Enhancing Retrieval in QA Systems with Derived Feature Association

Abhishek Goyal Keyush Shah Isaac Wasserman
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{abhi2358,keyush06,isaacrw}@seas.upenn.edu

Abstract

Retrieval augmented generation (RAG) has become the standard in long-context question answering (QA) systems. However, typical implementations of RAG rely on a rather naive retrieval mechanism, in which texts whose embeddings are most similar to that of the query are deemed most relevant. This has consequences in subjective QA tasks, where the most relevant text may not directly contain the answer. In this work, we propose a novel extension to RAG systems, which we call **R**etrieval from **AI D**erived **D**ocuments (RAIDD). RAIDD leverages the full power of the LLM in the retrieval process by deriving inferred features, such as summaries and example questions, from the documents at ingest. We demonstrate that this approach significantly improves the performance of RAG systems on long-context QA tasks.

1 Introduction

1.1 Preliminaries

First introduced by [8], retrieval augmented generation (RAG) allows LLMs to pull relevant information into context from a cache of documents. The system allows these models to access up-to-date information, rely less on their parameterized-memory, and leverage a large corpus of documents during generation, despite their limited context window [17]. RAG extends LLMs with a retrieval mechanism that takes a query, selects the most relevant texts from a given corpus, and hands them to the generator to inform its answer. Early approaches, were optimized end-to-end, using a jointly learned retriever and generator that communicated through a shared embedding space [8]. However, the requirement that such a system must be trained from scratch for each choice of generator architecture makes this approach expensive and cumbersome given the rapid pace with which new LLMs are developed. However, this paradigm was subverted by [14], which assumes the generator to be black-box, training a generator-agnostic retriever that simply prepends the retrieved text to the generator's input. In practice, the retriever is often further simplified to score documents based on their cosine similarity in a pretrained embedding space (dense retrieval); also popular is the use of BM25, a simple term-frequency based similarity metric (sparse retrieval) [17].

1.2 Motivation

RAG systems, especially those which rely on embedding cosine similarity or BM25 to measure relevance, are fast and remarkably effective for answering questions whose answers are explicitly stated in the text. However, from a user's perspective, this is only marginally more effective than a simple `ctrl+f` search. We expect more from the systems that we

Question	All of historians speak highly of Picardo's work, is this true? Why?
Target Text	"…was somewhat frowned upon in the 1960s and 1970s, and over half a century later is seen by archeologists and historians as a matter of significant controversy and regret …"
Ground Truth	False, because some people believe that Parrado destroyed the part of historical and architectural.
Retrieved Text	"…Picardo's published architectural drawings were highly regarded. They were described as "magnificent" by the leading Spanish restoration architect …"
Prediction ✗	Yes, because his architectural drawings were described as "magnificent" …

Figure 1: Example of a question from the LooGLE [9] dataset answered by a GPT-4 based RAG system. The system identifies the a text that describes how Picardo's work was regarded by one figure, but it fails to identify the more subtly worded target text which contains the answer.

call "artificially intelligent". In particular, we expect them to be able to answer questions whose answers are not explicitly stated in the text, but can be easily inferred from the text. Consider the example in Figure 1, using cosine similarity between the query and text; the retriever latches onto the text which most explicitly describes commentary on the artist's work and ignores the text which contains the answer but does not contain words like "regarded". This is a common failure mode for RAG systems, and it demonstrates how the retriever can be a hindrance to what is otherwise a powerful model for natural language understanding.

1.3 Related Work

RAG is an extremely active area of AI research with the goal of improving LLM performance and safety by augmenting the generation context with useful cached information [17]. The modern approach to RAG is generator-agnostic [14]. This is convenient as it allows for the use of the latest and greatest LLMs without the need for retraining; however, it places a much greater burden on the retriever to be effective and efficient.

In addition to the rather expensive and unportable practice of training domain-specific query encoders [14] and rerankers [3], simple modifications to the retrieval mechanism can make a profound impact on the system's performance. For example, [1] demonstrated the benefits of shorter chunk sizes and more granular indexing, while [11] and [13] construct multi-resolution document stores which afford a balance of context and precision. Input transformation has also been shown to be an effective tool for improving dense retrieval. In this paradigm, we use an LLM to generate a "pseudo-document" from the query: a piece of text that looks like the document we want to retrieve but it is actually contrived [2], serving as a template for the target document.

Data augmentation is an increasingly popular solution for improving dense retrieval. [6] uses an RAG system to retrieve audio clips using a text query. To achieve such behavior, they generate text aliases for each audio clip and index the clips according to these aliases. This approach serves as [1] demonstrates the utility of transforming RAG documents into more digestible, concise, and explicit forms. They propose "propositional retrieval", a method quickly adopted by the AI agent community [5] due to its effectiveness, portability, and runtime efficiency. Their method preprocesses documents to extract individual propositions from the text and indexes them instead of chunks of the original text. This greatly improves retrieval of information which is explicitly stated, but it sacrifices nuance that may be necessary for answering more complex questions.

