EKRAG: Benchmark RAG for Enterprise Knowledge Question Answering

Tan Yu*, Wenfei Zhou*, Lei Yang, Aaditya Shukla, Meenakshi Madugula, Pritam Gundecha
Nick Burnett, Anbang Xu, Vishal Seth, Tamar Bar, Rama Akkiraju ,Vivienne Zhang
NVIDIA
Santa Clara, California, USA

Abstract

Retrieval-augmented generation (RAG) offers a robust solution for developing enterprise internal virtual assistants by leveraging domain-specific knowledge and utilizing information from frequently updated corporate document repositories. In this work, we introduce the Enterprise-Knowledge RAG (EKRAG) dataset to benchmark RAG for enterprise knowledge question-answering (QA) across a diverse range of corporate documents, such as product releases, technical blogs, and financial reports. Using EKRAG, we systematically evaluate various retrieval models and strategies tailored for corporate content. We propose novel embedding-model (EM)-as-judge and ranking-model (RM)-as-judge approaches to assess answer quality in the context of enterprise information. Combining these with the existing LLM-as-judge method, we then comprehensively evaluate the correctness, relevance, and faithfulness of generated answers to corporate queries. Our extensive experiments shed light on optimizing RAG pipelines for enterprise knowledge QA, providing valuable guidance for practitioners. This work contributes to enhancing information retrieval and question-answering capabilities in corporate environments that demand high degrees of factuality and context awareness.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable zero-shot reasoning capabilities (Brown et al., 2020; Achiam et al., 2023). However, their static knowledge base, acquired during pre-training, poses significant challenges in generating factual, timely, and salient responses to ambiguous and complex queries, particularly in corporate environments. This limitation is critical in enterprise settings, where accuracy and up-to-date information are paramount for a functional product. Inaccurate or outdated responses to time-sensitive queries not only erode user trust but can render AI-assisted tools impractical. For instance, using obsolete data to answer financial queries like "What was the company's last quarter revenue?" could lead to misinformed decision-making, if not panic in the stock market, even if the information was accurate once upon a time.

Retrieval-augmented generation (RAG) (Guu et al., 2020; Lewis et al., 2020) emerges as a promising solution, integrating external knowledge from domain-specific corpora to potentially generate up-to-date, factually correct answers. It is particularly suited for corporate internal chatbots, facilitating employees and external investors access to current company policies, procedures, and knowledge across public facing corporate documents.

Despite its advantages, RAG presents unique evaluation challenges compared to standalone LLMs. Their answer quality depends not only on the LLM's pre-trained knowledge and reasoning capabilities but also on the relevance of retrieved context and the model's ability to integrate and synthesize information. While recent works (Adlakha et al., 2023; Chen et al., 2024b; Gao et al., 2023; Es et al., 2023; Xiong et al., 2024) have evaluated RAG systems in general domains, assessing both answer correctness and alignment with retrieved context, the evaluation of RAG in enterprise knowledge applications remains largely unexplored.

To address this gap, we introduce the Enterprise-Knowledge RAG (EKRAG) dataset, comprising 1,347 manually curated questions, designed to systematically evaluate the influence of each RAG component on enterprise knowledge question-answering over five core question types in multi-hop settings. Using this benchmark, our extensive evaluations of various retrieval models and strategies reveal that well-known techniques like HYDE and hybrid search do not significantly improve enterprise knowledge retrieval accuracy, while

*Equal contribution.

some straightforward mechanisms such as multi-embedding vector achieve excellent performance.

Furthermore, we propose novel evaluation methodologies: embedding-model-as-judge and ranking-model-as-judge approaches, complementing the prevailing LLM-as-judge technique. Our comprehensive investigation provides insights into optimizing RAG pipelines for enterprise knowledge systems, offering a nuanced understanding of answer quality that could lead to more robust and reliable RAG implementations in real-world corporate applications.

