

QuOTE: Question-Oriented Text Embeddings

Andrew K. Neeser

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Naren Ramakrishnan, Chair

Chris Latimer

Chang-Tien Lu

May 5, 2025

Blacksburg, Virginia

Keywords: Retrieval-Augmented Generation, Synthetic Question Generation, Document
Representation, Information Retrieval

Copyright 2025, Andrew K. Neeser

QuOTE: Question-Oriented Text Embeddings

Andrew K. Neeser

(ABSTRACT)

We present QuOTE (Question-Oriented Text Embeddings), a novel enhancement to retrieval-augmented generation (RAG) systems, aimed at improving document representation for accurate and nuanced retrieval. Unlike traditional RAG pipelines, which rely on embedding raw text chunks, QuOTE augments chunks with hypothetical questions that the chunk can potentially answer, enriching the representation space. This better aligns document embeddings with user query semantics, and helps address issues such as ambiguity and context-dependent relevance. Through extensive experiments across diverse benchmarks, we demonstrate that QuOTE significantly enhances retrieval accuracy, including in multi-hop question-answering tasks. Our findings highlight the versatility of question generation as a fundamental indexing strategy, opening new avenues for integrating question generation into retrieval-based AI pipelines.

QuOTE: Question-Oriented Text Embeddings

Andrew K. Neeser

(GENERAL AUDIENCE ABSTRACT)

Modern artificial intelligence tools often help users by searching through large collections of documents and then using those search results to generate answers. This process can sometimes misinterpret a question or miss important connections in the text. In our work, we introduce QuOTE, a simple yet powerful method that teaches the system to think in terms of questions: each piece of text is paired with relevant, hypothetical questions it could answer. By organizing information around questions and answers, QuOTE creates clearer, more meaningful representations of documents. In tests that include cases where answers require combining information from different parts of a document, QuOTE consistently retrieves more accurate and relevant information than traditional approaches. This question-based indexing approach makes search-and-answer systems more reliable and could enhance a wide range of everyday tools, from virtual assistants to online help desks.

Contents

List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Introduction	1
2 Review of Literature	4
2.1 Related Work	4
2.1.1 Dense vs Sparse Retrievers	4
2.1.2 Retrievers vs Rerankers	5
2.1.3 Exact search vs Approximate Nearest Neighbors (ANN)	5
2.1.4 Distractions vs Noise in RAG	5
2.1.5 Real vs Hypothetical Embeddings	6
2.1.6 Supporting Asymmetric QA Tasks	6
2.1.7 Neural Information Retrieval	7
2.1.8 End-to-End RAG Systems	7
3 Methodology	8

3.1	QuOTE	8
3.1.1	Question Generation at Pre-Query Time	9
3.1.2	Embedding	10
3.1.3	Retrieval and Deduplication at Query Time	11
3.2	Datasets and Metrics	11
3.2.1	Natural Questions (NQ)	11
3.2.2	SQuAD	12
3.2.3	MultiHop-RAG	13
3.2.4	Government Report Dataset	13
3.2.5	Ensuring No Data Leakage	15
3.2.6	Algorithms Compared	15
3.2.7	Evaluation Metrics	16
4	Evaluation	19
4.1	Evaluation	19
4.1.1	Performance Using Standard RAG Metrics	20
4.1.2	Effect of Different Prompts to Generate Questions	21
4.1.3	QuOTE Performance vis-a-vis Embedding Model	23
4.1.4	Effect of Number of Questions	23
4.1.5	Comparison with HyDE	27

4.1.6	Comparison with Doc2Query-QuOTE	28
4.1.7	Evaluation of Contextual Retrieval	31
4.1.8	Effect of Deduplication	33
4.1.9	What Happens If We Combine All Questions Into One Chunk?	34
4.1.10	Effect of Question–Question Similarity Pruning	36
4.1.11	Can we use a Cheaper LLM for Question Generation?	37
4.1.12	Effect of the Number of Contexts on Retrieval Accuracy	39
5	Conclusion	42
5.1	Discussion	42
5.2	Future Work	43
	Bibliography	45

List of Figures

3.1	Overview of QuOTE. Documents are split into chunks and processed by a question generator (LLM) to create relevant questions. Chunks along with the questions they purport to answer are embedded in a vector database. At query time, a retriever and deduplicator processes user queries to generate final responses.	10
4.1	Cosine similarity distribution between top-5 generated questions and evaluation queries for QuOTE and Doc2Query on the GovReport dataset. The QuOTE distribution (blue) exhibits higher mean and median similarity values than Doc2Query (orange), indicating that QuOTE produces questions more semantically aligned with the target queries.	30
4.2	Cosine similarity distribution between top-5 generated questions and evaluation queries for QuOTE and Doc2Query on the SQuAD dataset. Similar to the GovReport results, the QuOTE distribution shows higher semantic alignment with target queries than Doc2Query.	31
4.3	Distribution of contexts per title in SQuAD (N=442 titles). The mean of 42.74 contexts per title and maximum of 149 contexts demonstrate the dataset’s high context density.	39
4.4	Distribution of contexts per title in Natural Questions (N=48,525 titles). The highly concentrated distribution around a median of 1 context per title indicates predominantly singular contexts.	40

4.5	Percentage increase in Top-1 retrieval accuracy with QuOTE compared to naive retrieval across the number of contexts. SQuAD shows steady improvement that grows with the number of contexts, reaching 20.7% improvement at size 100, while NQ shows consistent but variable gains up to 18.3%. . . .	41
-----	--	----

List of Tables

4.1	Comparison of QuOTE and Naive RAG using standard retrieval metrics on SQuAD, Natural Questions (NQ), and GovReport datasets.	20
4.2	Prompt templates for Natural Questions (NQ), SQuAD, and MultiHop-RAG.	22
4.3	Key retrieval metrics across Natural Questions (NQ), SQuAD, and MultiHop-RAG for Naive , Basic , and Complex prompting strategies. Bolded values denote the best performance for each metric.	23
4.4	Performance of Naive vs. QuOTE modes on SQuAD, NQ, and MultiHop-RAG across five embedding models. Per-row bolded entries denote the better value for that metric.	24
4.5	Performance comparison across different numbers of generated questions on SQuAD, NQ, and MultiHop-RAG datasets. Results show Context and Title Accuracy at different k values for SQuAD and NQ, and Full/Partial Match for MultiHop-RAG. Bolded entries denote the best performance per metric. .	25
4.6	Comparison of Naive , HyDE , and QuOTE across three QA tasks. The fastest (lowest time) and most accurate (highest accuracy) entries in each column are bolded . In SQuAD , Naive is fastest while QuOTE achieves the highest accuracy; for NQ , Naive runs fastest while HyDE slightly outperforms the others in accuracy; and in MultiHop-RAG , Naive remains fastest, whereas QuOTE attains the highest full-match rates.	28

4.7	Comparison of Naive, Doc2Query-QuOTE, and QuOTE across standard retrieval metrics on SQuAD, Natural Questions (NQ), and GovReport.	29
4.8	Performance comparison of Contextual Retrieval against Naive and QuOTE pipelines on a 1000-document subset of SQuAD and GovReport datasets. . .	33
4.9	Comparison of retrieval approaches. Index=time to build database (seconds), Query=time to process all queries (seconds), ms/q=milliseconds per query, C@1=Context accuracy, T@1=Title accuracy. Bold indicates best per column.	34
4.10	Total number of documents in the index under each method.	35
4.11	Comparison of retrieval performance across Naive, QuOTE-OneDoc, and QuOTE variants. Bold indicates best performance per column.	35
4.12	SQuAD: Retrieval performance under varying Q-Q similarity thresholds. . .	36
4.13	Natural Questions: Retrieval performance under varying Q-Q similarity thresholds.	37
4.14	GovReport: Retrieval performance under varying Q-Q similarity thresholds.	37
4.15	Comparison of different models on a SQuAD subset. We report Context Accuracy (C@k) and Title Accuracy (T@k) at k=1 and k=5. Best value(s) in each column are bolded	38

List of Abbreviations

AI Artificial Intelligence

ANN Approximate Nearest Neighbor

GPU Graphics Processing Unit

IR Information Retrieval

LLM Large Language Model

MRR Mean Reciprocal Rank

NDCG Normalized Discounted Cumulative Gain

NLP Natural Language Processing

QA Question Answering

QuOTE Question-Oriented Text Embeddings

RAG Retrieval-Augmented Generation

TF-IDF Term Frequency–Inverse Document Frequency

AI is a branch of computer science focused on creating systems capable of tasks that typically require human intelligence.

NLP is the field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.

IR is the discipline that studies methods for obtaining relevant information from large collections of data.

LLM refers to large-scale neural network models trained on extensive text corpora to perform diverse language tasks.

RAG is an approach that enhances text generation by retrieving relevant documents and conditioning the model's output on them.

QA involves systems designed to automatically answer questions posed in natural language.

MAP@K measures the average precision of the top- K retrieved results across a set of queries.

QuOTE augments document chunks with hypothetical questions to improve embedding alignment with user queries.

TF-IDF is a numerical statistic reflecting how important a word is to a document in a corpus.

ANN refers to Approximate Nearest Neighbor algorithms for efficiently finding similar items in high-dimensional spaces.

GPU is specialized hardware optimized for parallel computations, commonly used to accelerate machine learning workloads.

MRR is an evaluation metric averaging the reciprocal rank of the first relevant result across queries.

NDCG is a ranking quality metric that accounts for the graded relevance of items and their positions in the result list.

Chapter 1

Introduction

1.1 Introduction

Retrieval-augmented generation (RAG [32, 33, 36]) serves as a significant contribution to the deployment and acceptance of LLMs in practice. Given a user’s prompt, RAG retrieves relevant information from a document collection, augments (prefixes) it to the user’s prompt, thus helping ensure that any generated content can be accurate, pertinent, and grounded in up-to-date information. In a typical RAG implementation, at pre-query time, the corpus is broken down into chunks, which are stored as vector embeddings. At query time, these chunks are searched and used to augment the user’s prompt. Several variants of RAG have been proposed over the years [2, 4, 12] to address specific use cases and challenges.

RAG has helped reinforce the criticality of information retrieval (IR) as a vital component of modern NLP and AI pipelines. Despite this resurgence, much of the focus has been on enhancing the G (generation) component, often leaving advancements in the R (retrieval) aspect comparatively underexplored. Recently, some notable efforts have emerged to address this imbalance.

For example, Anthropic introduced contextual retrieval [1] where each chunk is augmented with additional context before embedding; this approach is claimed to reduce incorrect chunk retrieval rates by up to 67%. Similarly, recent works have explored prompt caching [7], a

strategy to reuse previously retrieved or generated results to optimize latency and computation costs in iterative or repetitive query scenarios.

We present an LLM approach to introduce **synthetic questions** into the indexing process. While there has been prior work in using synthetic questions to enhance IR we believe our specific architecture is unique and explores more completely the space of possibilities at this intersection. Furthermore, synthetic data has emerged as a powerful tool across the LLM ecosystem—supporting fine-tuning, improving robustness, and enabling data augmentation in domains where labeled examples are scarce. However, IR has only just begun to explore the potential of synthetic data in a systematic and principled way. Our work aims to close this gap, demonstrating that when synthetic questions are generated and integrated with care, they can significantly enhance document representations and sharpen retrieval focus.