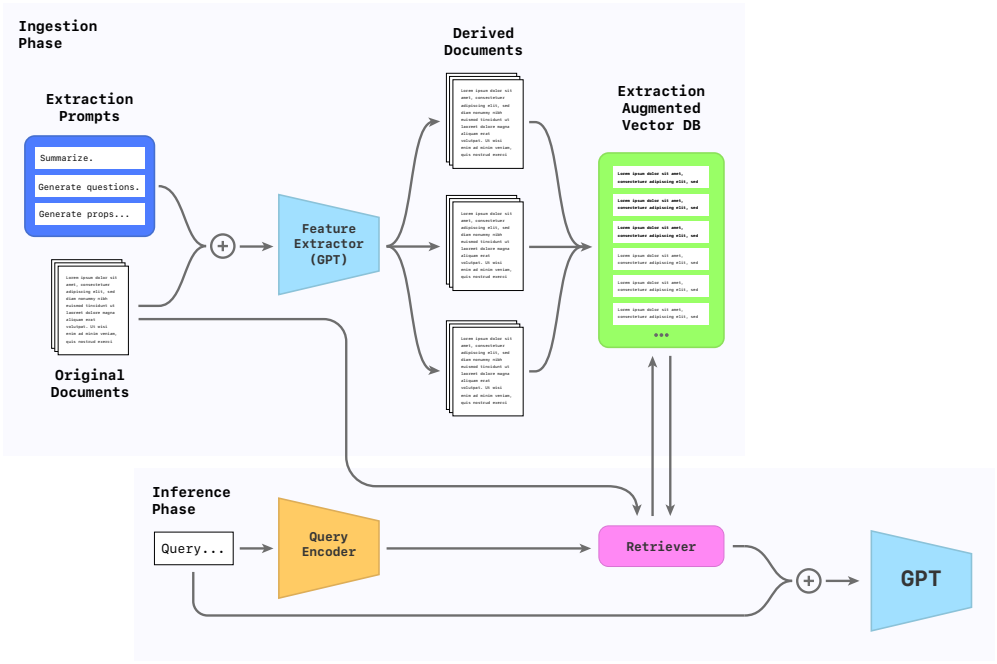

Figure 2: During the document ingest phase, RAIDD derives new documents from the input by prompting an GPT feature extractor to summarize and generate questions from the original documents. At inference, the retriever identifies the most relevant derived documents and places the corresponding source documents into context for question answering.

1.4 Contribution

TODO: This has to be explained in detail: how ours contribute to the previously stated results

2 Method

2.1 Derived Document Association

RAIDD generalizes the retrieval paradigms first introduced by Chen et al. (2023) [1]. In this original paradigm, LLM generated "propositional" documents completely supplant the original text, throughout the retrieval and generation processes. Our method, RAIDD, diverges from this paradigm by placing the original text in the generation context, rather than the derived documents themselves. We make this change in an effort to minimize the impact of information loss in the document derivation process, retaining the same level of detail as vanilla RAG. We call this practice of using the derived documents as handles for the original text "derived document association". Derived document association generalizes the retrieval technique of Huang et al. (2023), in which the system retrieves audio clips based on how well their derived text aliases match the query.

More concretely, our method involves two phases: ingest and inference. During ingest, we prompt an LLM to generate derived documents from each input document. These derived documents are either summaries of each chunk, sets of questions from each chunk, or both. We generate embeddings from these derived documents and store them in a vector database. At inference time, the retriever matches our query against all of the derived documents. For each of the top k derived documents, we place the corresponding original document into our question answering context.

2.2 RAIDD-S

Consider the following scenario where the document store has two documents: (a) "Johnny mixed 'ocean' and 'fire-engine' on his palette" and (b) "Johnny made so much green as a world-renowned painter", and the model is asked "What color paint did Johnny make?". Modern LLMs are more than capable of understanding that "green" is a euphemism for money, and that "ocean" and "fire-engine" are shades of blue and red that make purple when combined. However, dense retrieval with OpenAI's text-embedding-ada-002 [4] scores document (b) 4% higher than document (a), likely because of its inclusion of words like "made", "green", and "paint". While this is a contrived example, it demonstrates how dense retrieval systems place too much responsibility on a underpowered retriever, despite having access to incredibly performant natural language understanding models.

