score. The harmonic mean of precision and recall. This serves as our *primary* retrieval metric, as it best isolates retrieval quality without conflating it with downstream generation performance.

Relevancy. A semantic alignment score based on LLM evaluations, measuring how well retrieved passages support the query's intent.

Retrieval is evaluated at multiple depths, including Top-5, Top-10, and Top-15, to capture trade-offs between precision and coverage. All metrics are reported per depth and averaged to provide a comprehensive assessment.

In addition to retrieval metrics, we report *final answer accuracy*, defined as the correctness of a fixed reader model's response given the retrieved context (reader details in Section 5.1). While not a direct measure of retrieval quality, it reflects the downstream utility of the system for real-world financial question answering and analytical tasks.

To improve transparency, we also report token-level statistics in Appendix F, including average tokens per passage, total input length by depth, average tree depth, and maximum context constraints.

5 Experiments

5.1 Experimental Setup

Baselines. We benchmark against *General RAG* (Lewis et al., 2021), *Self-RAG* (Asai et al., 2023), *LightRAG* (Guo et al., 2024), *GraphRAG* (Edge et al., 2024), and *RAPTOR* (Sarthi et al., 2024). LightRAG is run in its "mix" mode (keyword+dense with local neighborhood expansion). GraphRAG follows the official local–global community summary traversal (community construction and inference-time community propagation). Unless noted, baselines use default settings with `text-embedding-ada-002` for dense similarity. (Full baseline configurations are summarized in Appendix C.)

Reader model. All results use `GPT-4o-mini` as a fixed reader with default settings; only the retriever varies across systems. It was chosen for being a strong, cost- and latency-efficient reader at the time of this study. Crucially, the reader choice is independent of the retrieval pipeline and algorithmic design as our conclusions target retrieval quality.

Model and index settings (promoted from appendices). **BM25:** Lucene; k_1=1.5, b=0.75; tokenizer: PyStemmer (eng).

Dense embeddings: `BAAI/bge-base-en-v1.5` (d=768); fine-tuned FinRAD & FinQA variants with MultipleNegatives + Matryoshka; learning rate 2×10^{-5}; batch size 32. Empirical gains from the embedding fine-tuning are summarized in Table 5.

[1] https://www.sec.gov/search-filings

Table 1: Retrieval performance comparison across baseline models. Results are reported at retrieval depths $k = 5, 10, 15$, where k denotes the number of retrieved passages. The best score for each metric is shown in **bold**, and the second-best is underlined.

Model	Precision			Recall			F1 Score			Relevancy		
	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$
General RAG	0.37	0.37	0.30	0.24	0.26	0.28	0.29	0.30	0.29	0.40	0.43	0.47
Self-RAG	0.74	0.60	0.55	0.27	0.28	0.31	0.39	0.38	0.40	0.30	0.31	0.33
LightRAG	**0.88**	0.85	0.85	0.39	0.42	0.47	0.54	0.56	0.60	0.38	0.37	0.39
GraphRAG	**0.88**	**0.89**	**0.87**	0.56	0.55	0.55	0.67	0.66	0.66	0.17	0.16	0.17
RAPTOR	0.69	0.65	0.62	0.11	0.14	0.22	0.19	0.23	0.32	0.38	0.41	0.45
FinGEAR	0.79	0.76	0.72	**0.61**	**0.62**	**0.65**	**0.69**	**0.68**	**0.68**	**0.50**	**0.64**	**0.62**

Cross-encoder reranker: `BAAI/bge-reranker-large` (default inference settings).
UMAP+GMM: UMAP dim=10 (cosine); GMM max components=50; threshold=0.1; max tree depth=2 (fan-out rationale from Item sizes).
Chunking: ∼2,000 tokens with 100-token overlap.

Retrieval Pipeline. We use **BM25s** (Lù, 2024) for sparse matching and **FinLang** domain-specific sentence embeddings (FinLang, 2025) for dense similarity. The embeddings are fine-tuned on FinQA (question–passage) and FinRAD (lexicon–sentence) objectives (Appendix D). Candidate financial terms are extracted with spaCy's `PhraseMatcher` to drive FLAM's term clustering and Item weighting. Unless noted otherwise, all experiments use this BM25s+FinLang configuration; hierarchical indexing and reranking are detailed in Section 3.

Retrieval Settings. Retrieval performance is reported at depths $k = 5, 10, 15$, reflecting trade-offs between retrieval quantity and contextual precision. All models are evaluated using identical traversal logic and input constraints to ensure comparability. Additional configuration details, including chunk size, overlap settings, and node budget, are reported in Appendix F, Table 6.

Ablation Settings. To assess the contribution of FinGEAR's core components, we conduct a series of ablation studies. We begin by disabling individual modules, the *Summary Tree*, *Question Tree*, and *FLAM*, to measure their impact on retrieval F1 and Relevancy. We then perform multi-component ablations by jointly removing pairs of modules to analyze interaction effects between structural and lexical guidance (Appendix I). In addition, we evaluate alternative lexicon weighting strategies within the FLAM module, including *Relative Frequency*, *Exponential Scaling*, and *Softmax Weighting*. These modules affect Item prioritization during retrieval. Ablation results are presented in Section 5.2.