One of our key insights is that documents can often be more effectively represented by the questions they can answer, rather than by their direct content. To this end, for each chunk, we propose generating a set of questions that the chunk is likely to answer, embedding these alongside the original chunk content. We refer to such embeddings as Question-Oriented Text Embeddings (QuOTE). Although the idea of synthetic questions has been explored in IR before, we demonstrate key differences in our methodology that lead to significant gains in performance.

This thesis makes the following contributions.

1. We demonstrate that the idea of embedding (hypothetical) questions along with text chunks significantly enhances retrieval performance, particularly in scenarios where nuanced understanding of the content is required. This idea holds promise beyond RAG by opening up the possibility of question generation as a fundamental indexing strategy.

2. We prioritize retrieval performance rather than generation quality in our evaluation, and conduct an exhaustive empirical analysis of QuOTE with multiple language models, several key datasets, a range of query workloads, and compare it versus other comparable systems. This gives insight into specific regions of the configuration space where QuOTE performs best and future directions of research.
3. Beyond empirical results, we characterize the features of RAG settings where (and why) QuOTE works, and how we can anticipate performance improvements prior to embarking on QuOTE-style indexing for given corpora.

Chapter 2

Review of Literature

2.1 Related Work

Many studies have highlighted the impact of key design choices for the success of a RAG implementation [23, 25, 28].

2.1.1 Dense vs Sparse Retrievers

The debate between dense and sparse retrievers continues into RAG research [3, 27]. Dense retrievers, such as those based on vector embeddings, excel at capturing semantic similarity, making them particularly effective for nuanced queries. However, sparse retrievers like BM25 and TF-IDF continue to dominate in scenarios where explicit token matches, such as named entities, acronyms, or abbreviations, are critical to relevance. This distinction has led to hybrid approaches in many RAG systems, which combine dense and sparse retrievers. For example, a typical implementation involves first running a keyword-based sparse retrieval to gather an initial pool of relevant chunks, followed by a dense retrieval to refine the results.

2.1.2 Retrievers vs Rerankers

Many RAG systems employ a two-step pipeline: a fast retriever selects the top-k candidate chunks, and a reranker, typically a computationally intensive cross-encoder, reorders these candidates for final use. While rerankers generally improve the quality of retrieved results, recent research [11] cautions against extending reranking to larger candidate sets. Beyond a certain threshold, performance tends to plateau and may even degrade, likely due to noise introduced in larger retrieval pools. These findings underscore the importance of balancing efficiency and effectiveness in the retrieval-reranking pipeline.

2.1.3 Exact search vs Approximate Nearest Neighbors (ANN)

Approximate nearest neighbor (ANN) techniques [10] have become the de facto standard for scalable dense retrieval due to their ability to handle large corpora efficiently. However, exact search methods, while computationally more demanding, offer greater precision in certain use cases, such as high-stakes QA tasks. Several studies [19, 34] compare these approaches, highlighting trade-offs in latency, accuracy, and robustness to query variations. For instance, ANN methods may struggle with long-tail queries or datasets containing subtle semantic distinctions.

2.1.4 Distractions vs Noise in RAG

Cuconasu et al. [5] study the performance of RAG for QA tasks in the presence of so-called *distracting* and *noise* documents. Distracting documents are those with high retrieval scores, but that do not contain the answer; noise documents are picked at random from the corpus. The interesting finding from this study was that while distracting documents lead

to performance deterioration as expected, noise documents lead to improved performance, presumably due to better reliance on pretrained reasoning. However, these findings are somewhat questioned by recent work [16], which suggests that noise documents can degrade system reliability in certain settings, calling for further investigation.

2.1.5 Real vs Hypothetical Embeddings

Contextual retrieval techniques, such as Anthropic’s approach to augmenting chunks with additional information before embedding, have emerged as promising ways to reduce retrieval errors. Similarly, Hypothetical Document Embeddings (HyDE) [6] involve generating synthetic text based on the query and embedding it alongside real documents. These methods aim to capture query-specific nuances, resulting in more robust retrieval in open-domain and QA contexts. Our work builds on these approaches by leveraging question-based chunk representations for improved relevance.

2.1.6 Supporting Asymmetric QA Tasks

In many QA scenarios, particularly in customer support and enterprise search, there exists a fundamental asymmetry: user queries are often brief, while answers require detailed, structured information. RAG systems addressing this imbalance have incorporated techniques such as hierarchical retrieval [18], multi-hop reasoning [20], and weighted retrieval pipelines [13] to bridge this gap. Recent efforts in this domain include query-expansion strategies [31] and retrieval conditioning [35] to better align user intent with document granularity.

2.1.7 Neural Information Retrieval

Neural information retrieval methods aim to model complex semantic relationships and contextual relevance more effectively than traditional approaches. While approaches like ColBERT [14] and DPR [24] have made significant strides in dense retrieval, they continue to struggle with nuanced information seeking behaviors, involving hierarchical relationships, managing distributed information across multiple documents, and dealing with context-dependent relevance ranking.

2.1.8 End-to-End RAG Systems

Fully integrated, end-to-end RAG systems (e.g., from companies like Vectorize.io) are becoming increasingly popular for tasks requiring seamless interaction between retrieval and generation. Recent work [26] has focused on optimizing these systems for efficiency, scalability, and robustness. End-to-end designs often integrate prompt caching, hybrid retrieval, and adaptive reranking to achieve state-of-the-art performance across diverse NLP tasks.

Chapter 3

Methodology

3.1 QuOTE

QuOTE can be viewed in the lineage of query reformulation [29] and multi-hop reasoning [17], but takes a unique perspective by focusing on question generation as a fundamental indexing strategy. As discussed earlier, the naive RAG approach can fail to capture the *intent* behind user queries, especially when queries are succinct (e.g., entity lookups) or require extracting specific details from a chunk. In QuOTE we transform each chunk of text into *multiple* (question + chunk) representations capturing a range of opportunities for retrieval. Note that each generated question (plus chunk) is then stored as a separate “document” or embedding in the vector database.

See Algorithm 3.1 for pseudocode to illustrate how QuOTE builds an index, and Algorithm 3.2 for how it is queried. We next describe key stages of the pipeline (see Fig. 3.1):

See Fig. 3.1 for how QuOTE works.

Algorithm 3.1 (ht). **Require:** Corpus C , LLM, VectorDB, NumQuestions

- 1: **function** BUILDQUOTEINDEX(C , LLM, VectorDB, NumQuestions)
- 2: $\mathcal{P} \leftarrow \text{SplitCorpus}(C)$ ▷ Split into chunks/passages
- 3: **for all** $p \in \mathcal{P}$ **do**
- 4: $Q \leftarrow \text{LLMGenerateQuestions}(p, \text{NumQuestions})$

```

5:     for all  $q \in Q$  do
6:          $doc \leftarrow q \parallel p$  ▷ concatenate or store separately
7:         VectorDB.Add(doc, metadata={originalChunk: p})
8:     end for
9: end for
10: end function

```

Algorithm 3.2 (ht). **Require:** UserQuery u , VectorDB, k , M

```

1: function QUOTEQUERY( $u$ , VectorDB,  $k$ ,  $M$ )
2:      $u_{emb} \leftarrow \text{EMBED}(u)$ 
3:      $\mathcal{R} \leftarrow \text{VECTORDB.QUERY}(u_{emb}, top = k \times M)$ 
4:     uniqueResults  $\leftarrow \text{DEDUPLICATE}(\mathcal{R})$ 
5:     finalContexts  $\leftarrow \text{TOPK}(uniqueResults, k)$ 
6:     answer  $\leftarrow \text{LLM}(UserQuery \parallel finalContexts)$ 
7:     return answer
8: end function

```

3.1.1 Question Generation at Pre-Query Time

We split the corpus into smaller passages (or chunks). For each chunk, we prompt an LLM to generate a set of questions that the chunk can answer. While question generation is a well studied topic in NLP [8, 9, 37]. The quality and diversity of generated questions play a significant role in QuOTE’s effectiveness. We use an LLM with prompt engineering (see Section 4.1.2) to create a representative set of questions with specificity and coverage. By creating multiple question-based embeddings for each chunk, QuOTE better captures diverse user queries that reference the same text in different ways. If a user’s query is similar

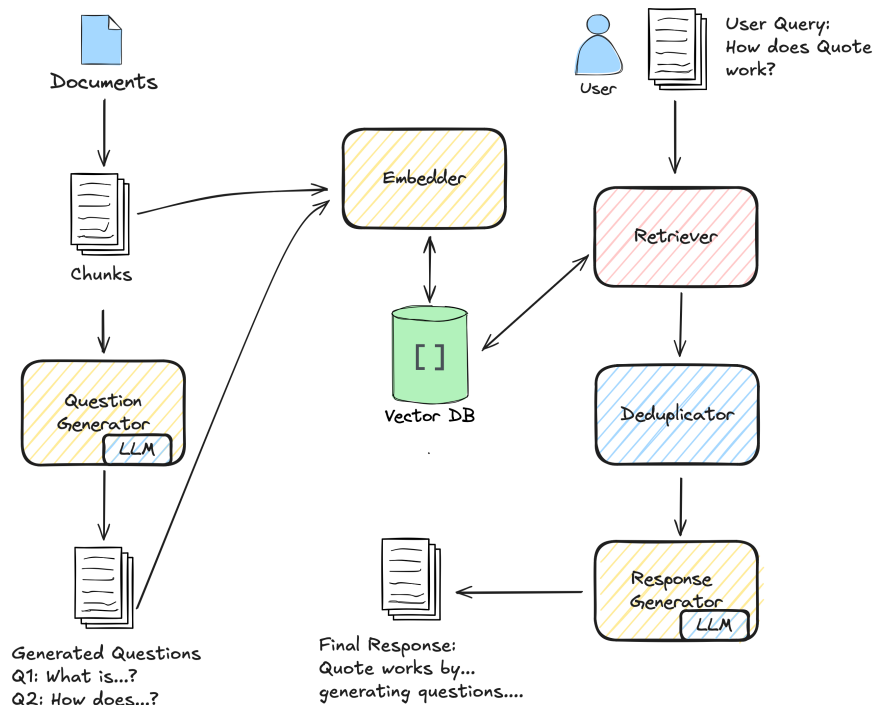


Figure 3.1: Overview of QuOTE. Documents are split into chunks and processed by a question generator (LLM) to create relevant questions. Chunks along with the questions they purport to answer are embedded in a vector database. At query time, a retriever and deduplicator processes user queries to generate final responses.

(semantically) to one of the chunk-generated questions, that chunk becomes more likely to rank highly, leading to more accurate retrieval.

3.1.2 Embedding

Instead of storing just the original chunk embedding, *we store each generated question* (along with the original chunk) in the vector database. In Section 4.1.3 we demonstrate that the performance of QuOTE is agnostic to the choice of embedding model.

