Question-Based Retrieval using Atomic Units for Enterprise RAG

Anonymous ACL submission

Abstract

Enterprise retrieval augmented generation (RAG) offers a highly flexible framework for combining powerful large language models (LLMs) with internal, possibly temporally changing, documents. In RAG, documents are first chunked. Relevant chunks are then retrieved for a specific user query, which are passed as context to a synthesizer LLM to generate the query response. However, the retrieval step can limit performance, as incorrect chunks can lead the synthesizer LLM to generate a false response. This work proposes a zero-shot adaptation of standard dense retrieval steps for more accurate chunk recall. Specifically, a chunk is first decomposed into atomic statements. A set of synthetic questions are then generated on these atoms (with the chunk as the context). Dense retrieval involves finding the closest set of synthetic questions, and associated chunks, to the user query. It is found that retrieval with the atoms leads to higher recall than retrieval with chunks. Further performance gain is observed with retrieval using the synthetic questions generated over the atoms. Higher recall at the retrieval step enables higher performance of the enterprise LLM using the RAG pipeline.

1 Introduction

Since the recently popularized ChatGPT as the first instruction-finetuned large language model (LLM) deployed at scale to the lay market, there has been a substantial uptake on the interest of businesses to incorporate LLMs in their products for a variety of downstream tasks (Bahrini et al., 2023; Castelvecchi, 2023; Badini et al., 2023; Kim and Min, 2024). For most companies, they are interested in using such models as enterprise LLMs where the model can handle queries related to proprietary on-premise data.

It has been repeatedly demonstrated that these LLMs have general (public) knowledge implicitly embedded in their parametric memory which can be extracted upon querying (Yu et al., 2023a). However, the LLMs do not have implicit knowledge about a specific enterprise's textual database in a custom domain and hence are prone to hallucinate in such situations (Xu et al., 2024b; Yu et al., 2023b). Additionally, the transformer-based (Vaswani et al., 2017) LLMs typically have a limited context window (due to quadratic order in cost of the attention mechanism), which means information for a specific company to be queried over cannot be directly fed-in as a prompt to the LLM. Due to limited budget, it is typically not feasible to finetune LLMs on a specific enterprise's data. In particular, with evolving data from ongoing projects, it is challenging to maintain a constantly updated company-specific LLM finetuned on new data without catastrophic forgetting on old data (Luo et al., 2023).

To tackle this issue, and with retrieval augmented generation (RAG) proposed initially by Lewis et al. (2020), RAG-inspired systems have rapidly become the de-facto as a zero-shot solution for enterprise LLMs. At the essence, there are 2 steps: 1. retrieval and 2. synthesis. Documents are split into independent chunks, and a retrieval process is applied to identify the relevant chunks to a given query. The retrieved chunks (which should fit into the context window) with the query are passed as the prompt to the synthesizer LLM to get the desired response.

Currently, the bottleneck for most enterprise LLMs is the retrieval step, where the correct information is not retrieved for the LLM to answer the question (Arora et al., 2023). Hence, this work focuses on building upon zero-shot approaches to improve the retrieval step for RAG. A potential limitation of the RAG set-up is that an embedding model is used to retrieve the relevant chunks efficiently when given a query. Each pre-calculated chunk has its corresponding embedding stored in memory, which allows the closes chunks to be re-

trieved by embedding the incoming query into the same space. However, there is a mismatch in trying to match the space of queries and chunks as each chunk can carry a large amount of information.

Instead, our work looks to represent each chunk as a set of atomic pieces of information. Min et al. (2023) introduced atomization of text for improving the assessment of summary consistency. These atoms can be structural (e.g. sentences of a chunk) or unstructured where a set of atoms is generated for any chunk. By embedding the atoms instead of the chunks themselves, the relevant atoms can instead be identified (that correspond to a specific set of chunks) for the posed query in the embedding space. The atomic breakdown of the chunk enables more accurate retrieval.

We further identify that even with the atomic embedding representations of the chunk, a given atom and the query do not necessarily best align for retrieval as the former is a statement with a piece of information while the latter is a question about locating a missing piece of information. Thus, we propose generating synthetic atomic questions. Each atom has a set of questions generated (with the chunk text as contextual information), which in turn are embedded. Therefore, the embedded incoming query is used to identify the closest set of atomic questions which in turn point to the relevant set of chunks to be passed to the synthesizer LLM in the RAG pipeline.