2 Related Work

Adlakha, BehnamGhader, Lu, Meade, and Reddy (2023) utilizes the existing QA datasets and evaluates the answer quality of RAG along the dimensions of correctness and faithfulness. To be specific, the correctness dimension reveals the relevance between the response and the ground-truth answer. On the other hand, faithfulness measures the relevance between the answer and the retrieved context to evaluate model's capability to ground the retrieved context. RGB (Chen et al., 2024b) evaluates RAG from four aspects including noise robustness, negative rejection, information integration, and counterfactual robustness. ALCE (Gao et al., 2023) proposes evaluation metrics of RAG along three dimensions including fluency, correctness, and citation quality. RAGAS (Es et al., 2023) develops an automated evaluation pipeline by prompting LLMs and evaluate RAG from the dimensions of faithfulness, answer relevance and context relevance. MIRAGE (Xiong et al., 2024) benchmarks different RAG solutions in the field of medicine and ablates the influence of each component on the overall answer quality from a multi-dimensional perspective. CRAG (Yang et al., 2024) proposes a comprehensive RAG benchmark consisting of questions across five domains and eight categories.

3 Enterprise-Knowledge RAG Dataset

3.1 Overview

The Enterprise-Knowledge RAG (EKRAG) dataset (v.1) is an expert-curated, comprehensive evaluation resource for Retrieval-Augmented Generation (RAG) systems operating on corporate documents. Developed by Corporate's IT and data teams, EKRAG serves as a benchmark for assessing agentic RAG systems, particularly those requiring a high degree of factuality in reporting corporate financial performance and product information. Key features of EKRAG include:

1. **Composition:** 1,348 human curated data points, each consisting of a query, relevant context chunk(s), referenced document, and ground truth answer, complemented by associated metadata. These are derived from a diverse corpus of 5,000 documents, encompassing Corporate technical blogs, news releases, SEC filings, and leadership communications.

2. **Manual Curation:** Annotations are provided by a team of 14 human experts with backgrounds in business and finance, ensuring high-quality, domain-specific data points that reflect real-world complexity and nuance.

3. **Comprehensive Task Coverage:** The dataset facilitates evaluation across five core tasks of varying complexities:

   - Extractive Question Answering
   - Abstractive Question Answering
   - Summarization
   - Financial Numerical Reasoning
   - No Answer Questions

   Each task type encompasses questions of varying complexities, characterized by 1) modality utilization (text, tables, or both), 2) intra-document reasoning (synthesizing information from non-contiguous chunks), and 3) inter-document reasoning (integrating information across multiple documents). These multi-hop tasks simulate real-world, enterprise-level scenarios, evaluating both the RAG *retriever's* ability to fetch complex information from diverse sources and the *generator's* capacity to synthesize coherent responses from retrieved chunks. This design assesses the system's end-to-end capability in handling practical information retrieval and integration problems in a corporate setting.

3.2 Annotation Methodology

3.2.1 Data Sourcing

Reference Documents: The EKRAG dataset is derived from 5,000 publicly available Corporate documents, including web pages, earnings call transcripts, and SEC reports in PDF, HTML, .docx, and .txt formats. These documents are categorized into four main groups: Corporate News and Blogs, Corporate Technical Blogs, leadership communications (including public fireside chats), and SEC

10-K and 8-K filings. The current v.1 of the dataset utilizes only the Corporate News and Blogs and Corporate Technical Blogs categories.

Queries: To ensure real-world relevance, we analyzed approximately 200 queries made by corporate finance analysts during the initial release of the chatbot. These queries were rigorously categorized to create a comprehensive taxonomy, which formed the basis for the annotation guidelines. This approach ensured alignment between annotators' efforts and real-world use cases.

3.2.2 Data Preprocessing

To facilitate human annotation, the corpus underwent a thorough cleaning process. Using the Beautiful Soup package, the team extracted crucial metadata (e.g., publication dates and titles) from HTML and PDF files while removing extraneous content such as CSS. Rigorous filtering criteria were applied, resulting in $3,620$ high-quality documents suitable for annotation.

For multi-document annotations, the team employed topic modeling and preprocessing techniques to group similar documents together, resulting in clusters of 1-4 topically related documents.

Throughout the annotation process, the annotators provided active feedback. They were given the option to skip document groupings that are 1) low-quality; 2) impossible to derive coherent queries. The annotation process is described in Appendix B.