RAIDD-S seeks to improve the retrieval of implied information by retrieving text chunks based on their LLM generated summaries. By forcing our LLM to make the document more direct, concise, and explicit, we hope to improve the retriever's ability to identify the most relevant text. We generate summaries for each chunk of text in the document store, requesting that the LLM provide concept-level summaries which paraphrase the original text. We condition this generation with the original text as well as the summary of the previous chunk to maintain coherence.

2.3 RAIDD-Q

For RAIDD-Q, we draw inspiration from the work of [2] and [16]. While in their work, they generate pseudo-documents from the query in order to provide the retriever with a template for the target document, we perform the inverse process, generating pseudo-queries from the documents. We prompt the LLM to generate a set of unique reading comprehension questions from each chunk of text in the document store. To accomodate multiple chunk sizes, we generate 32 questions for every 1024 tokens of text. Since each chunk now has multiple questions associated with it, during retrieval, we retrieve the top k questions that correspond to unique chunks of the original text. We then place the original text into context for question answering.

2.4 RAIDD-U

2.5 Dataset and Evaluation

We evaluate long-context question answering performance using the long-dependency QA subset of LooGLE [9]. To minimize cost, we use the first 100 questions of the dataset, which utilize a total of 14 input contexts. Question answering performance is measured using ROUGE [10] and GPT-4 [12] prompted to decide whether the generated answer was sufficiently similar to the ground-truth given the question.

Note that generating summarizations and questions used as the AI derived documents are just the means to an end that is answering the questions.

2.6 Implementation Details

Our experiments are implemented using the LlamaIndex [11] RAG library. For generating derived documents, we prompt an instruction tuned Mixtral-8x7B model [7]. We use the frontier-class Mistral Large model [15] as our question answering model. Our retriever uses a simple cosine similarity metric between the query and document embeddings, which are encoded using OpenAI's text-embedding-ada-002 [4]. The RAIDD process comes in three distinct flavors, whose implmentations are each described below.

3 Experiments

3.1 Improvements over Baseline

3.2 Ablation Studies

Table 1: Performance comparison of various flavors of RAIDD. RAIDD-S uses summary generation, while RAIDD-Q uses question generation. RAIDD-U matches queries against an index of summaries, questions, and the raw text combined.

Method	Chunk Size	Overlap	Top-k	GPT-4 Score	ROUGE-1	ROUGE-L
RAG	64	10	32	0.43	0.216	0.174
	128	25	16	0.46	0.209	0.167
	256	50	8	0.48	0.249	0.206
	512	100	4	0.48	0.223	0.189
	1024	200	2	0.39	0.212	0.167
	2048	400	1	0.35	0.191	0.152
RAIDD-S	64	10	32	0.48	0.214	0.171
	128	25	16	0.41	0.204	0.167
	256	50	8	<u>0.49</u>	0.230	0.183
	512	100	4	0.43	0.224	0.178
	1024	200	2	0.31	0.205	0.178
	2048	400	1	0.31	0.201	0.163
RAIDD-Q	64	10	32	0.46	0.217	0.173
	128	25	16	0.46	0.222	0.180
	256	50	8	0.44	**0.269**	**0.208**
	512	100	4	0.46	0.218	0.180
	1024	200	2	0.44	0.201	0.167
	2048	400	1	0.39	0.203	0.164
RAIDD-U	64	10	32	0.47	0.220	0.181
	128	25	16	0.46	0.214	0.172
	256	50	8	**0.52**	<u>0.254</u>	<u>0.207</u>
	512	100	4	0.45	0.225	0.182
	1024	200	2	0.46	0.209	0.171
	2048	400	1	0.36	0.203	0.163

Our best model will likely use a combination of derived documents. For each feature used, remove it from the base model and evaluate performance difference.

3.3 Analysis

Here, we will perform analyses of success and failure modes. We will also look at how RAIDD improves the retrieval of target passages and changes the rank of individual documents. We will also look at a few example questions and how our method compares to the baseline.

4 Conclusion

References

[1] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense X Retrieval: What Retrieval Granularity Should We Use? *arXiv*, 2023.

[2] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval without Relevance Labels.

[3] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. Re2G: Retrieve, Rerank, Generate.

[4] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. New and improved embedding model, Dec 2022.

[5] Chase Harrison. Langchain – propositional-retrieval, 2024.

[6] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models. *arXiv*, 2023.

[7] 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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts. *arXiv*, 2024.

[8] 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. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv*, 2020.

[9] Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*, 2023.

[10] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[11] Jerry Liu. LlamaIndex, 11 2022.

[12] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power,

Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[13] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval.

[14] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-Augmented Black-Box Language Models. *arXiv*, 2023.

[15] Mistral AI Team. Au large | mistral ai | frontier ai in your hands, Feb 2024.

[16] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query Expansion with Large Language Models. *arXiv*, 2023.

[17] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey, 2024.