5.2 Results

We evaluate FinGEAR's performance on four core retrieval metrics—*precision*, *recall*, *F1 score*, and *relevancy*—alongside ablation studies and lexicon-weighting variants. While FinGEAR is retrieval-first, we also report *answer accuracy* to assess how improved retrieval supports downstream tasks.

5.2.1 Retrieval Performance Across Baselines

Table 1 compares FinGEAR with five baselines: General RAG, Self-RAG, LightRAG, GraphRAG, and RAPTOR. Across all retrieval depths and evaluation metrics, including precision, recall, F1 score, and relevancy, FinGEAR consistently outperforms the baselines, demonstrating its effectiveness for structure-aware retrieval over disclosures.

Recall. FinGEAR achieves the highest recall at every depth, reaching 0.65 at k=15, ahead of LightRAG (0.47), Self-RAG (0.31), and General RAG (0.28). RAPTOR lags behind, highlighting limited adaptability to the standardized section hierarchy and terminology of 10-Ks. Strong recall indicates that FinGEAR reliably recovers the reasoning spans dispersed across long documents.

F1. FinGEAR also leads on F1 at all depths, reflecting a balanced trade-off between precision and coverage. The combination of FLAM-guided Item selection and dual-tree traversal yields candidate sets that are both relevant and diverse—important in high-stakes financial analysis.

Relevancy. FinGEAR delivers the most semantically aligned results at all depths, peaking at 0.64. Retrieved passages are not only topically related but also faithful to the specific intent of the query. By contrast, General RAG and RAPTOR return more diffuse or off-target content.

Table 2: Ablation study on the impact of removing individual components from FinGEAR. We evaluate the effect of disabling the Summary Tree, Question Tree, FLAM modules and Reranker. Results are reported at retrieval depths $k = 5, 10, 15$ and averaged across metrics. **Bold** indicates the best-performing configuration.

Ablation Setting	Precision			Recall			F1 Score			Relevancy		
	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$
Full FinGEAR	**0.79**	**0.76**	**0.72**	**0.61**	**0.62**	**0.65**	**0.69**	**0.68**	**0.68**	**0.50**	**0.64**	**0.62**
No Summary Tree	0.41	0.61	0.58	0.33	0.36	0.40	0.37	0.46	0.47	0.29	0.57	0.63
No Question Tree	0.43	0.62	0.57	0.35	0.37	0.40	0.39	0.47	0.47	0.29	0.56	0.63
No FLAM Module	0.42	0.61	0.58	0.34	0.35	0.42	0.38	0.44	0.49	0.29	0.58	0.63
No Reranker	0.51	0.44	0.36	0.37	0.30	0.31	0.43	0.36	0.34	0.45	0.55	0.58

System	$k=5$	$k=10$	$k=15$
General RAG	29.8%	30.3%	30.5%
Self-RAG	28.7%	29.8%	27.4%
LightRAG	35.7%	58.8%	36.5%
GraphRAG	28.4%	29.1%	29.4%
RAPTOR	34.0%	20.9%	37.3%
FinGEAR (Ours)	**49.1%**	**49.7%**	**50.0%**

Table 3: Final answer accuracy with GPT-4o-mini as reader. Accuracy is reported for each retrieval depth.

LightRAG. LightRAG attains the highest precision in isolation (e.g., 0.88 at k=5), but this comes with reduced recall and F1, reflecting a narrower candidate set after entity/relation hops. FinGEAR maintains competitive precision while substantially improving recall and F1, yielding a more balanced profile for long, heterogeneous filings.[2]

GraphRAG. GraphRAG shows high precision and strong recall (0.56/0.55/0.55), yielding second-best F1 (0.67/0.66/0.66). However, its relevancy is lowest (0.17/0.16/0.17), and its final answer accuracy is also low (Table 3: 28.4–29.4%). In our setting, FinGEAR achieves higher F1 and much higher relevancy at all depths, aligning with its stronger downstream accuracy.

5.2.2 Evaluating QA as a Downstream Task: Answer Accuracy

To assess downstream utility, we measure final answer accuracy with the fixed reader introduced above. FinGEAR achieves the strongest accuracy at k=5 and k=15 (49.1% and 50.0%), and remains competitive at k=10 (49.7%), where LightRAG shows a one-off spike (58.8%) but degrades at the other depths (35.7% at k=5, 36.5% at k=15). General RAG and GraphRAG hover near 30% across depths, and RAPTOR is unstable, dropping from 34.0% at k=5 to 20.9% at k=10, then rebounding to 37.3% at k=15. Overall, FinGEAR's accuracy is consistently high and increases with k, mirroring its recall/F1 trends and indicating that it surfaces numerically faithful, context-grounded evidence for fact-sensitive financial QA.