4 RAG Pipeline

4.1 Indexing

We denote the cropped chunks by $\{c_i\}_{i=1}^N$ where N denotes the number of chunks. For each chunk, the embedding vector is extracted by

$$\mathbf{c}_i = f_{\text{emb}}(c_i), \tag{1}$$

where $f_{\text{emb}}(\cdot)$ denotes the embedding model. Since the embedding of chunks are independent to the query, a retrieval system normally precomputes chunk embedding vectors $\{\mathbf{c}_i\}_{i=1}^N$. In the retrieval phase, we just need compute the query's embedding online and compare it with the pre-computed chunk embedding vectors to retrieve the most relevant chunks. When N is large, to speed up the retrieval, approximated nearest neighbor (ANN) search methods such as Hashing, Product Quatization, and HNSW are used in indexing.

4.2 Embedding Models

Dense Embedding. Trationally, the dense embedding model adapts a bi-encoder Transformer architecture (Reimers and Gurevych, 2019). To be specific, the query encoder maps the textual query into the query embedding:

$$\mathbf{q} = \text{bi-enc}_{\text{query}}(q). \tag{2}$$

In parallel, the doc encoder maps a text chunk into the chunk embedding:

$$\mathbf{c} = \text{bi-enc}_{\text{doc}}(c). \tag{3}$$

The relevance between the query and the chunk is measured by the distance/similarity between the query embedding $\mathbf{q}$ and the chunk embedding $\mathbf{c}$. Recently, with the emergence of LLM, ecoder-based embedding models (Lee et al., 2024) are obtained by fine-tuning immediate output of LLM, achieving promising performance in retrieval.

Sparse Embedding often represents the occurrence or statistics of specific features (*e.g.*, words or n-grams). It was widely used in traditional information retrieval and text classification. Traditionally, TF-IDF (Ramos et al., 2003) and BM25 (Robertson et al., 1995) are widely used term-weighting sparse embedding methods. Recently, some works such as SPLADE (Formal et al., 2021) and BGE-M3 (Chen et al., 2024a) utilize foundational BERT architecture to generate sparse embedding vectors.

4.3 Retrieval.

Hybrid search is a widely used strategy for RAG applications (Finardi et al., 2024). It conducts dense retrieval and the sparse retrieval simultaneously. To be specfic, given a query q, we retrieve t most relevant queries based on dense embedding:

$$[c_{i_1}, \cdots, c_{i_t}] = \text{dense}(q, \{c_i\}_{i=1}^t). \tag{4}$$

In parallel, another t most relevant queries are retrieved based on sparse embedding:

$$[c_{j_1}, \cdots, c_{j_t}] = \text{sparse}(q, \{c_i\}_{i=1}^t). \tag{5}$$

Then $2t$ retrieved chunks from both sparse and dense retrieval are de-duplicated and re-ranked by a ranking model, and top s ranked chunks are as the context for LLM to generate the answer:

$$[c_{k_1}, \cdots, c_{k_s}] = \text{rerank}(\{c_{i_l}\}_{l=1}^t, \{c_{j_l}\}_{l=1}^t) \tag{6}$$

Embedding Model	Recall@1	Recall@3	Recall@5	Recall@10
BM25 (Robertson et al., 1995)	0.004	0.010	0.023	0.074
NV-QA (Verma et al.)	0.277	0.540	0.651	0.752
Arctic-L (Merrick et al., 2024)	0.313	0.552	0.660	0.786
NV-EMB (Lee et al., 2024)	**0.360**	**0.618**	**0.742**	**0.839**

Table 1: The evaluation of the embedding models.

NV-QA	Arctic-L	NV-EMB	Recall@1	Recall@3	Recall@5	Recall@10
		✓	**0.360**	0.618	0.742	0.839
✓	✓		0.325	0.590	0.697	0.808
✓		✓	0.350	0.640	0.746	0.858
	✓	✓	0.358	0.639	**0.751**	**0.863**
✓	✓	✓	0.348	**0.648**	0.748	0.857

Table 2: The results from combining multiple embedding models.