3.1.3 Retrieval and Deduplication at Query Time

During query time, multiple retrieved “documents” often reference the same underlying chunk. Hence, a deduplication step is necessary to ensure we select the top- k distinct chunks, avoiding wasted slots. To this purpose we ‘over-retrieve’ top- $k \times M$ results (for some value of M) from the question-based embeddings. (Note that this de-duplication step is unique to the QuOTE pipeline and is not a feature in classical RAG pipelines.)

3.2 Datasets and Metrics

While there exist a variety of datasets for RAG evaluation (see Table ??) not all are geared toward evaluating retrieval performance as distinct from generation, which is our focus here. For instance, QA datasets where we are evaluated against the quality of the generated answer, or where the original ground truth chunks are not available, do not support assessing the performance of QuOTE in helping improve retrieval of relevant chunks. Accordingly, we focus on three benchmark datasets commonly used for question answering: *Natural Questions* (NQ) [15], *SQuAD* [21, 22], and *MultiHop-RAG* [30]. These datasets vary in complexity, domain coverage, and the style of questions, providing a broad platform to test the retrieval capabilities of our approach.

3.2.1 Natural Questions (NQ)

The Natural Questions (NQ) dataset [15] is a large-scale benchmark, with questions directly sourced from real user queries and answers keyed to Wikipedia articles. The dataset is split into approximately 307k training examples and roughly 7.8k each in the development and test sets. For each query, the dataset provides the relevant passages (long answer) and the

precise phrases or entities (short answer) where the answer resides.

One non-trivial issue pertains to *multiple, highly similar passages in the same article*. For example, consider passages about the song “Heroes” by David Bowie from the Wikipedia article titled “Heroes (David Bowie song)”. This article has two passages that are nearly identical, differing only in minor wording (e.g., “in the UK” vs. “in the United Kingdom”). These slight variations do not change the factual content but result in multiple, nearly duplicate contexts.

Such minor differences unnecessarily fragment the dataset into multiple contexts, each labeled as distinct. This discrepancy complicates retrieval-based evaluations because systems are penalized if they return an almost-correct chunk that differs by only a few words from the one labeled as ground-truth.

To address this issue, we merge highly similar chunks based on a text-similarity threshold, combining their respective questions into a single context group. This merging strategy reduces noise and ensures that semantically equivalent passages (or chunks) are treated as one, allowing retrieval mechanisms to focus on true distinctions in content rather than trivial rephrasings.

3.2.2 SQuAD

The Stanford Question Answering Dataset (SQuAD) [21] is widely recognized as a benchmark for reading comprehension and extractive QA. Each question is associated with an exact answer span in the corresponding article, ideal for our extractive evaluation purposes.

3.2.3 MultiHop-RAG

MultiHop-RAG [30] is specifically designed to test multi-hop question answering. Unlike SQuAD and NQ, which pair each question with a single relevant paragraph or article, MultiHop-RAG associates multiple ground-truth documents with each query. For instance, a query such as: “*Which company is being scrutinized by multiple news outlets for anticompetitive practices and is also suspected of foul play by individuals in other reports?*” will require cross-referencing two or more articles to gather the necessary evidence.

3.2.4 Government Report Dataset

Dataset Overview. We introduce a novel evaluation corpus derived from the Hugging Face repository `launch/gov_report`, consisting of technical reports and corresponding human-authored summaries issued by U.S. government research agencies, including the Congressional Research Service and the U.S. Government Accountability Office. These reports are notably longer and delve deeply into specialized policy, budgetary, and regulatory content compared to conventional QA benchmarks, making them particularly suitable for testing long-range retrieval capabilities.

Chunking and Preprocessing. To prepare this dataset for the QuOTE indexing pipeline, we extracted a randomly selected subset comprising 10,000 textual passages (“chunks”). The preprocessing steps included:

- **Sentence Segmentation:** Reports were segmented into sentences using NLTK’s `sent_tokenize`, preserving natural linguistic boundaries.
- **Token-Length Control:** We employed a RoBERTa-base tokenizer for GPU-accelerated

batch token counting (maximum length of 512 tokens per batch). Sentences were then greedily combined into chunks ranging from 256 to 512 tokens, ensuring sentence integrity was maintained.

Synthetic Query Generation. We augmented each chunk with synthetic retrieval-oriented queries using Google’s `gemini-2.0-flash-lite-001` model via the OpenRouter API. Query generation involved:

- **Prompt Design:** A specifically designed system prompt guided the model to generate multiple precise and pronoun-free queries formatted explicitly as "Question: ...". These queries targeted specific factual details, entities, dates, or policy outcomes explicitly mentioned within the chunk while explicitly avoiding generic references such as "according to the passage." representation.

Transformation to QA Format. The generated data was then transformed into a compatible format, facilitating direct integration with our existing evaluation framework. The transformation included:

- Using each chunk’s original document identifier as a "title" under which contexts were organized.
- Identical contexts appearing under the same title were merged, unifying their sets of generated questions.
- This process resulted in a structured JSON mapping of each document title to associated context-question pairs, prepared explicitly for retrieval evaluations.

The introduction of this Government Report dataset enables QuOTE to be rigorously evaluated on extensive, domain-specific documents, providing insights into the scalability and

efficacy of question-oriented indexing strategies in realistic policy and regulatory document scenarios.

3.2.5 Ensuring No Data Leakage

To rigorously maintain the integrity of our evaluations and prevent any potential data leakage, we implemented a stringent verification step. Specifically, we ensured that none of the synthetic queries generated during the QuOTE indexing pipeline matched any queries from the evaluation sets. This critical step guarantees that the evaluation process accurately reflects the retrieval performance without contamination, allowing us to confidently attribute observed performance improvements solely to the effectiveness of our QuOTE approach.

3.2.6 Algorithms Compared

To contextualize our evaluation results, we compare the following retrieval pipelines:

- **NaiveRAG.** Standard retrieval-augmented generation using ChromaDB without any query or index-time augmentation. Chunks are embedded and indexed directly, and at query time the user’s prompt is embedded and matched against these raw chunk embeddings.
- **HyDE.** Hypothetical Document Embeddings [6], a query-time transformation approach that first uses an LLM to generate a “pseudo-document” from the user’s query, embeds that document, and retrieves chunks based on this enriched representation.
- **Contextual Retrieval.** Anthropic’s contextual embedding strategy [1], which prepends a short descriptive context—derived by prompting an LLM with both the full document

and each chunk—to the chunk before embedding. This situates each chunk within its originating document at index time.

- **Doc2Query-QuOTE.** A lightweight variant of QuOTE where, instead of using a full LLM, we employ a trained `doc2query` transformer model to generate questions for each chunk. These questions are embedded alongside the chunk and deduplicated at query time exactly as in the standard QuOTE pipeline.
- **QuOTE-OneDoc.** An index-time variant of QuOTE in which all questions generated for a chunk are concatenated into a single “super-chunk” document. This removes the need for a deduplication step at query time but still leverages question-based index augmentation.

3.2.7 Evaluation Metrics

To evaluate retrieval performance across both single-hop and multi-hop settings, we employ a combination of custom task-specific metrics and standard information retrieval (IR) metrics. These metrics are chosen to reflect both the precise retrieval demands of real-world QA applications and the broader effectiveness of the QuOTE framework.

Single-hop QA (SQuAD, NQ, Gov Report). In these datasets, each query typically maps to a single relevant Wikipedia article and a specific paragraph containing the answer span. Our primary evaluation focuses on whether QuOTE can isolate both the correct document and the exact supporting passage:

- **Context Accuracy (C@k):** The fraction of queries for which the correct *paragraph-level* context appears within the top- k retrieved results. A retrieval is considered successful if the paragraph containing the short answer is retrieved within the top- k .

- **Title Accuracy (T@k):** The fraction of queries for which the correct *article title* is present in the top- k results. This is a coarser but useful indicator of whether the system locates the right document, even if it misses the exact span.

Multi-hop QA (MultiHop-RAG). These queries are more complex and may require integrating multiple evidence pieces across different documents. We use two levels of accuracy to capture this added difficulty:

- **Full Match Accuracy (Full@k):** All necessary evidence documents must appear within the top- k . Missing even one piece of required evidence results in a failed retrieval under this strict criterion.
- **Partial Match Accuracy (Part@k):** Measures the proportion of required evidence that is successfully retrieved in the top- k . This reflects how incomplete retrieval still contributes to partial reasoning and highlights how retrieval errors degrade overall QA performance.

Standard IR Metrics. To enable direct comparison with prior RAG systems, and as reported in Section 4.1.1, we also include standard retrieval metrics commonly used in the IR literature:

- **Normalized Discounted Cumulative Gain (NDCG@k):** Measures ranking quality by rewarding systems that place relevant documents higher in the retrieval list.
- **Mean Average Precision (MAP@k):** Computes the mean of average precision scores across queries, particularly useful when queries may have multiple relevant documents.

- **Mean Reciprocal Rank (MRR@k)**: Captures the inverse rank of the first relevant document, making it particularly sensitive to how quickly a relevant result is retrieved.

While NDCG, MAP, and MRR provide useful insight—especially for comparing embedding models—we primarily rely on **Context Accuracy (C@k)** and **Title Accuracy (T@k)** throughout this work. These metrics offer clear and interpretable signals about retrieval effectiveness at both the paragraph and article level, aligning closely with real-world applications where finding the right information with high precision is paramount.

Chapter 4

Evaluation

4.1 Evaluation

We conduct a comprehensive evaluation to answer the below questions:

1. (Section [4.1.1](#)) How does QuOTE perform using standard RAG retrieval metrics such as NDCG, MAP, and MRR compared to Naive retrieval?
2. (Section [4.1.2](#)) Is QuOTE able to automatically generate questions that improve the performance of retrieval-augmented generation?
3. (Section [4.1.3](#)) How sensitive is QuOTE performance to the choice of embedding model?
4. (Section [4.1.4](#)) How many questions must be generated for QuOTE to be effective?
5. (Section [4.1.5](#)) How does QuOTE compare to HyDE, the state-of-the-art approach to query enrichment?
6. (Section [4.1.6](#)) Can we replace LLM-based question generation with lighter-weight alternatives like doc2query without sacrificing retrieval performance?
7. (Section [4.1.7](#)) How does Contextual Retrieval compare to QuOTE and Naive RAG in terms of retrieval effectiveness and computational cost?
8. (Section [4.1.8](#)) How negligible or significant is QuOTE's deduplication overhead?

9. (Section 4.1.9) How does QuOTE perform when all questions for a chunk are combined into one document instead of stored separately?
10. (Section 4.1.10) How does question–question similarity pruning affect retrieval performance?
11. (Section 4.1.11) Because QuOTE uses an LLM for question generation as well as for answer generation, can we employ a cheaper model for question generation and does this significantly affect performance?
12. (Section 4.1.12) Can we characterize the properties of contexts for which QuOTE has selective superiority?

4.1.1 Performance Using Standard RAG Metrics

Table 4.1 compares QuOTE against the Naive RAG baseline using standard retrieval metrics across multiple datasets, alongside Context and Title Accuracy for reference.