5.2.3 Analysis of Retrieval Depths

FinGEAR maintains strong retrieval quality across all evaluated depths. At $k = 5$, it achieves a high precision of 0.79 and F1 score of 0.69, indicating strong early-stage retrieval performance. As retrieval depth increases, recall steadily improves, reaching 0.65 at $k = 15$, while precision and F1 remain consistently high. This stability reflects FinGEAR's ability to scale retrieval scope without sacrificing semantic accuracy or contextual fit.

Relevancy scores follow a similar pattern—peaking at 0.64 at $k = 10$ and holding at 0.62 at $k = 15$, showing that even as the system retrieves more passages, it continues to surface content that is contextually faithful to the query. These trends suggest that FinGEAR balances depth and alignment effectively, making it suitable for long-context applications where both coverage and interpretability are critical.

In contrast, baselines exhibit more pronounced trade-offs. LightRAG shows strong precision at shallow depths but plateaus in recall and F1. Self-RAG and RAPTOR struggle to maintain performance as retrieval depth increases, reflecting limited adaptability to structured disclosures. FinGEAR's consistent gains across all depths demonstrate its robustness and domain alignment in retrieval from complex financial documents.

5.2.4 Ablation Study

We now report the outcomes of the ablations defined in Section 5.1. We conduct single-component ablations: (1) removing the *Summary Tree*, which supports hierarchical sparse abstraction; (2) remov-

[2] In LightRAG, each retrieval combines entities, relations, and vector text, which can inflate precision but requires approximately fifteen times the runtime of FinGEAR per document.

Table 4: Ablation study of lexicon weighting strategies in FLAM. We compare three methods for computing weights from financial terminology: Relative Frequency, Logarithmic Weighting, and Softmax Weighting. Scores are reported at retrieval depths $k = 5, 10, 15$ and averaged across all metrics. **Bold** indicates the best-performing configuration.

Weighting Strategy	**Precision**			**Recall**			**F1 Score**			**Relevancy**		
	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$
Relative Frequency	**0.79**	**0.76**	**0.72**	**0.61**	**0.62**	**0.65**	**0.69**	**0.68**	**0.68**	**0.50**	**0.64**	0.62
Logarithmic Weighting	0.70	0.66	0.63	0.47	0.45	0.45	0.56	0.53	0.52	**0.50**	0.60	0.60
Softmax Weighting	0.70	0.68	0.66	0.47	0.44	0.44	0.56	0.53	0.53	0.48	0.59	**0.63**

ing the *Question Tree*, which enables dense query alignment; and (3) disabling the *FLAM module*, which provides lexicon-driven domain targeting. As shown in Table 2, all components are critical to retrieval quality—particularly at lower depths ($k = 5$), where removing the Summary Tree alone reduces F1 from 0.69 to 0.37. These results confirm that FinGEAR's performance arises from the interplay of structural, semantic, and domain-specific signals, rather than any isolated component.

We further assess the impact of lexicon weighting strategies within FLAM. Table 4 compares three approaches: *Relative Frequency*, *Logarithmic Weighting*, and *Softmax Weighting*. Relative Frequency consistently delivers the strongest performance, with an F1 of 0.69 and relevancy scores of 0.50, 0.64, and 0.62 at $k = 5$, $k = 10$, and $k = 15$, respectively. Logarithmic Weighting downweights frequent but meaningful terms, reducing recall to 0.45 at $k = 15$, while Softmax Weighting slightly improves relevancy at $k = 15$ (0.63) but lowers F1 by over-concentrating on dominant keywords. These results affirm Relative Frequency as the most robust and interpretable default strategy within FLAM.

Multi-component ablations in Appendix I (Table 8) reveal compounded degradation when two modules are removed. In particular, eliminating both FLAM and the Question Tree results in the steepest drops in F1 and relevancy, underscoring their complementary roles in domain grounding and query sensitivity. Removing FLAM also increases variance across depths, highlighting its stabilizing role and the complementarity of the three modules. These findings reinforce that FinGEAR's effectiveness and stability stem from the coordinated integration of all three core modules.

Question-type breakdown. We also report performance by *Numerical vs. Categorical* and *Simple vs. Complex (multi-hop)* questions; full results are in Appendix J (Table 10). At k=10, FinGEAR attains F1 = 0.68 (Numerical) vs. 0.81 (Categorical), and 0.70 (Simple) vs. 0.67 (Complex).

Two patterns are observed. (1) **Categorical** questions achieve higher F1, likely because answers are stated directly (e.g., presence/absence) and require less aggregation; by contrast, *relevancy* is higher on **Numerical** queries (0.64 vs. 0.48 at k=10), reflecting tighter alignment around figures and units. (2) **Complex** (multi-hop) questions trail **Simple** ones in F1 yet show comparable or higher *relevancy* at larger k (e.g., 0.65 at k=10, 0.67 at k=15), indicating that dual-tree traversal helps surface distributed evidence; the remaining F1 gap is consistent with broader evidence requirements.