4.4 Reranking

Cross-encoder. In the reranking phase, we normally takes a cross-encoder architecture. Different from bi-encoder model used in embedding model, cross-encoder takes a query-chunk pair (q, c_{k_i}) as the input and output a relevance score:

$$r_i = \text{cross-enc}(q, c_{k_i}). \quad (7)$$

Chunks $\{c_{k_i}\}_{i=1}^s$ are sorted by relevance scores. Detailed discussions on the cross-encoder and bi-encoder are in Appendix A.

4.5 Answer Generation

We fill the query and the chunks $\{c_{k_i}\}_{i=1}^s$ from reranking into the prompt template and feed the prompt into a LLM to generate answer.

5 Benchmark on Retrieval

Metrics. There are multiple metrics to evaluate the retrieval such as mAP, Precision@K, Recall@K, NDCG. In RAG scenarios, Recall@K is the most cruial metric, which reveals the coverage of relevant information in the retrieved chunks. Thus, by default, we use Recall@K as the evaluation metric.

5.1 Ablation on Embedding Models

Ablation on embedding models. We compare 4 models, including BM25 (Robertson et al., 1995), NV-QA (Verma et al.), Arctic-L (Merrick et al., 2024) and NV-EMB (Lee et al., 2024). NV-EMB is a decoder-only large language model (LLM)-based embedding model. Compared with BERT-based models, it achieves significantly higher accuracy on public benchmarks but takes much high computational cost. As shown in Table 1, NV-EMB significantly outperforms other embedding models.

Multi-embedding vector. To make use of multiple embedding model simultaneously, a straightforward method is to concatenate the embedding vectors from multiple models into a long vector, which we term as multi-embedding vector. For example, we denote the query/chunk embedding from the first embedding model by $\mathbf{q}_1/\mathbf{c}_1$ and that from the second embedding model by $\mathbf{q}_2/\mathbf{c}_2$. We normalize and concatenate the query/chunk embedding vectors into a multi-embedding vector:

$$\begin{aligned}\bar{\mathbf{q}} &= [\alpha_1 \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|_2}, \alpha_2 \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|_2}], \\ \bar{\mathbf{c}} &= [\alpha_1 \frac{\mathbf{c}_1}{\|\mathbf{c}_1\|_2}, \alpha_2 \frac{\mathbf{c}_2}{\|\mathbf{c}_2\|_2}],\end{aligned} \quad (8)$$

where α_1 and α_2 are pre-defined constants to weight the contributions of each embedding vector. By default, we set $\alpha_1 = \alpha_2 = 1$. Empercally, we could assign a higher weight to a better embedding. The relevance between the query and the chunk is computed by the dot product between $\bar{q}$ and $\bar{c}$:

$$\text{r}(q, c) = \langle \bar{\mathbf{q}}, \bar{\mathbf{c}} \rangle = \alpha_1^2 \text{cos}(\mathbf{q}_1, \mathbf{c}_1) + \alpha_2^2 \text{cos}(\mathbf{q}_2, \mathbf{c}_2). \quad (9)$$

We evaluate multi-embedding vectors in Table 2. As shown, multi-embedding vector using Arctic-L and NV-EMB achieves the highest recall@5. By default, we use this setting for embedding.

Hybrid Search. In Table 4, we compare the hybrid search with methods using solely dense-embedding. To be specific, dense-based method adopts a multi-embedding settings using both NV-EMB and Archtic-L. As shown, hybrid search does not bring considerable improvement for recall. Thus, by default, we exclude hybrid search.

5.2 Retrieval boosting strategies

Average query expansion (AQE) (Carpineto and Romano, 2012) refines the query embedding by

LLM for HYDE	Recall@1	Recall@3	Recall@5	Recall@10
w/o	0.360	0.618	**0.742**	0.839
Mistal-7B (Jiang et al., 2023)	0.358	**0.625**	0.736	0.836
Llama3-8B (AI@Meta, 2024)	0.358	0.621	0.737	0.830
Gemma2-7B (Team, 2024)	0.351	0.621	0.737	0.838
Llama3-70B (AI@Meta, 2024)	**0.361**	0.624	0.740	**0.842**
Mixtral-8x7B (Jiang et al., 2024)	0.351	0.621	**0.742**	0.836
Mixtral-8x22B (Mistral AI team, 2024)	0.358	0.629	0.740	0.838

Table 3: The evaluation of the HYDE.