Table 4.1: Comparison of QuOTE and Naive RAG using standard retrieval metrics on SQuAD, Natural Questions (NQ), and GovReport datasets.

Metric	SQuAD	NQ	GovReport
Naive vs QuOTE			
C@1	59.02 / 64.68	27.41 / 27.77	31.76 / 40.25
C@5	81.71 / 82.55	63.34 / 66.96	53.77 / 61.50
T@1	80.38 / 81.23	81.03 / 80.32	65.46 / 65.90
T@5	90.58 / 88.49	94.81 / 95.09	78.14 / 74.57
MRR@1	0.6799 / 0.7171	0.4061 / 0.4238	0.4006 / 0.4845
MRR@5	0.6954 / 0.7322	0.4472 / 0.4653	0.4389 / 0.5221
NDCG@1	0.6799 / 0.7171	0.4061 / 0.4238	0.4006 / 0.4845
NDCG@5	0.7143 / 0.7441	0.4627 / 0.4851	0.4348 / 0.5172
MAP@1	0.6799 / 0.7171	0.4061 / 0.4238	0.4006 / 0.4845
MAP@5	0.6954 / 0.7322	0.4472 / 0.4653	0.4389 / 0.5221

We observe consistent improvements with QuOTE over the Naive approach across all metrics and datasets, notably in NDCG@5 and MAP@5, indicating better-ranked retrieval results. These metrics validate QuOTE’s effectiveness in accurately ranking and retrieving relevant passages, further supporting findings indicated by Context and Title Accuracy.

4.1.2 Effect of Different Prompts to Generate Questions

A central consideration for QuOTE-style indexing is how the prompt itself influences the *quality* of generated questions. We compare two main prompt templates:

- **Basic Prompt:** Instructs the model to “Generate enough questions to properly capture all the important parts of the text”. The questions are short, direct, and do not include advanced reasoning cues.
- **Complex Prompt:** Adds instructions for more detailed or multi-hop reasoning. In MultiHop-RAG, for example, the complex prompt explicitly requests multi-hop questions referencing multiple pieces of information. In SQuAD or NQ, it encourages short factual queries without referencing the text directly, thereby aiming for more robust coverage of the chunk’s content.

We compare the performance of both these prompts with a naive RAG implementation. As Table 4.3 shows, the Complex Prompt achieves the highest Top-1 Context Accuracy overall. Title Accuracy metrics remain near-perfect across all methods beyond Top-1, indicating that differences among prompts are most pronounced at the paragraph selection level. These observations suggest that more advanced prompting yields modest but meaningful improvements in precisely identifying relevant questions for specific passages.

Table 4.2: Prompt templates for Natural Questions (NQ), SQuAD, and MultiHop-RAG.

Dataset Prompt	+ Prompt Text (Single String)
NQ, SQuAD, Gov Report (Basic)	"Generate numerous questions to properly capture all the important parts of the text. Separate each question-answer pair by a new line only; do not use bullets. Format each question-answer pair on a single line as 'Question? Answer' without any additional separators or spaces around the question mark. Text:\{chunk_text\}"
NQ, SQuAD, Gov Report (Complex)	"Read the following text and generate numerous factual question-answer pairs designed to resemble authentic user search queries and natural language variations. Each question should accurately and semantically capture important aspects of the text, with varying lengths and complexities that mirror real-world search patterns. Include both shorter, keyword-focused questions such as 'who founded Tesla Motors' and longer, natural style questions like 'when did Elon Musk first start Tesla company'. Incorporate 'how' and 'why' questions to reflect genuine user curiosity. Avoid using phrases like 'according to the text' and abstain from pronouns by specifying names or entities. Ensure questions are not overly formal or artificial, maintaining a natural query style. Immediately follow each question with its precise answer on the same line, formatted as 'Question? Answer', without any additional formatting or commentary. Each pair should be on its own line. Text:\{chunk_text\}"
MultiHop-RAG (Basic)	"Generate enough multi-hop questions along with their answers to properly capture all the important parts of the text. These questions should require integrating multiple pieces of information to answer. Separate each question-answer pair by a new line only; do not use bullets. Format each question-answer pair on a single line as 'Question? Answer' without any additional separators or spaces around the question mark. Text:\{chunk_text\}"
MultiHop-RAG (Complex)	"Read the following text and generate complex, multi-hop questions that require integrating multiple pieces of information from the text to answer. The questions should involve reasoning and synthesis, referring to different parts or aspects of the text. Do not use phrases like 'according to the text', 'mentioned in the text', or 'in the text'. All questions should be one sentence long. Never use pronouns in questions; instead, use the actual names or entities. Format each question on a single new line as Question? Answer without any additional separators or spaces around the question mark. Text:\{chunk_text\}"

Table 4.3: Key retrieval metrics across Natural Questions (NQ), SQuAD, and MultiHop-RAG for **Naive**, **Basic**, and **Complex** prompting strategies. Bolded values denote the best performance for each metric.

Method	NQ				SQuAD				MultiHop-RAG			
	C@1	C@5	T@1	T@5	C@1	C@5	T@1	T@5	Full@5	Full@20	Part@5	Part@20
Naive	32.92	89.23	99.85	100.00	82.58	96.00	98.70	99.13	8.00	21.50	29.6	50.0
Basic	34.77	92.46	99.85	100.00	85.82	98.16	99.46	99.89	3.00	16.50	27.7	47.5
Complex	38.00	92.15	99.69	100.00	90.26	98.81	99.68	99.78	8.50	26.50	31.7	54.9

4.1.3 QuOTE Performance vis-a-vis Embedding Model

Table 4.4 compares **Naive** vs. **QuOTE** modes on three datasets—**SQuAD** (single-hop), **NQ** (single-hop), and **MultiHop-RAG** (multi-hop) across a range of datasets. We report Top- k context/title accuracy for SQuAD and NQ, and full/partial match for MultiHop. Despite large differences in baseline quality (e.g., **jinaai** vs. **WhereIsAI** vs. **Alibaba**), note that QuOTE generally improves retrieval metrics (especially Top-1 Context Accuracy or Full@20) *regardless of the underlying embedding model*. QuOTE often raises Top-1 Context Accuracy by 5–17 points on SQuAD and 1–3 points on NQ, and can improve Full@20 by up to several points in MultiHop-RAG. MultiHop-RAG remains challenging, as even large gains may yield relatively modest absolute numbers (e.g., 9% or 10% full match at $k = 5$). However, QuOTE still outperforms or closely matches a naive approach across all embedding models.

4.1.4 Effect of Number of Questions

One key factor in *question-oriented* retrieval is deciding how many questions an LLM should generate for each chunk of text. Generating too few may overlook critical details, while generating too many can introduce redundancy or noise. We therefore tested multiple settings across our three datasets (*Natural Questions*, *SQuAD*, and *MultiHop-RAG*), varying

Table 4.4: Performance of **Naive** vs. **QuOTE** modes on SQuAD, NQ, and MultiHop-RAG across five embedding models. Per-row bolded entries denote the better value for that metric.

Embedding Model	Approach	C@1	C@20	C@1	C@20	Full@5	Full@20	Part@5	Part@20
		SQuAD		NQ					
gte-base-en-v1.5	Naive	59.94	96.35	54.25	91.59	6.50	25.50	29.2	55.7
	QuOTE	77.16	97.68	57.18	92.72	9.00	27.50	32.0	56.9
text-embedding-3-small	Naive	68.59	97.58	51.15	92.59	7.00	32.50	33.6	62.0
	QuOTE	79.07	96.69	51.57	90.16	7.00	27.50	33.7	56.0
UAE-Large-V1	Naive	69.49	96.98	56.84	94.14	4.00	23.00	31.8	57.9
	QuOTE	80.64	97.51	58.35	94.31	9.00	27.00	31.4	54.9
jina-embeddings-v3	Naive	65.45	94.45	53.87	93.76	4.50	25.00	26.0	53.0
	QuOTE	76.97	96.94	55.17	93.72	6.50	24.00	29.9	51.6
MiniLM-L6-v2	Naive	65.53	95.03	50.77	87.23	5.00	20.50	26.8	51.2
	QuOTE	72.35	97.75	51.19	88.57	5.50	20.00	26.9	47.0

the number of questions (1, 5, 10, 15, 20, 30) and also including an ‘LLM decides’ setting. In each case, we measure how **Context Accuracy** and **Title Accuracy** changes, or in the case of MultiHop-RAG, how **Full Match** and **Partial Match** scores are affected.

To systematically investigate the effect of varying the number of generated questions, we parameterize our LLM prompt to either generate:

- **Fixed # Questions:** If a desired quantity *num_questions* is provided, the prompt includes a directive such as:

```
"Generate exactly {num_questions} questions
to properly capture all the important parts
of the text."
```

- **An LLM Decides # Questions:** Here, the LLM is simply instructed to:

```
"Generate enough questions to properly capture
all the important parts of the text."
```

SQuAD Table 4.5 shows that as the number of generated questions per chunk increases from 5 to around 10 or 20, Top-1 Context Accuracy rises from about 73% to as high as 76%, and *Top-5* surpasses 97% in most settings. *Title Accuracy* also remains consistently high, crossing 99% even at Top-1 for 10+ questions. Interestingly, letting the LLM decide how many questions to generate (“LLM Decides”) yields a strong Top-1 Context Accuracy of 76.17% and Top-1 Title Accuracy of 99.30%.

- **Naive vs. 10 questions.** A naive approach (66.60% Top-1 Context) significantly lags behind generating 10 questions (74.91% Top-1), showing that question augmentation dramatically helps correct chunk retrieval.
- **Diminishing returns.** Beyond 10–15 questions, the gains in Top-1 Context Accuracy plateau around 74–76%. For instance, 20 questions achieve 76.66%, comparable to 10 questions at 74.91%.

Table 4.5: Performance comparison across different numbers of generated questions on SQuAD, NQ, and MultiHop-RAG datasets. Results show Context and Title Accuracy at different k values for SQuAD and NQ, and Full/Partial Match for MultiHop-RAG. Bolded entries denote the best performance per metric.

Questions	SQuAD				NQ				MultiHop-RAG			
	C@1	C@5	T@1	T@5	C@1	C@5	T@1	T@5	Full@5	Full@20	Part@5	Part@20
Naive	67.11	90.06	96.45	98.34	32.92	89.23	99.85	100.00	8.00	22.50	29.8	50.7
1 Question	66.02	90.12	96.07	98.58	32.46	91.85	100.00	100.00	15.00	30.00	37.0	61.2
5 Questions	77.05	94.90	97.81	99.24	35.85	92.46	99.69	100.00	12.00	33.00	37.9	61.6
10 Questions	77.58	95.03	97.39	98.41	35.85	92.46	99.54	100.00	16.00	35.00	38.2	62.2
15 Questions	77.22	94.05	96.49	97.47	38.31	90.92	99.69	100.00	14.00	34.00	32.1	60.3
20 Questions	76.24	93.07	95.73	96.45	35.85	92.31	99.69	100.00	14.00	35.00	34.9	63.8
30 Questions	76.05	92.18	95.05	95.73	34.00	90.15	99.54	99.69	11.00	30.00	31.5	58.5
LLM Decides	77.65	93.97	96.30	97.13	38.00	92.15	99.69	100.00	18.00	35.00	41.9	62.7

Natural Questions (NQ) We observe a similar pattern in *Natural Questions* (Table 4.5). The naive approach (and letting the LLM decide automatically) both hover around 61–64% Top-1 Context Accuracy. Generating 15 or 20 questions per chunk can push Top-1

Context Accuracy slightly higher, surpassing 64%. In general, *Title Accuracy* improves more noticeably, approaching or exceeding 79% at Top-1 when 15+ questions are used.