Overall, this breakdown shows that FinQA contains diverse question types, such as discrete (categorical) and quantitative (numerical), single-step and multi-hop, providing a comprehensive testbed for retrieval. It also indicates that FinGEAR's gains are not confined to a particular question class; improvements hold across categories, suggesting the method is broadly applicable.

6 Conclusion

We present FinGEAR, a retrieval-first framework for 10-K filings that integrates a finance lexicon (FLAM) for Item-level mapping and dual trees for within-Item indexing. Using full 10-Ks aligned with FinQA queries, FinGEAR demonstrates improved retrieval quality across depths compared with flat, graph-based, and prior tree-based RAG baselines, and yields higher downstream answer accuracy. Ablation studies show that each module is necessary for overall performance. We focus on 10-Ks for their length, standardized section hierarchy, and regulatory importance. The design is modular: FLAM enables global navigation across Items, while the dual trees index and traverse local content. This supports adaptation of FinGEAR to other semi-structured documents by enhancing domain lexicons to reflect evolving terminology.

7 Limitations

FinGEAR is a structured retrieval framework designed for financial filings, but its performance and applicability are subject to several limitations.

Domain specificity. FinGEAR is developed and evaluated primarily on U.S. 10-K reports, which follow standardized regulatory structures. While the framework is not hard-coded to SEC formats, it assumes the existence of segmentable content with hierarchical or pseudo-hierarchical cues. Its generalizability to unstructured financial documents (e.g., earnings calls) or reports from different jurisdictions remains untested and requires further adaptation.

Lexicon dependence. The system relies on stable financial terminology across filings. While this holds for regulatory disclosures, emerging financial language or sector-specific terms may weaken the quality of keyword mapping and clustering, impacting retrieval alignment over time.

Parsing sensitivity. FinGEAR assumes that documents are accurately parsed and structurally consistent. Severe formatting inconsistencies (e.g., OCR errors or HTML-to-text misalignments) could affect the quality of tree construction and retrieval, especially in noisy or historical filings.

Limited reasoning. FinGEAR focuses on semantic retrieval and does not perform explicit financial reasoning or computation (e.g., ratio calculation, time-series forecasting). Retrieved evidence supports qualitative assessments, but numerical understanding remains out of scope.

Evaluation coverage. The evaluation is conducted on a specific dataset (FinQA) with a limited number of annotated gold spans. As FinQA only labels one relevant span per query, retrieval performance may be underestimated. Broader assessment across multiple QA datasets or real-world analyst workflows is needed for a fuller view of utility.

Source bias. Although FinGEAR does not introduce new biases, it inherits those embedded in financial documents and lexicons. If companies omit, downplay, or frame certain disclosures, the retrieved content will reflect those reporting biases.

Despite these limitations, FinGEAR offers a modular, interpretable foundation for structure-aware retrieval in financial analysis. Future work should explore its generalization to more diverse corpora, integration with lightweight reasoning modules, and robustness to document parsing noise.

Acknowledgments

We acknowledge support from the Centre for Investing Innovation at the University of Edinburgh.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom A Alberg. 1970. Sec disclosure requirements for corporations. *Bus. Law.*, 26:1223.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Xinyue Chen, Pengyu Gao, Jiangjiang Song, and Xiaoyang Tan. 2024. Hiqa: A hierarchical contextual augmentation rag for massive documents qa. *arXiv preprint arXiv:2402.01767*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567. Association for Computational Linguistics.

FinLang. 2025. Finance embeddings investopedia. https://huggingface.co/FinLang/finance-embeddings-investopedia. Accessed: 2025-02-14.

Sohom Ghosh, Shovon Sengupta, Sudip Naskar, and Sunny Kumar Singh. 2021. FinRead: A transfer learning based tool to assess readability of definitions of financial terms. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 658–659, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *Preprint*, arXiv:2410.05779.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Peter Henderson, Justin Gilmer, Rishabh Agarwal, Guillaume Bouchard, David Belanger, Gaurav Tomar, and Yann Dauphin. 2023. Matryoshka representation learning. *arXiv preprint arXiv:2305.17137*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *Preprint*, arXiv:2403.14403.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Koduvayur Subbalakshmi, Guojun Xiong, et al. 2024. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. *arXiv preprint arXiv:2412.18174*.

Huaxia Li, Haoyun Gao, Chengzhang Wu, and Miklos A. Vasarhelyi. 2023. Extracting financial data from unstructured sources: Leveraging large language models. *Forthcoming in the Journal of Information Systems*. Available at SSRN: https://ssrn.com/abstract=4567607 or http://dx.doi.org/10.2139/ssrn.4567607.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *Preprint*, arXiv:2407.03618.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Preprint*, arXiv:2302.00083.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. 2024. Docfinqa: A long-context financial reasoning dataset. *Preprint*, arXiv:2401.06915.

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

Vikash Singh. 2017. Replace or retrieve keywords in documents at scale. *Preprint*, arXiv:1711.00046.