	Recall@1	Recall@3	Recall@5
Dense	0.403	0.688	0.707
Hybrid	0.404	0.687	0.707

Table 4: Evaluation on hybrid search.

n	0	1	2	3	4
NV-QA	0.651	0.660	0.647	0.652	0.662
Arctic-L	0.660	0.689	0.667	0.663	0.652
NV-EMB	0.742	0.725	0.716	0.721	0.731

Table 5: The influence of average query expansion.

the embeddings from the top-retrieved chunks. To be speicifc, we denote the original query embedding by $\mathbf{q}$ and the embeddings for the retrieved top n chunks as $\{\mathbf{c}_i\}_{i=1}^n$. AQE generates the revised query embedding by suming up the original query embedding with each chunk embedding $\mathbf{c}_i$:

$$\hat{\mathbf{q}} = \frac{\mathbf{q} + \sum_{i=1}^n (\mathbf{c}_i)}{n+1} \tag{10}$$

We evaluate the influence of AQE on the retrieval Recall@5. As shown in Table 5, when we set $n = 1$, AQE consistently improves Recall@5 for NV-QA and Arctic-L embedding models, but it drops Recall@5 for the NV-EMB. Considering the computation cost, we do not use AQE, by default.

HYDE (Gao et al., 2022) utilizes the LLM's knolwedge to generate a hypothetical document $d = \text{LLM}(q)$ for the query q. Then the embedding model encodes document d into the embedding vector $\mathbf{d} = f_{\text{emb}}(d)$. The refined query embedding $\hat{\mathbf{q}}$ is obtained by summing up $\mathbf{q}$ and $\mathbf{d}$:

$$\hat{\mathbf{q}} = \alpha\mathbf{q} + (1-\alpha)\mathbf{d}, \tag{11}$$

where α is a pre-defined positive constant ($\alpha \in [0, 1]$) controlling the contribution from each component. We explore multiple open-source LLMs including Mistal-7B (Jiang et al., 2023), Llama3-8B (AI@Meta, 2024), Gemma2-7B (Team, 2024), Llama3-70B (AI@Meta, 2024), Mixtral-8x7B (Jiang et al., 2024) and Mixtral-8x22B (Mistral AI team, 2024) to generate the hypothetical document. As shown in Table 3, HYDE could not consistently improve the recall. Thus, we exclude HYDE in retrieval pipeline.

5.3 Reranking

By default, we use QA-Mistral-4B(NVIDIA AI, 2024) for reranking. As shown in Table 6, the reranking model consistently and significantly improves the recall of all embedding models. Therefore, we use reraking model in default settings.

6 Benchmark on Answer Generation

6.1 Metrics

Correctness measures the alignment between the generated answer and the ground-truth answer.
Relevance quantizes the relevance between the generated answer and the ground-truth context.
Faithfulness measures the relevance between the retrieved context and generated answer.
We design three types of judge mechanisms including embedding-model-as-judge, ranking-model-as-judge and LLM-as-judge.

6.2 Embedding-model-as-judge

We denote the embedding of the ground-truth answer by $\mathbf{a}_g$, that of answer generated from RAG by $\mathbf{a}_r$, the embedding of the ground-truth context by $\mathbf{c}_g$ and that of the retrieved context by $\mathbf{c}_r$. When using embedding model as judge, correctness/relevance/faithfulness is measured by the cosine similarity between embeddings:

$$\begin{aligned}
\text{C}_{\text{emb}} &= \frac{\langle \mathbf{a}_r, \mathbf{a}_g \rangle}{\|\mathbf{a}_r\|_2 \|\mathbf{a}_g\|_2}, \\
\text{R}_{\text{emb}} &= \frac{\langle \mathbf{a}_r, \mathbf{c}_g \rangle}{\|\mathbf{a}_r\|_2 \|\mathbf{c}_g\|_2}, \\
\text{R}_{\text{emb}} &= \frac{\langle \mathbf{a}_r, \mathbf{c}_r \rangle}{\|\mathbf{a}_r\|_2 \|\mathbf{c}_r\|_2}.
\end{aligned} \tag{12}$$