- **Moderate Gains.** Unlike SQuAD, the gains from adding more questions in NQ are more modest (e.g., from 61% to 65% in Top-1 Context Accuracy).
- **Title Accuracy.** By contrast, Title Accuracy climbs above 79% at Top-1 (with 15–20 questions), indicating that question generation consistently helps the system find the right *article*, even if the precise paragraph-level retrieval remains challenging.

MultiHop-RAG Because MultiHop-RAG tasks require retrieving *all relevant documents*, we track both *Full Match Accuracy* and *Partial Match Statistics*. As seen in Table 4.5:

- **Full Match Accuracy.** Baseline naive retrieval (LLM Naive) achieves only about 10% at $k = 5$ and 37% at $k = 20$. Using question generation with 5 or 10 questions can improve the $k = 5$ Full Match from 10–12% to 15–16%, and from 19% to 24–25% at $k = 10$. Even at higher k values (15 or 20), best-case Full Match remains in the 30–35% range, underscoring the challenge of truly multi-hop retrieval. Interestingly, letting the LLM decide (without specifying a question count) yields 18% at $k = 5$ and 35% at $k = 20$.
- **Partial Match.** We also evaluate the *average percentage of required evidence* retrieved. Generating around 5–20 questions consistently pushes partial match rates above 50–60% at $k = 15$ or $k = 20$, compared to 35–45% with naive retrieval. This indicates that even when the system does not achieve a complete full match, it still locates *some* of the essential documents for partial reasoning.

These findings confirm that, while *multi-hop* queries remain significantly harder, carefully

chosen question sets (i.e., 10–20 questions) yield noticeable improvements over a naive approach.

- Generating more questions typically improves retrieval performance, but returns diminish beyond about *10–15 questions per chunk*.
- Even a moderate number of questions (5–10) can outperform naive retrieval by a wide margin in both single-hop (NQ, SQuAD) and multi-hop (RAG) settings.
- In *multi-hop* scenarios, *partial match* is also improved by question generation, indicating that the system at least retrieves some relevant documents more reliably.

Overall, most datasets show a *sweet spot* around 10–15 questions, balancing coverage with potential redundancy. Although letting the LLM fully decide the number of questions can yield strong results in certain cases (e.g., SQuAD), the performance varies by dataset. Consequently, the optimal question count appears to depend on domain complexity and the specifics of the QA task.

4.1.5 Comparison with HyDE

A popular technique for query enrichment is **HyDE**, which generates a hypothetical document at *query time* before embedding it and retrieving relevant chunks. Although HyDE can improve coverage, it requires an LLM call for each incoming query, introducing significant latency. In contrast, **QuOTE** moves question generation to *index time*, incurring a one-time cost but speeding up the overall querying process. We compare **Naive RAG** (no query transformations), **HyDE**, and **QuOTE** on all three benchmarks. For **SQuAD** and **NQ** we showcase *Top-1 Context Accuracy* and for **MultiHop-RAG (multi-hop)**, because queries need multiple pieces of evidence, we focus on *Full Match* at $k = 20$.

Table 4.6 shows that while HyDE sometimes boosts accuracy compared to a Naive approach, its **per-query LLM calls lead to a drastic rise in average retrieval time** (+1–2 seconds per query). By contrast, QuOTE often equals or surpasses Naive’s retrieval accuracy with only a modest query-time overhead, as most of its work is done in the indexing phase.

Overall, these findings illustrate that while **HyDE** can be valuable in certain multi-hop or complex queries, it incurs a substantial latency cost. In fact, **HyDE** can be viewed as primarily a **test-time compute** innovation. **QuOTE** offers significant advantages: higher retrieval accuracy than Naive in most cases, a one-time, amortized cost for question generation, and dramatic query latency improvements over HyDE.

Table 4.6: Comparison of **Naive**, **HyDE**, and **QuOTE** across three QA tasks. The **fastest** (lowest time) and **most accurate** (highest accuracy) entries in each column are **bolded**. In **SQuAD**, Naive is fastest while QuOTE achieves the highest accuracy; for **NQ**, Naive runs fastest while HyDE slightly outperforms the others in accuracy; and in **MultiHop-RAG**, Naive remains fastest, whereas QuOTE attains the highest full-match rates.

Approach	SQuAD				NQ				MultiHop-RAG			
	Time(s)	ms/q	C@1	C@5	Time(s)	ms/q	C@1	C@5	Time(s)	ms/q	Full@5	Full@20
Naive	99.97	108.31	79.31%	95.88%	198.96	187.34	32.92%	89.23%	3.10	15.14	7.00%	23.00%
HyDE	1176.20	1274.32	76.60%	92.63%	2707.56	2549.49	33.23%	90.46%	1155.80	4157.83	6.00%	23.50%
QuOTE	130.56	141.46	90.03%	98.48%	388.38	365.71	38.00%	92.15%	926.41	100.75	8.00%	29.00%

4.1.6 Comparison with Doc2Query-QuOTE

To evaluate whether large language models (LLMs) are essential for high-quality question generation in retrieval pipelines, we introduce a new baseline: ****Doc2Query-QuOTE****. In this variant, we replace the LLM used in QuOTE with a lightweight ****doc2query**** transformer model trained to generate questions from passages. These questions are then used identically to QuOTE—i.e., embedded alongside their source chunk, stored in the vector index, and retrieved with the same deduplication pipeline.

Table 4.7 compares Doc2Query-QuOTE against both Naive and standard QuOTE across three benchmark datasets.

Table 4.7: Comparison of Naive, Doc2Query-QuOTE, and QuOTE across standard retrieval metrics on SQuAD, Natural Questions (NQ), and GovReport.

Metric	SQuAD			NQ			GovReport		
	Naive	Doc2Query	QuOTE	Naive	Doc2Query	QuOTE	Naive	Doc2Query	QuOTE
Top-1 Context Accuracy	59.02	56.51	64.68	27.41	26.87	27.77	31.76	30.09	40.25
Top-5 Context Accuracy	81.71	78.96	82.55	63.34	63.20	66.96	53.77	50.88	61.50
Top-10 Context Accuracy	87.48	84.32	86.19	74.62	67.60	78.65	62.10	53.70	67.84
Top-20 Context Accuracy	91.50	84.65	88.23	81.65	67.71	83.62	69.18	53.73	71.83
Top-1 Title Accuracy	80.38	78.48	81.23	81.03	79.70	80.32	65.46	62.91	65.90
Top-5 Title Accuracy	90.58	88.60	88.49	94.81	94.33	95.09	78.14	74.39	74.57
Top-10 Title Accuracy	93.14	91.04	90.01	96.93	95.73	96.66	81.86	76.21	76.86
Top-20 Title Accuracy	95.08	91.29	91.05	98.13	95.82	97.39	85.19	76.25	78.88

Discussion. While Doc2Query-QuOTE follows the QuOTE pipeline structurally, its performance consistently falls short—not only compared to QuOTE but also the Naive RAG baseline. This underperformance is most pronounced in **Top-1 Context Accuracy**, where Doc2Query-QuOTE lags significantly behind both alternatives across all datasets. These results suggest that simply appending automatically generated questions is not sufficient; the quality and contextual alignment of those questions are critical.

To further investigate the semantic alignment of generated questions with evaluation queries, we compute the cosine similarity between the top-5 generated questions for each passage and the corresponding evaluation query. Figure 4.1 presents the distribution of cosine similarity scores for QuOTE and Doc2Query-QuOTE on the GovReport dataset. Figure 4.2 shows the corresponding results for the SQuAD dataset, demonstrating consistent trends across domains.

We attribute the shortcomings of Doc2Query-QuOTE to two key limitations:

- **Shallow Question Generation:** The doc2query model lacks the reasoning depth and

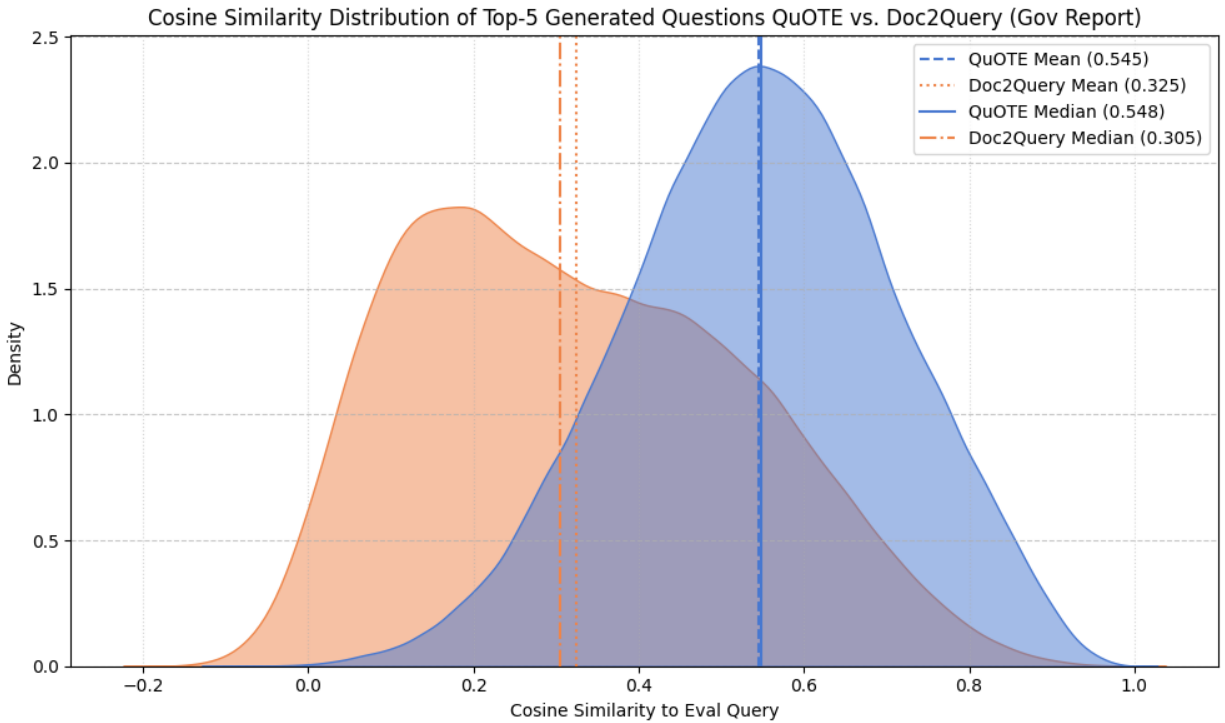


Figure 4.1: Cosine similarity distribution between top-5 generated questions and evaluation queries for QuOTE and Doc2Query on the GovReport dataset. The QuOTE distribution (blue) exhibits higher mean and median similarity values than Doc2Query (orange), indicating that QuOTE produces questions more semantically aligned with the target queries.

linguistic precision of an LLM, often producing generic or loosely related questions. This weakens the semantic link between the query and relevant passage during retrieval.