Mengyu Wang, Shay B Cohen, and Tiejun Ma. 2024. Modeling news interactions and influence for financial market prediction. *arXiv preprint arXiv:2410.10614*.

Mengyu Wang and Tiejun Ma. 2024. Mana-net: Mitigating aggregated sentiment homogenization with news weighting for enhanced market prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2379–2389.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *Preprint*, arXiv:2303.17564.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023a. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Junhan Yang, Zheng Liu, Chaozhuo Li, Guangzhong Sun, and Xing Xie. 2023b. Longtriever: a pre-trained long text encoder for dense document retrieval. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3655–3665. Association for Computational Linguistics.

Tan Yu, Anbang Xu, and Rama Akkiraju. 2024. In defense of rag in the era of long-context language models. *Preprint*, arXiv:2409.01666.

Bit Yuan et al. 2021. Finbert-qa: Financial question answering with pre-trained bert language models. https://github.com/yuanbit/FinBERT-QA. Accessed: 2025-04-29.

Boyu Zhang, Hongyang Yang, Tianyu Zhou, Ali Babar, and Xiao-Yang Liu. 2023. Enhancing financial sentiment analysis via retrieval augmented large language models. *Preprint*, arXiv:2310.04027.

Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. 2025. Mix-of-granularity: Optimize the chunking granularity for

retrieval-augmented generation. *arXiv preprint arXiv:2406.00456*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*.

A Potential Risks

FinGEAR restricts retrieval strictly to 10-K filings, thereby reducing the risk of hallucinated content in high-stakes financial applications. Nonetheless, two primary risks remain.

First, the system may occasionally fail to retrieve the most relevant disclosure due to semantic drift, ambiguous query interpretation, or traversal limitations. While this challenge is not unique to automated systems—human readers are similarly affected—it underscores the importance of improving query-to-structure alignment.

Second, although FinGEAR effectively retrieves relevant context, it does not constrain the generation behavior of downstream language models. As a result, models may still produce incorrect or speculative reasoning based on retrieved evidence, potentially leading to confirmation bias if users do not critically assess generated answers.

FinGEAR is intended to serve as a retrieval-first component within broader financial QA workflows. Its outputs should be interpreted in conjunction with the retrieved content, and human verification is recommended to ensure decision-making accuracy in professional settings.

B Implementation Details

Baselines. All baseline systems are based on publicly available implementations. General RAG[3] and Self-RAG[4] are based on the LangChain implementations. RAPTOR[5] and LightRAG[6] use official GitHub repositories.

API Costs. FinGEAR uses external APIs for LLM access, while financial embeddings are computed locally without cost. Using `gpt-4o-mini`, the estimated one-time cost to construct FinGEAR's index for a 127-page 10-K filing (76,114 words) is approximately \$0.11: \$0.057 for Summary Tree construction and \$0.029 for Question Tree construction. Answering a FinQA query costs approximately \$0.00048 (Top-5 retrieval), \$0.00076 (Top-10), and \$0.00100 (Top-15).

C Baseline Configurations

All baseline models are evaluated under consistent retrieval settings with Top-$k \in \{5, 10, 15\}$. Each baseline retrieves from the same pool of recovered full 10-K filings (Section 4) or uniformly segmented chunks, without curated summaries. All systems—including FinGEAR and baselines—use the same reader model (`gpt-4o-mini`) for answer generation to isolate retrieval effects.

General RAG. Vanilla BM25 + dense retrieval (default LangChain recipe).[7] Dense encoder: `text-embedding-ada-002` unless noted.

Self-RAG. Self-evaluation guided retrieval using the LangGraph implementation.[8] Default settings for critique and selection.

RAPTOR. Hierarchical summarization and tree retrieval using the official repository.[9] Default depth policy; flat chunking inputs; no domain priors.

LightRAG. Official implementation in "mix" mode (keyword + dense with lightweight neighborhood expansion).[10] Default neighborhood expansion; entity/relation indexing as released.

GraphRAG. We use the official local–global community summary traversal with default community construction and inference-time community propagation settings.

D Embedding Fine-Tuning Details

FinGEAR fine-tunes sentence embeddings for two key financial retrieval tasks using the base `finance-embeddings-investopedia` model from FinLang (FinLang, 2025):

Lexicon-based Fine-Tuning: Trained on FinRAD (Ghosh et al., 2021) (public) by pairing financial lexicons with disclosure sentences that contain

[3] https://python.langchain.com/docs/tutorials/rag/

[4] https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_self_rag/

[5] https://github.com/parthsarthi03/raptor/tree/master

[6] https://github.com/HKUDS/LightRAG

[7] https://python.langchain.com/docs/tutorials/rag/

[8] https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_self_rag/

[9] https://github.com/parthsarthi03/raptor/tree/master

[10] https://github.com/HKUDS/LightRAG

or exclude them. This optimizes embedding alignment for lexicon-based indexing.

Hierarchical Tree Fine-Tuning: Trained on FinQA (Chen et al., 2021) using question-passage pairs to improve semantic clustering and navigation within the Summary and Question Trees.