6.3 Ranking-model-as-judge

We denote the labeled groundtruth answer by a_g, the answer generated from RAG by a_r, the groundtruth context by c_g and the embedding of the retrieved context by c_r. We denote the cross(x, y) as the cross-encoder model maps a pair of texts (x, y) into a relevance score $s \in [-\infty, +\infty]$. The higher score means that x is more relevant with

Rerank	NV-QA		Arctic-L		NV-EMB		Arctic-L + NV-EMB	
	Recall@1	Recall@5	Recall@1	Recall@5	Recall@1	Recall@5	Recall@1	Recall@5
w/o	0.277	0.651	0.313	0.552	0.360	0.742	0.358	0.751
QA-Mistral-4B	0.389	0.757	0.391	0.767	0.401	0.782	0.403	0.787

Table 6: The influence of ranking model.

Models	Embedding-model-as-judge			Ranking-model-as-judge			LLM-as-judge		
	Corre.	Relev.	Faith.	Corre.	Relev.	Faith.	Corre.	Relev.	Faith.
Mistral-7B	0.789	**0.747**	0.707	0.891	0.693	0.668	0.925	0.894	0.977
Llama3-8B	0.779	0.744	0.724	0.880	0.691	**0.692**	**0.945**	**0.915**	0.976
Mixtral-8x7B	0.792	0.738	0.691	0.891	0.684	0.651	0.913	0.875	0.963
Mixtral-8x22B	0.769	0.760	**0.742**	0.885	**0.698**	0.674	0.939	0.912	0.977
Llama3-70B	**0.793**	0.739	0.694	**0.894**	0.693	0.695	0.944	**0.915**	**0.978**

Table 7: The evaluation results on the generated answers from multiple open-source LLMs.

y. To normalize the score to the range $[0, 1]$, we process the relevance score by a sigmoid function: $\hat{s} = \text{sigmoid}(s)$. We define the correctness/relevance/faithfullness as below:

$$\begin{aligned} C_{\text{rank}} &= \text{sigmoid}(\text{cross}(a_r, a_g)), \\ R_{\text{rank}} &= \text{sigmoid}(\text{cross}(a_r, c_g)), \\ F_{\text{rank}} &= \text{sigmoid}(\text{cross}(a_r, c_r)). \end{aligned} \tag{13}$$

6.4 LLM-as-judge

We design the prompt templates for correctness, relevance and faithfulness, respectively. Given the query q, ground-truth answer a_g, answer generated from RAG a_r, ground-truth context by c_g, retrieved context by c_r, we fill them into the pre-defined templates, which is further feed into an LLM:

$$\begin{aligned} C_{\text{LLM}} =& \text{LLM}(\text{Template}_{\text{C}}(q, a_r, a_g)) \\ R_{\text{LLM}} =& \text{LLM}(\text{Template}_{\text{R}}(q, a_r, c_g)) \\ F_{\text{LLM}} =& \text{LLM}(\text{Template}_{\text{F}}(q, a_r, c_r)). \end{aligned} \tag{14}$$

Since LLM is not good at evaluating outputs in continuous range, we prompt the LLM to output a integer score within $[1, L]$ and divide the integer score by L to normalize it. By default, we use Mistral-Large as the judge.

6.5 Experimental results

Ablation on LLMs for answer generation. We evaluate the generation capabilities across a spectrum of Large Language Models (LLMs). The assessment includes smaller-scale models such as Mistral-7B and Llama3-8B, as well as larger-scale models including Mixtral-8x7B, Mixtral-8x22B, and Llama3-70B. The comparative results, presented in Table 7, reveal that the larger-scale models do not demonstrate a substantial performance advantage over their smaller counterparts in this specific task. Given these findings, and taking into account computational efficiency, we have opted to utilize Llama3-8B as our default model for answer generation. This choice represents an optimal balance between performance and resource utilization in our experimental framework.