- **Lack of Answer Supervision:** Unlike QuOTE, which pairs each question with an LLM-generated answer, Doc2Query-QuOTE relies on ungrounded queries alone. This omission reduces the precision of chunk representations and limits their utility during top-k retrieval.

These findings underscore the necessity of LLM involvement in both question and answer generation for effective question-oriented indexing. QuOTE’s performance gains stem not only from augmenting document representations with questions, but from doing so with

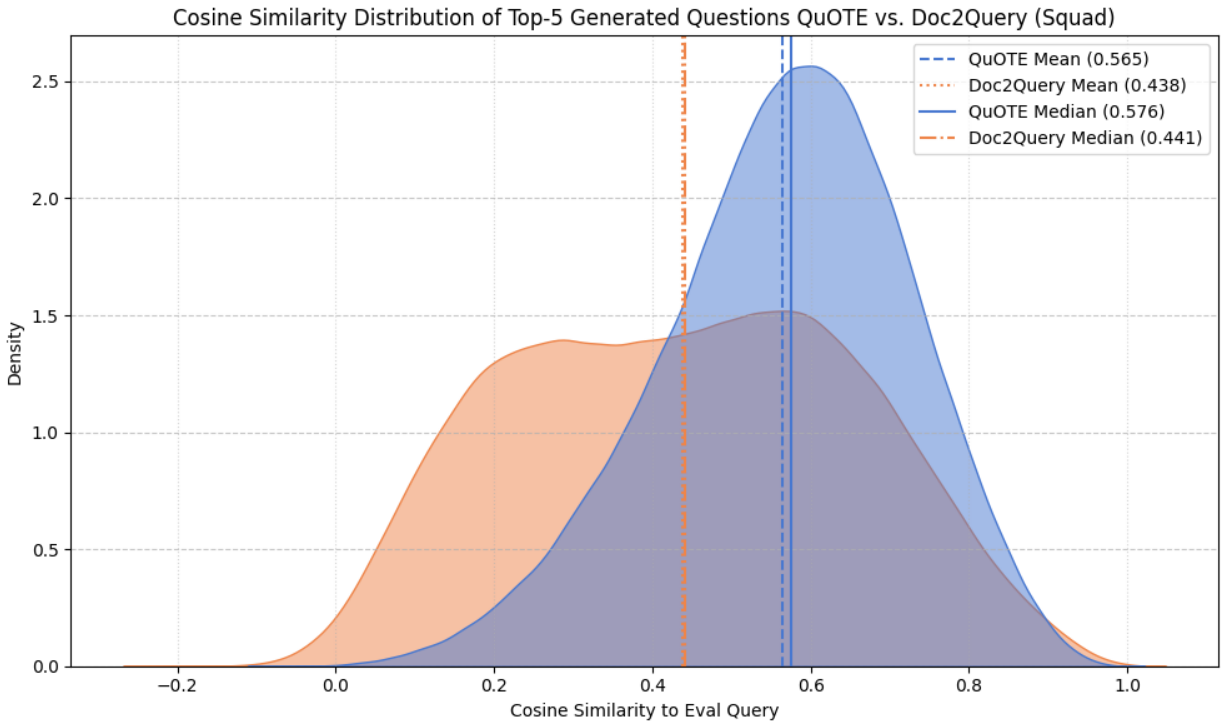


Figure 4.2: Cosine similarity distribution between top-5 generated questions and evaluation queries for QuOTE and Doc2Query on the SQuAD dataset. Similar to the GovReport results, the QuOTE distribution shows higher semantic alignment with target queries than Doc2Query.

high-quality, context-sensitive, and answer-aware generations.

4.1.7 Evaluation of Contextual Retrieval

Motivation. Contextual Retrieval is a recent method introduced by Anthropic that enhances document chunk embeddings by prepending chunk-specific explanatory context before embedding. The goal is to create more semantically rich and situationally aware representations—termed *Contextual Embeddings*—that better align with user queries.

Implementation. In our pipeline, we adopted the contextual embedding strategy by prompting an LLM with both the full document and each chunk. The prompt was adapted from Anthropic’s original specification, where Claude is instructed to generate a short snippet situating the chunk within the document context:

```
<document>
```

```
{{WHOLE_DOCUMENT}}
```

```
</document>
```

```
Here is the chunk we want to situate within the whole document
```

```
<chunk>
```

```
{{CHUNK_CONTENT}}
```

```
</chunk>
```

```
Please give a short succinct context to situate this chunk  
within the overall document for the purposes of improving  
search retrieval of the chunk. Answer only with the succinct  
context and nothing else.
```

This generated context is then prepended to the chunk before vector embedding. Unlike Anthropic’s method, we do **not** apply this technique to a BM25 index—we evaluate purely in the vector retrieval setting.

Limitations. Due to the computational overhead of including the entire document per chunk prompt, we restricted this evaluation to a 1000-document subset of each dataset. Compared to QuOTE, which augments once per chunk using isolated prompts, Contextual Retrieval requires a much larger prompt context window, increasing latency and API costs significantly.

Results. Table 4.8 compares Contextual Retrieval against Naive and QuOTE pipelines for both SQuAD and GovReport datasets. While Contextual Retrieval achieves moderate accuracy, it consistently underperforms compared to QuOTE and in some cases even the Naive baseline. This is likely due to variability in generated context quality and limited benefit from the added explanatory text when semantic similarity already suffices.

Table 4.8: Performance comparison of Contextual Retrieval against Naive and QuOTE pipelines on a 1000-document subset of SQuAD and GovReport datasets.

Metric	SQuAD			GovReport		
	Naive	Contextual	QuOTE	Naive	Contextual	QuOTE
C@1	72.76	71.23	77.14	38.49	36.38	50.51
C@5	92.07	91.56	92.38	63.68	63.40	76.13
C@10	96.03	95.41	94.78	74.12	73.56	83.60
C@20	98.53	97.96	95.75	82.92	82.97	87.65
T@1	95.67	94.66	94.74	88.31	88.10	88.66
T@5	98.41	97.76	96.49	93.59	93.44	91.97
T@10	99.11	98.73	96.68	95.32	95.32	93.37
T@20	99.52	99.21	96.92	96.80	96.85	94.33

Summary. While Contextual Retrieval introduces a compelling conceptual shift, the cost-performance tradeoff is notable. On this 1000-document subset, it did not outperform existing QuOTE or Naive strategies. Future improvements may include hybrid models that combine context-enhanced indexing with question-oriented augmentation or apply context conditioning selectively based on document structure.

4.1.8 Effect of Deduplication

Deduplication is essential in **QuOTE** because each chunk can be indexed multiple times—once per generated question—leading to redundant matches at query time. Again, We

Table 4.9: Comparison of retrieval approaches. Index=time to build database (seconds), Query=time to process all queries (seconds), ms/q=milliseconds per query, C@1=Context accuracy, T@1=Title accuracy. Bold indicates best per column.

App	Index	Query	ms/q	C@1	T@1
Naive	10.36	99.97	108.31	79.31%	96.64%
HyDE	10.14	1176.20	1274.32	76.60%	94.58%
QuOTE	170.32	130.56	141.46	90.03%	98.37%

compare **Naive RAG**, **HyDE** and **QuOTE**. When $k = 1$, deduplication is unnecessary, as only one chunk is retrieved. However, when $k \in \{5, 10, 20\}$, QuOTE systems fetch more than k results from the vector index (e.g., $k \times 5$) and then deduplicate by original chunk text. This extra step introduces a small overhead, but we find that QuOTE remains much faster than HyDE (which invokes an LLM at *each* query) and substantially outperforms Naive in Top-1 Context Accuracy.

Table 4.9 shows a head-to-head comparison of the three approaches on a SQuAD subset. Each approach processes 923 queries for all benchmarks, we see that **Naive** is the fastest and **QuOTE** the most accurate. The added overhead incurred by **QuOTE** over **Naive** is small relative to the cost of *per-query* generation in HyDE. Hence, **QuOTE** obtains both *superior accuracy* and *faster query times* than HyDE, while incurring a one-time cost for indexing. In settings where repeated queries are common, paying a higher index-time cost can significantly improve responsiveness and end-user experience.

4.1.9 What Happens If We Combine All Questions Into One Chunk?

To further investigate the necessity of QuOTE’s deduplication step, we introduce a variant we term **QuOTE-OneDoc**. Instead of storing a separate (question + chunk) embedding per question, QuOTE-OneDoc aggregates all questions generated for a chunk and appends

them to that chunk as a single unified document. This method eliminates the need for deduplication at query time, potentially reducing index size and simplifying retrieval.

We evaluate QuOTE-OneDoc across three benchmark datasets— SQuAD, Natural Questions (NQ), and GovReport—and compare it to both the original QuOTE and the Naive baseline.

Index Size. Table 4.10 illustrates the dramatic reduction in index size using QuOTE-OneDoc compared to the standard QuOTE approach.

Table 4.10: Total number of documents in the index under each method.

Method	SQuAD	NQ	GovReport
QuOTE-OneDoc	18,891	7,368	9,960
QuOTE	237,977	106,384	134,364

Retrieval Accuracy. Table 4.11 summarizes the retrieval performance across the three datasets.

Table 4.11: Comparison of retrieval performance across Naive, QuOTE-OneDoc, and QuOTE variants. Bold indicates best performance per column.

Metric	SQuAD			NQ			GovReport		
	Naive	OneDoc	QuOTE	Naive	OneDoc	QuOTE	Naive	OneDoc	QuOTE
C@1	59.02	55.32	64.68	27.41	26.08	27.77	31.76	33.50	40.25
C@5	81.71	78.19	82.55	63.34	65.11	66.96	53.77	56.93	61.50
C@10	87.48	84.41	86.19	74.62	77.15	78.65	62.10	65.37	67.84
C@20	91.50	89.08	88.23	81.65	84.75	83.62	69.18	72.20	71.83
T@1	80.38	78.42	81.23	81.03	81.95	80.32	65.46	66.63	65.90
T@5	90.58	89.09	88.49	94.81	96.29	95.09	78.14	79.21	74.57
T@10	93.14	91.93	90.01	96.93	98.19	96.66	81.86	82.93	76.86
T@20	95.08	94.16	91.05	98.13	98.95	97.39	85.19	86.25	78.88

Discussion. While QuOTE-OneDoc achieves moderate improvements over the Naive baseline—especially in GovReport and NQ—the standard QuOTE variant consistently outperforms OneDoc across most Top- k retrieval metrics. This suggests that distributing individual (question + chunk) pairs across separate embeddings, despite increasing index size and necessitating deduplication, contributes substantially to QuOTE’s retrieval accuracy.