Evaluation Metrics. We use cosine-based Pearson and Spearman correlations for *FinRAD*, measuring embedding similarity between financial terms and disclosure sentences. For *FinQA*, we apply NDCG@10 to assess ranked retrieval performance per query.

Loss Functions. We use Matryoshka Representation Learning (Henderson et al., 2023) (dim=768) and `MultipleNegativesRankingLoss` to support semantically rich representations across both clustering and ranking stages.

Hardware. All embedding are fine-tuned (FinRAD and FinQA) on a machine with a NVIDIA A100-SXM4-80GB GPU. Training details can be found in Section K.

Dataset	Metric	Baseline	Fine-Tuned
FinRAD	Pearson Cosine	0.1082	**0.4063**
	Spearman Cosine	0.0904	**0.3655**
FinQA	NDCG@10 (Cosine)	0.0282	**0.2902**

Table 5: Embedding evaluation before and after fine-tuning on FinRAD and FinQA. Metrics reflect task-specific objectives: similarity alignment for FinRAD, and ranked retrieval quality for FinQA.

E FLAM Clustering Configuration

The FLAM (Financial Lexicon-Aware Mapping) module utilizes a clustering pipeline to group semantically related financial terms, facilitating more robust and lexicon-guided retrieval. The configuration follows three key stages: (1) Embedding Encoding is performed using sentence embeddings that have been fine-tuned on financial QA datasets to capture domain-specific semantic similarity; (2) Dimensionality Reduction is applied via UMAP, reducing embeddings to a 10-dimensional space using cosine similarity as the distance metric to preserve neighborhood structure; (3) Clustering is conducted using Gaussian Mixture Models (GMM) with a maximum of 50 clusters and a convergence threshold set to 0.1, enabling soft assignments of terms into semantically coherent groups.

F Token-Level Statistics

To balance retrieval efficiency with financial context fidelity, FinGEAR applies a fixed-length chunking strategy prior to indexing. Each chunk contains approximately 2,000 tokens with a 100-token overlap to maintain contextual continuity, which is same across all baseline. Tree-based indexing is then applied to support structure-aware retrieval.

"Tokens per summary node" refers to the average length of high-level disclosure summaries used in the Summary Tree, while "tokens per page node" refers to the full-length segments prior to hierarchical expansion. For latency and deployment details, refer to Appendix G.

Statistic	Min	Mean	Max
Tree depth (max)	2	3.14	4
Tree depth (mean)	1.22	1.43	1.80
Total node count	27	254.86	1322
Internal node count	0	17.84	85
Leaf node count	27	237.02	1237
Page count	84	168.89	555
Document count	184	1342.77	7078
Tokens per document	291.98	331.03	382.23
Tokens per summary node	634.72	657.78	687.46
Tokens per query	16.16	17.27	18.60
Tokens per page node	598.71	2129.56	7021.38

Table 6: Token-level and structural statistics across the FinGEAR indexing pipeline.

Under this configuration, the total retrieval volume scales linearly with depth k as:

$$\text{Total tokens retrieved} \approx 2000 \times k.$$

G Practical Deployment Details

G.1 System Integration

FinGEAR is designed as a modular retrieval layer that integrates into RAG pipelines with minimal configuration effort. It exposes a unified retriever interface supporting both sparse and dense retrieval strategies, along with hybrid reranking. These strategies are fully configurable by depth k, retrieval type, and domain specificity.

The system supports hierarchical retrieval via two independent traversal mechanisms: the Summary Tree, which provides disclosure-level navigation based on financial document structure, and the Question Tree, which guides retrieval using query semantics through sub-question expansion. This separation enables controlled traversal of financial disclosures at varying levels of abstraction.

FinGEAR is compatible with vector store backends such as FAISS for indexing and searching over domain-tuned embeddings, and its reranking modules are cross-encoder based for contextual precision. While optimized for structured regulatory disclosures like 10-K filings, its traversal and retrieval logic can be extended to semi-structured corpora—such as earnings calls or ESG reports—by replacing FLAM with unsupervised segmentation or alternative indexing methods. The system optionally integrates with FAISS for dense indexing and LangChain for pipeline orchestration, though these components are modular and replaceable.

G.2 Latency Statistics

All evaluations were conducted on a MacBook Pro with an M3 Max chip and 64GB of RAM. FinGEAR demonstrates practical deployability on standard enterprise hardware, maintaining tractable inference latency suitable for real-world financial workflows.

H Error Analysis Examples

Table 7 presents paired examples of successful and imperfect retrievals from FinQA-aligned questions, illustrating how FinGEAR handles both precise matching and interpretable failure cases. For example, in response to the question *"What is the company's debt-to-equity ratio?"*, FinGEAR successfully surfaces the "Balance Sheet" section disclosing the necessary financial figures. In contrast, a failure case retrieves a "Capital Management" section that is conceptually related but lacks numeric content—highlighting a mismatch between strategic language and quantitative requirements.