# chunks	0	1	3	5	10
EM-as-J	0.717	0.764	0.777	0.779	0.775
RM-as-J	0.772	0.861	0.879	0.880	0.876
LLM-as-J	0.630	0.902	0.940	0.945	0.945

Table 8: Impact of the count of chunks on correctness.

Influence of the retrieved chunks. Table 8 illustrates the impact of the number of retrieved documents on answer correctness. The results demonstrate that the absence of retrieved chunks significantly diminishes answer correctness compared to scenarios where chunks are utilized, thus validating the efficacy of RAG over standalone LLMs. As the number of retrieved chunks increases from 1 to 5, we observe a consistent improvement in answer accuracy. However, this trend plateaus and potentially reverses when the number of chunks increases from 5 to 10. This phenomenon can be attributed to the introduction of extraneous information as the chunk count rises, which may impede the generation of accurate responses. These findings suggest an optimal retrieval window that balances comprehensive context with focused relevance, highlighting the importance of judicious document retrieval in RAG systems.

7 Conclusion

Our comprehensive study on Retrieval-Augmented Generation (RAG) for enterprise knowledge question-answering has yielded several significant insights. Through the creation and utilization of the EKRAG dataset, comprising 3200 manually curated questions, we have conducted an extensive evaluation of various components within the RAG pipeline. The insights gained from this study have broad implications for the optimization of RAG pipelines in enterprise knowledge management systems. By shedding light on the effectiveness of various retrieval strategies and introducing new evaluation paradigms, our work contributes to the ongoing refinement of question-answering systems tailored to enterprise needs.

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*.

Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2023. Evaluating correctness and faithfulness of instruction-following models for question answering. *arXiv preprint arXiv:2307.16877*.

AI@Meta. 2024. Llama 3 model card.

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

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, 44(1):1–50.

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

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024b. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

Paulo Finardi, Leonardo Avila, Rodrigo Castaldoni, Pedro Gengo, Celio Larcher, Marcos Piau, Pablo Costa, and Vinicius Caridá. 2024. The chronicles of rag: The retriever, the chunk and the generator. *arXiv preprint arXiv:2401.07883*.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*.

Mistral AI team. 2024. Mixtral8x22b.

NVIDIA AI. 2024. Rerank-QA-Mistral-4B.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Gemma Team. 2024. Gemma.

Shashank Verma, Vinh Nguyen, Nguyen Lee, Nave Algarici, Gabriel Moreira, Ronay AK, Caroline Gottlieb, Benedikt Schifferer, and Wei Ping. Build Enterprise Retrieval-Augmented Generation Apps with NVIDIA Retrieval QA Embedding Model.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2402.13178*.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. 2024. Crag–comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*.

A Bi-encoder versus Cross-encoder

Bi-encoders individually encodes each input (*e.g.*, a query or a document) into dense embeddings. Bi-encoders are scalable because the embeddings for inputs can be pre-computed and stored. This allows for efficient retrieval from large datasets using approximate nearest neighbor (ANN) search techniques. Cross-encoder jointly encodes a query and a document by concatenating them and feeding the concatenated sequence through a single model. It allows the model to directly capture the interactions between the inputs. Cross-encoder normally achieves higher retrieval accuracy. Nevertheless, cross-encoder is computationally expensive since it could not pre-compute embeddings like bi-encoder. This limitation forbids its application in large-scale retrieval scenarios. Therefore, cross-encoder is only applied in re-ranking tens of candidates retrieved based on embedding models.

B Annotation Process

The annotation was conducted using the LabelStudio platform by a team of 14 experienced annotators with professional backgrounds in business and finance. The process occurred in two phases:

- **Retrieval Annotation:** Annotators were presented with individual or grouped documents and tasked with formulating queries and selecting relevant context(s) that answered these queries.
- **Generator Evaluation:** A different set of annotators reviewed the query-context pairs along with multiple LLM-generated answers (using GPT-4, Claude 3.5, and Mixtral 7x22B). These LLM outputs served as guidelines for the annotators in creating the final ground truth answers.

Each annotation underwent rigorous vetting by a lead annotator and a data team lead to ensure adherence to guidelines and maintain quality.