These findings highlight a critical tradeoff between retrieval precision and index size. While QuOTE-OneDoc reduces storage and may simplify deployment, the standard QuOTE method remains superior in accuracy, particularly for challenging single-hop tasks like SQuAD.

4.1.10 Effect of Question–Question Similarity Pruning

When generating multiple questions per chunk, many of those questions can be highly redundant, inflating the index size without corresponding gains in retrieval performance. To address this, we apply a *question–question* (Q–Q) similarity filter: after initial question–context (Q–C) pruning, we compute pairwise cosine similarities among the remaining questions and discard any question whose similarity to an earlier one exceeds a threshold. This retains semantic diversity while reducing index size. These pruning experiments were conducted on a subset of each dataset; the resulting document counts therefore differ from the full-index sizes reported in Section 4.1.9. Tables 4.12, 4.13, and 4.14 show the impact on SQuAD, Natural Questions, and GovReport benchmarks. Best values in each column are **bolded**.

Table 4.12: SQuAD: Retrieval performance under varying Q–Q similarity thresholds.

Mode	QQ	Docs	C@1	C@5	C@10	C@20	T@1	T@5	T@10	T@20
Naive	N/A	789	72.76%	92.07%	96.03%	98.53%	95.65%	98.39%	99.11%	99.52%
Quote (none)	—	9,931	77.33%	92.43%	94.86%	95.84%	94.88%	96.42%	96.73%	97.09%
Quote (0.3)	0.3	1,402	71.44%	91.25%	95.75%	97.88%	94.88%	98.51%	99.25%	99.52%
Quote (0.5)	0.5	2,904	74.95%	93.53%	96.37%	98.20%	95.91%	98.46%	99.16%	99.40%
Quote (0.7)	0.7	6,330	77.28%	93.32%	95.96%	97.26%	95.72%	97.60%	98.05%	98.25%

Table 4.13: Natural Questions: Retrieval performance under varying Q–Q similarity thresholds.

Mode	QQ	Docs	C@1	C@5	C@10	C@20	T@1	T@5	T@10	T@20
Naive	N/A	776	23.83%	56.59%	70.38%	79.14%	97.89%	99.30%	99.75%	99.96%
Quote (none)	—	11,595	24.91%	61.30%	75.96%	83.06%	97.48%	99.05%	99.38%	99.59%
Quote (0.3)	0.3	1,737	24.20%	60.68%	75.59%	84.88%	98.02%	99.42%	99.88%	100%
Quote (0.5)	0.5	3,853	23.63%	61.21%	76.91%	86.58%	97.98%	99.55%	99.92%	100%
Quote (0.7)	0.7	7,645	24.00%	60.55%	76.95%	86.16%	97.77%	99.38%	99.75%	99.88%

Table 4.14: GovReport: Retrieval performance under varying Q–Q similarity thresholds.

Mode	QQ	Docs	C@1	C@5	C@10	C@20	T@1	T@5	T@10	T@20
Naive	N/A	787	38.49%	63.70%	74.15%	82.97%	88.31%	93.59%	95.32%	96.80%
Quote (none)	—	10,499	50.31%	75.98%	83.15%	87.49%	88.66%	92.02%	93.44%	94.38%
Quote (0.3)	0.3	1,995	42.73%	68.96%	78.83%	86.86%	90.04%	95.07%	96.59%	97.76%
Quote (0.5)	0.5	4,223	47.18%	74.12%	82.92%	89.60%	90.34%	94.31%	95.86%	96.98%
Quote (0.7)	0.7	8,315	50.86%	76.66%	84.09%	89.10%	89.32%	92.78%	94.15%	95.32%

Across all three benchmarks (SQuAD, Natural Questions, GovReport), moderate pruning (QQ = 0.5) reduces the index size by over 70% on SQuAD, over 65% on Natural Questions, and over 60% on GovReport, while preserving or improving Top-5 and higher- k Context Accuracy and Title metrics. Aggressive pruning (QQ = 0.3) yields the smallest index with only modest Top-1 drops and strong gains at higher k . Notably, on GovReport, QQ = 0.7 not only shrinks the index relative to unpruned QuOTE but also **outperforms the unpruned Quote baseline across all metrics**, demonstrating that semantic deduplication can boost retrieval while reducing storage.

4.1.11 Can we use a Cheaper LLM for Question Generation?

An important practical consideration in RAG-based pipelines is whether *cheaper, smaller models* can generate effective questions for indexing, or if premium, large-scale LLMs (e.g., GPT-4) are necessary. To investigate, we experimented with a variety of local language models (e.g., gemma2-9b, llama3-8b, and qwen2.5-7b), as well as gpt-4o-mini, gpt-4o, and

Table 4.15: Comparison of different models on a SQuAD subset. We report Context Accuracy (C@k) and Title Accuracy (T@k) at k=1 and k=5. Best value(s) in each column are **bolded**.

Model	C@1	C@5	T@1	T@5
Naive	66.60	92.80	98.32	99.44
QuOTE gemma2-9b	74.49	96.93	99.09	99.72
QuOTE gpt-4o-mini	76.73	97.20	98.81	99.30
QuOTE gpt-4o	76.38	96.72	99.09	99.58
QuOTE llama3-8b	73.58	95.95	99.09	99.79
QuOTE llama3.1-8b	71.42	96.09	99.09	99.72
QuOTE llama3.2-3b	70.86	94.90	98.81	99.65
QuOTE phi4	74.07	95.95	98.81	99.30
QuOTE qwen2.5-7b	72.05	96.23	99.02	99.72

a baseline **Naive** approach that relies solely on the chunk text without question generation. All runs were conducted on a *SQuAD-based subset*. Table 4.15 summarizes the results in terms of **Top-*k* Context Accuracy** and **Top-*k* Title Accuracy**.

We observe that even smaller models such as llama3.2-3b achieve over 70% Top-1 Context Accuracy—only a few percentage points behind the more capable gpt-4o-mini or gpt-4o models. For nearly all models, *Top-1 Title Accuracy* remains around or above 98%, indicating that the question generation step—regardless of the model size—helps QuOTE hone in on the correct article. Once we allow for more retrieved chunks (Top-10 or Top-20), nearly all approaches exceed 98–99% Context Accuracy. This suggests that question augmentation significantly reduces misalignment with relevant passages. The main trade-off is that gpt-4o and gpt-4o-mini exhibit slightly higher Top-1 Context Accuracy (up to $\sim 76\%$) compared to cheaper models (70–74%); however, local LLMs still offer near-parity in mid- to high-*k* retrieval settings with notably lower inference cost.

4.1.12 Effect of the Number of Contexts on Retrieval Accuracy

Analysis of the impact of the number of contexts on retrieval performance reveals distinct patterns between SQuAD and NQ, reflecting their fundamentally different dataset characteristics.

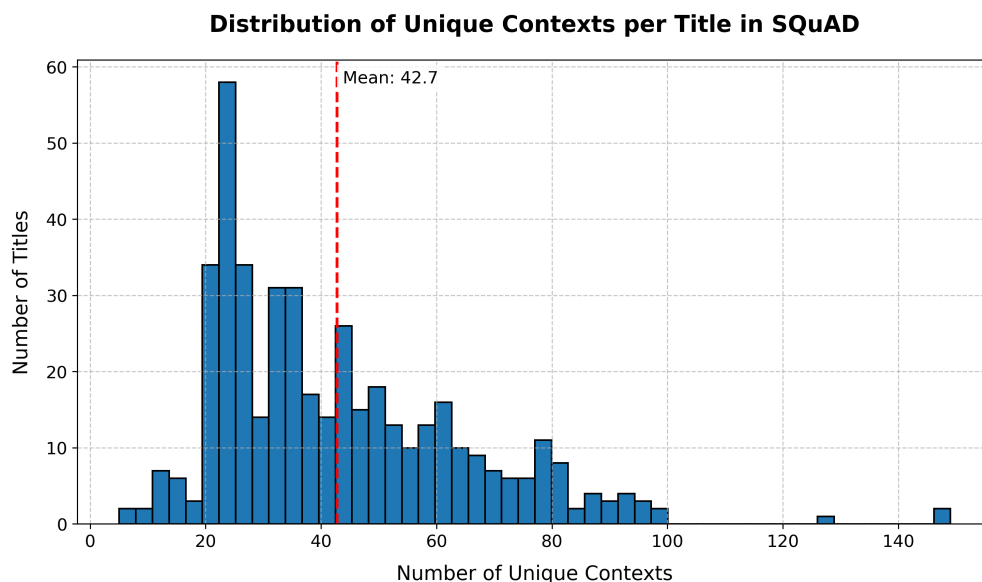


Figure 4.3: Distribution of contexts per title in SQuAD (N=442 titles). The mean of 42.74 contexts per title and maximum of 149 contexts demonstrate the dataset’s high context density.

SQuAD exhibits a rich context structure, with 442 titles having a mean of 42.74 contexts per title (median=36). This substantial density, ranging from 5 to 149 contexts per title, creates significant potential for confusion with naive retrieval approaches, particularly when similar passages exist within the same document.

In stark contrast, NQ presents a much sparser context landscape. Across its 48,525 titles, NQ maintains a mean of just 1.52 contexts per title, with a median of 1, indicating that most titles have unique contexts.

These structural differences manifest clearly in QuOTE’s relative performance gains (Figure

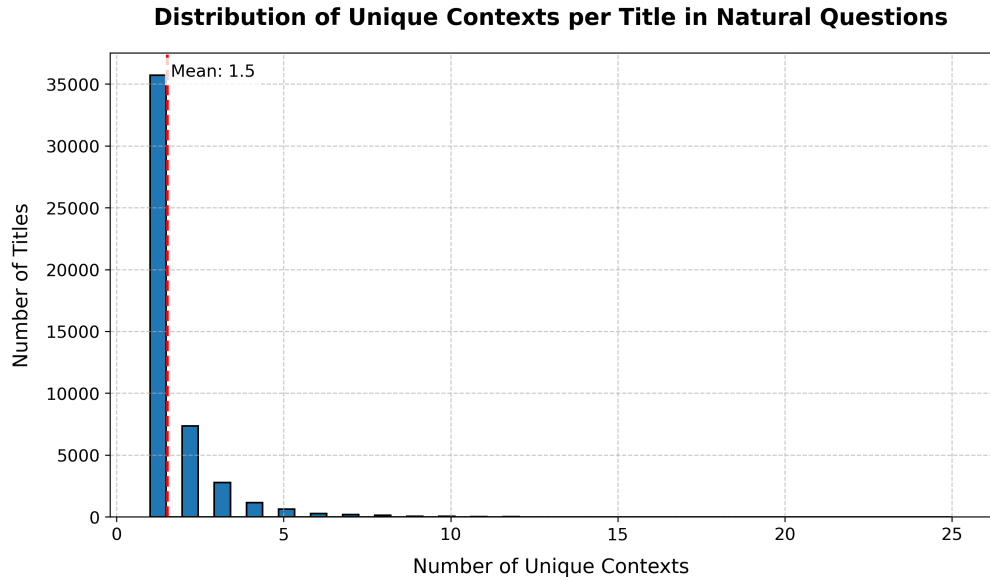


Figure 4.4: Distribution of contexts per title in Natural Questions (N=48,525 titles). The highly concentrated distribution around a median of 1 context per title indicates predominantly singular contexts.