Similarly, for *"What litigation is the company currently facing?"*, the system initially retrieves generic "Risk Factors" content due to shared legal phrasing, but the reranker recovers the correct "Legal Proceedings" section, demonstrating partial recovery. In other cases, such as the misretrieval of a "Commodity Risk" section when asked about foreign exchange exposure, errors stem from keyword ambiguity across structurally distinct topics.

These examples underscore the role of FinGEAR's hybrid architecture in surfacing relevant content while also providing interpretable signals for tracing failure modes. The system's transparency allows users and developers to adjust lexicon weights, reranking sensitivity, or structural traversal logic—enabling more robust deployment and auditability in financial applications.

I Multi-Component Ablation Study

To evaluate the interdependence of FinGEAR's core modules, we conduct a multi-component ablation study by disabling pairs among *FLAM*, the *Summary Tree*, and the *Question Tree*. This setup allows us to assess how retrieval quality degrades under partial configurations and whether FinGEAR's performance stems from isolated components or coordinated design.

Findings. Disabling multiple modules results in substantial, compounded performance drops across all retrieval metrics, consistently observed at depths $k = 5, 10, 15$. The largest declines in F1 score and relevancy occur when both *FLAM* and the *Question Tree* are removed, revealing their complementary roles in guiding retrieval semantically and contextually. When both *FLAM* and the *Summary Tree* are ablated, precision and recall drop sharply, indicating that lexical anchoring and hierarchical structuring are jointly essential for balancing coverage and specificity. Furthermore, these ablations introduce instability across depths—particularly in precision and recall—suggesting that without FLAM's lexicon-targeted control, the traversal process becomes overly uniform and sensitive to document variation. This leads to inconsistent candidate selection and noisier retrieval. These results confirm that FinGEAR's performance emerges from the coordinated integration of financial mapping, semantic indexing, and query-aware guidance—rather than from any individual module in isolation.

J Question-Type Ablation

To support analysis in Section 5.2.4, we provide the full per-question-type retrieval results in Table 10.

K Runtime and Preprocessing Analysis

Embedding Fine-Tuning. FinGEAR employs dense retrieval powered by sentence embeddings fine-tuned on financial data. We train two embedding models: one on FinQA for question–answer alignment, and one on FinRAD for lexicon-level semantic proximity. Both models are trained for 50 epochs. FinQA training (6,251 train / 1,147 test examples) completes in approximately 1,972 seconds, while FinRAD training (17,300 train / 5,190 test, sampled across filings) takes roughly 3,607 seconds. Training is performed on a single NVIDIA

Query	Successful Retrieval	Failure Case	Interpretation
What is the debt-to-equity ratio in 2022?	"Consolidated Balance Sheets" showing total debt and equity.	"Liquidity and Capital Resources" discussing debt repayment but omitting equity.	Partial match—missed denominator component due to narrative emphasis.
What was the YoY revenue growth for Q4?	"Results of Operations" section with quarterly revenue breakdown.	"Management Overview" with general trends but no figures.	High-level commentary lacks granularity—numeric context lost.
How much did R&D expenses increase year-over-year?	"Operating Expenses" table with line items for R&D across periods.	"Business Overview" noting investment in innovation but not amounts.	Lexical match on "R&D" misaligned with quantitative query intent.
What percentage of revenue came from the US?	"Segment Reporting" with revenue by geography.	"Market Strategy" outlining international expansion plans.	Structural mismatch: retrieved future projections instead of current data.
What is the gross margin for 2021?	"Income Statement" showing gross profit and revenue.	"Risk Factors" discussing cost pressures without figures.	Terminology overlap led to non-numerical section—filtered out post-reranking.

Table 7: Examples of FinGEAR's retrieval outcomes for representative FinQA-style queries. Failure cases illustrate interpretable mismatches from lexical ambiguity, structural misalignment, or contextual drift—often mitigated by reranking.

Table 8: Multi-component ablation results. Disabling pairs of modules leads to compounded performance degradation, underscoring the synergistic design of FinGEAR and the importance of coordinated retrieval.

Ablation Setting	Precision			Recall			F1 Score			Relevancy		
	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$	$k=5$	$k=10$	$k=15$
Full FinGEAR	**0.79**	**0.76**	**0.72**	**0.61**	**0.62**	**0.65**	**0.69**	**0.68**	**0.68**	**0.50**	**0.64**	**0.62**
No FLAM + No Summary Tree	0.48	0.59	0.51	0.40	0.30	0.33	0.44	0.40	0.40	0.25	0.34	0.34
No FLAM + No Question Tree	0.48	0.58	0.52	0.40	0.31	0.36	0.44	0.40	0.43	0.26	0.34	0.34

Metric	Value (seconds)
Retrieval Latency ($k=5$)	12.05
Retrieval Latency ($k=10$)	18.28
Retrieval Latency ($k=15$)	22.42
Average Retrieval Latency	17.58[(a)]

Table 9: Latency statistics for FinGEAR deployment. [(a)] Includes tree traversal, FLAM scoring, and reranking averaged across depths.