4.5). For SQuAD, we observe a steady increase in QuOTE’s advantage as the number of contexts grows. Starting from around 8% improvement in a small number of contexts, it increases consistently to about 16% at moderate number of contexts, and continues to improve to exceed the improvement 20% for a larger number of contexts. This steady improvement trend aligns with the increasing challenge of disambiguating similar contexts in longer documents.

NQ shows a more constrained but generally positive pattern, reflecting its simpler context structure. The improvements start at about 6% for a small number of contexts and reach peaks of approximately 18% for a moderate number of contexts. Although the magnitude of the improvements varies, QuOTE consistently enhances the retrieval accuracy in most contexts, although its impact is more variable than in SQuAD.

These insights demonstrate how dataset characteristics fundamentally influence QuOTE’s effectiveness. For collections with many contexts per document like SQuAD, QuOTE pro-

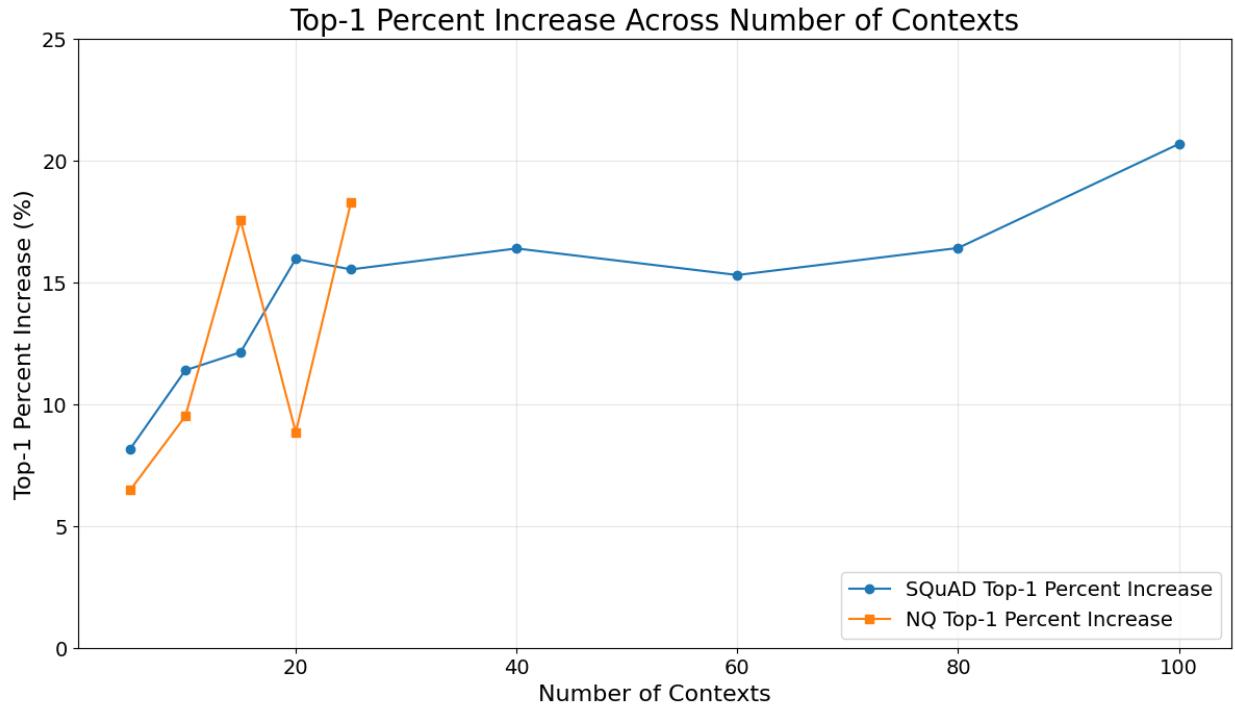


Figure 4.5: Percentage increase in Top-1 retrieval accuracy with QuOTE compared to naive retrieval across the number of contexts. SQuAD shows steady improvement that grows with the number of contexts, reaching 20.7% improvement at size 100, while NQ shows consistent but variable gains up to 18.3%.

vides increasingly valuable disambiguation as document length grows. For collections like NQ where most documents contain just a single relevant chunk, QuOTE still provides consistent benefits, though the magnitude varies with the number of contexts.

Chapter 5

Conclusion

5.1 Discussion

This work has demonstrated that augmenting document representations with synthetic questions can significantly enhance retrieval accuracy in retrieval-augmented generation (RAG) pipelines. By embedding multiple question–chunk pairs per passage, QuOTE improves alignment between user queries and indexed content, especially in cases requiring fine-grained or semantically diverse information access. Importantly, although our approach introduces the need for deduplication at query time, this overhead is minimal and is offset by notable gains in retrieval quality across single-hop and multi-hop benchmarks.

The improvements hold across a variety of embedding models and prompt strategies, reinforcing the generality of our approach. Moreover, we show that prompt design, question count, and redundancy filtering (via question–question pruning) all play a meaningful role in optimizing performance. Even with smaller or open-source models used for question generation, QuOTE yields measurable gains, indicating broad accessibility and practicality of our method.

QuOTE also advances a shift in perspective: treating *question generation as an indexing strategy*, not merely a downstream QA task. This reframing opens up new possibilities for retrieval systems that are better attuned to user intent.

5.2 Future Work

There are several promising directions for extending this work:

Self-Improving Indexing. An adaptive variant of QuOTE could monitor user interactions—e.g., which retrieved contexts were clicked or led to correct answers and selectively incorporate new (query, context) pairs. Such a system might use LLM fine-tuning to refine future question generation or directly embed frequently asked or corrected queries into the index. This would yield a dynamic, feedback-driven retriever that evolves over time.

Automated Prompt Optimization. While this work used static prompt templates, future versions of QuOTE could explore prompt search or reinforcement learning-based tuning. Automatically optimizing prompts to maximize coverage and semantic specificity of generated questions could increase the utility of each chunk-level embedding and improve domain transferability.

Hybrid Indexing. QuOTE currently generates question embeddings for every chunk, but some documents may already be well-represented without augmentation. A future extension might embed a mix of raw and question-augmented chunks, based on content type, document complexity, or empirical performance. Developing scaling laws to understand trade-offs between index size, number of questions, and retrieval accuracy would support the design of such hybrid systems.

Multimodal Extension. A particularly exciting avenue is applying QuOTE to **multimodal data**. Using vision-language models (VLMs), one could generate natural language questions about images, charts, or diagrams, and embed those question–image pairs into

a multimodal vector index. This enables retrieval over complex datasets—such as technical manuals, visual documentation, or educational content—by leveraging question-oriented indexing for non-textual content. For instance, a question like “What is the pressure reading on the gauge in the top right corner?” could be associated with an image and stored alongside its embedding, enabling image-based QA without requiring full image captioning. Together, these directions position QuOTE as a foundation for retrieval systems that are not only more accurate, but more adaptable, multimodal, and intelligent in how they represent and access knowledge.

Bibliography

- [1] Anthropic contextual retrieval. <https://www.anthropic.com/news/contextual-retrieval>.
- [2] Raviteja Anantha, Tharun Bethi, Danil Vodianik, and Srinivas Chappidi. Context tuning for retrieval augmented generation. *arXiv preprint arXiv:2312.05708*, 2023.
- [3] Xilun Chen, Kushal Lakhotia, Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. Salient phrase aware dense retrieval: can a dense retriever imitate a sparse one? *arXiv preprint arXiv:2110.06918*, 2021.
- [4] Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *arXiv preprint arXiv:2405.13792*, 2024.
- [5] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729, 2024.
- [6] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.
- [7] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin

- Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.
- [8] Michael Heilman and Noah A. Smith. Ranking automatically generated questions as a shared task. In *Proceedings of the AIED Workshop on Question Generation*, Brighton, UK, July 2009.
- [9] Michael Heilman and Noah A. Smith. Good question! statistical ranking for question generation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Los Angeles, CA, June 2010.
- [10] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.
- [11] Mathew Jacob, Erik Lindgren, Matei Zaharia, Michael Carbin, Omar Khattab, and Andrew Drozdov. Drowning in documents: Consequences of scaling reranker inference. *arXiv preprint arXiv:2411.11767*, 2024.
- [12] Ziyang Jiang, Xueguang Ma, and Wenhui Chen. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024.
- [13] Rajat Khanda. Agentic ai-driven technical troubleshooting for enterprise systems: A novel weighted retrieval-augmented generation paradigm. *arXiv preprint arXiv:2412.12006*, 2024.
- [14] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

- [15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- [16] Alexandria Leto, Cecilia Aguerrebere, Ishwar Bhati, Ted Willke, Mariano Tepper, and Vy Ai Vo. Toward optimal search and retrieval for rag. *arXiv preprint arXiv:2411.07396*, 2024.
- [17] Jiawei Li, Mucheng Ren, Yang Gao, and Yizhe Yang. Ask to understand: Question generation for multi-hop question answering. In *China National Conference on Chinese Computational Linguistics*, pages 19–36. Springer, 2023.
- [18] Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip S Yu. Dense hierarchical retrieval for open-domain question answering. *arXiv preprint arXiv:2110.15439*, 2021.
- [19] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.
- [20] Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. Multi-hop question answering. *arXiv preprint arXiv:2204.09140*, 2022. URL <https://doi.org/10.48550/arXiv.2204.09140>. Published at Foundations and Trends in Information Retrieval.
- [21] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [22] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’t know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

- [23] Keshav Rangan and Yiqiao Yin. A fine-tuning enhanced rag system with quantized influence measure as ai judge. *Scientific Reports*, 14(1):27446, 2024.
- [24] Benjamin Reichman and Larry Heck. Dense passage retrieval: Is it retrieving? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13540–13553, 2024.
- [25] Tolga Şakar and Hakan Emekci. Maximizing rag efficiency: A comparative analysis of rag methods. *Natural Language Processing*, pages 1–25, 2024.
- [26] Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24*, page 2395–2400, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657957. URL <https://doi.org/10.1145/3626772.3657957>.
- [27] Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. Simple entity-centric questions challenge dense retrievers. *arXiv preprint arXiv:2109.08535*, 2021.
- [28] Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17, 2023.
- [29] Mingyang Song and Mao Zheng. A survey of query optimization in large language models. *arXiv preprint arXiv:2412.17558*, 2024.
- [30] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

- [31] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*, 2023.
- [32] Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, et al. Searching for best practices in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17716–17736, 2024.
- [33] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*, 2024.
- [34] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.
- [35] Hamed Zamani, Michael Bendersky, Donald Metzler, Honglei Zhuang, and Xuanhui Wang. Stochastic retrieval-conditioned reranking. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 81–91, 2022.
- [36] 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. *arXiv preprint arXiv:2402.19473*, 2024.
- [37] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143, 2023.