A100-SXM4-80GB GPU. These embeddings are applied throughout the retrieval pipeline, including document chunking, hierarchical traversal, and candidate scoring.

Preprocessing Pipeline. The full preprocessing workflow comprises: (1) keyword mapping, (2) keyword tree construction, and (3) hierarchical clustering for the Summary and Question Trees.

Keyword Mapping, implemented using spaCy's `PhraseMatcher`, identifies relevant domain terms and runs between 53.5 and 434.6 seconds per document (mean: 248.9 seconds). **Keyword Tree Construction** is fast, requiring an average of 7.6 seconds per company. **Hierarchical Clustering and Summarization**, including UMAP-based dimensionality reduction and GMM clustering, builds the Summary and Question Trees. Across 10 filings, total runtime ranges from 243.2 to 602.6 seconds per document, with a mean of 408.2 seconds—primarily driven by the Summary Tree's iterative summarization.

Deployment Considerations. All preprocessing steps are one-time operations per document and can be cached for repeated use. FinGEAR supports dual-tree traversal and modular re-indexing, making it scalable for real-world financial workloads. Runtime remains stable across filings of similar size, and preprocessing cost is amortized across multiple downstream queries.

L Glossary of Terms and Notation

This appendix consolidates the terms, symbols, and procedure names used throughout the paper for quick reference. It denotes definitions for the FinGEAR components (e.g., traversal logic, and scoring functions). Unless stated otherwise, scalars are in italics (e.g., k, w_i), vectors are in bold, and Item labels follow SEC notation (Item 1, Item 1A, Item

Table 10: Categorical = yes/no answers, Numerical = numeric answers, Simple = one reasoning step, Complex = multiple reasoning steps

Question Type	Precision			Recall			F1 Score			Relevancy		
	$k = 5$	$k = 10$	$k = 15$	$k = 5$	$k = 10$	$k = 15$	$k = 5$	$k = 10$	$k = 15$	$k = 5$	$k = 10$	$k = 15$
Numerical	0.79	0.76	0.71	0.61	0.62	0.64	0.69	0.68	0.68	0.50	0.64	0.63
Categorical	0.83	0.92	0.87	0.71	0.72	0.82	0.77	0.81	0.86	0.46	0.48	0.49
Simple	0.79	0.76	0.73	0.62	0.64	0.67	0.70	0.70	0.70	0.51	0.62	0.58
Complex	0.78	0.76	0.72	0.59	0.59	0.61	0.67	0.67	0.66	0.48	0.65	0.67

7). For mechanics in the pipeline, see Sections 3 and 3.2; the table below serves as a concise lookup.

M Prompt Design

All prompts are used in zero-shot mode unless otherwise stated. Structured examples may be incorporated in future versions to enhance control and reproducibility. Here are three specialized prompts:

M.1 Summarization Prompt

```
The content provided below is a subset of a 10-K filing. The 10-K report is a comprehensive document outlining the company's financial performance, including revenue, expenses, and profits. Your task is to generate a detailed summary using only the provided content, without embellishment. Summarize main topics, key insights (5-7), and unusual observations (1-2). Use clear paragraphs and Markdown headings.
```

M.2 Title Generation Prompt

```
Generate a title for a subsection of a 10-K report based on the provided summary.
```

M.3 Question Generation Prompt

```
Suppose you are a financial analyst. Generate {num_questions} questions based on the provided summary, focusing on financial aspects and factual details.
```

Term / Symbol	Definition
UMAP	Dimensionality reduction used before clustering; preserves local neighborhoods for tree construction. We use cosine distance with output dimension 10 (see Section 5.1).
GMM	Gaussian Mixture Models used for soft clustering over UMAP embeddings to form internal tree nodes; max components 50; convergence threshold 0.1.
FLAM	*Financial Lexicon-Aware Mapping*. Clusters financial terms corpus-wide and computes Item weights to guide global allocation (Section 3.2). Default weighting: Relative Frequency.
Summary Tree	Content hierarchy within an Item. Internal nodes are summaries of clustered chunks; leaves are original text chunks (Section 3.1.1).
Question Tree	Mirrors the Summary Tree topology; nodes store LLM-generated sub-questions embedded in the same space as user queries; leaves reference the same chunk IDs (Section 3.1.1).
k	Total retrieval budget per query (Top-k evaluation).
k^*	Per-Item budget after FLAM weighting; $\sum_i k_i^* = k$ (Section 3.2).
Semantic traversal	Top-down navigation of a tree: score parent's children, select the best, and descend until leaves or max depth (2).
Hybrid scoring	Within-Item node scoring that combines BM25 over summaries (sparse) and cosine similarity over embeddings (dense).
Reranking (Stage 1)	Cross-tree reranking that jointly scores candidates from Summary and Question Trees using `BAAI/bge-reranker-large`.
Reranking (Stage 2)	Cross-Item reranking over the pooled top spans from all Items to prioritize globally informative answers.

Table 11: Glossary of key components, symbols, and procedures used in FinGEAR.