As enterprise RAG operates over a closed set of documents, the generation of the atoms and corresponding synthetic questions is a one-off cost. Similarly, the increased set of embeddings to search over for the closest matches for the query embedding is of less concern given the various very efficient algorithms for embedding search such as FAISS (Douze et al., 2024).

Current information retrieval approaches in the RAG pipeline look at improving the quality of dense retrieval through generation augmented retrieval (GAR), where a query is rewritten for high recall retrieval. However, we focus our attention on representing the chunks more efficiently for retrieval. The contributions are summarized as:

- An exploration of how the retrieval step in the RAG pipeline is improved with structured and unstructured atomic representation of a document chunk.
- Demonstrate further improvement in retrieval with the generation of atomic questions.

2 Related Work

In recent months, several works have extended RAG in various directions (Zhao et al., 2024). Many approaches finetune the components of the RAG pipeline. For example, Siriwardhana et al. (2023) explore adapting end-to-end RAG systems for open-domain question-answering while Zhang et al. (2024) introduce RAFT for finetuning RAG systems on specific domains by learning to exclude distractor documents. Additionally, Siriwardhana et al. (2021); Lin et al. (2023) jointly train the retriever and the generator for target domains. However, our work focuses on exploring zero-shot solutions as finetuning can be a computationally infeasible procedure for many enterprises.

In terms of zero-shot approaches, there have been several extensions proposed. Gao et al. (2023a) propose hypothetical document embedding (HyDE) where an LLM is used to transform the input query into an answer form (hallucinations are acceptable) for improved dense retrieval over the chunks. Similarly, Wang et al. (2023b) suggest a query expansion approach termed query2doc where an LLM is used to expand the query (Jagerman et al., 2023) with a pseudo-generated document, which they demonstrate to be effective for dense retrieval. Alternatively, we propose approaches that focus on modifying the knowledge base on which retrieval is performed rather than modifying the user queries as is common in GAR (Shen et al., 2023; Feng et al., 2023; Arora et al., 2023).

Song et al. (2024) retrieve a superfluous number of chunks during the retrieval step. They then re-rank the retrieved chunks with a re-ranker system to identify the most relevant set. Similarly, Wang et al. (2023c) propose FILCO to filter out the retrieved documents as an additional step in the RAG pipeline. Sun et al. (2023) explore the zero-shot use of LLMs as alternatives for traditional re-rankers. Arora et al. (2023) additionally incorporate the re-rank steps with GAR in an iterative feedback loop. Leveraging the comparative abilities of LLMs, Qin et al. (2023) propose using pairwise comparisons for the re-ranking of retrieved documents. Alternatively, Sarthi et al. (2024) propose RAPTOR as an iterative technique to pass a summarized context (based on the retrieved documents) to the synthesizer. Iter-RetGen by Shao et al. (2023) follow a similar iterative summarization strategy with LLMs. Finally, ActiveRAG (Xu et al., 2024a) encourages the synthesizer to consider parametric

memory rather than just relying on the set of retrieved documents. Gao et al. (2023b) summarize all advanced RAG approaches as additional pre-retrieval or post-retrieval steps. Pre-retrieval steps include query routing, query re-writing and query expansion. Post-retrieval steps include re-ranking summarization and fusion. The synthetic question retrieval over atomized units from the document set is a form of pre-retrieval that operates on the knowledge store rather than on the user query. Hence, our work remains complementary with all forms post-retrieval advanced RAG.

Traditionally, retrieval of relevant documents for a given query has been well studied (Hambarde and Proenca, 2023) with approaches such as BM25 (Robertson et al., 2009). In recent years, dense retrieval approaches have dominated as efficient retrieval processes where queries and documents are represented as dense vectors (embeddings) and documents are retrieved based on the similarity between these vectors. Semantically meaningful vectors have been possible with the series of regularly updated sentence transformers for generating general purpose embeddings including Sentence-BERT (Reimers and Gurevych, 2019), ConSERT (Yan et al., 2021), SimCSE (Gao et al., 2021), DiffCSE (Chuang et al., 2022), sentence-T5 (Ni et al., 2022) and E5 (Wang et al., 2022). More recently, there have been a series of more powerful embedding models that adapt instruction-finetuned language models as embedders (Li et al., 2023; Meng et al., 2024; Muennighoff et al., 2024; Wang et al., 2023a; BehnamGhader et al., 2024). Therefore, this work restricts exploration to dense retrieval.

3 Retrieval for RAG

In enterprise RAG systems, the core pipeline can be summarized as follows.

1. **Split**: Given a textual corpus of documents, a set of chunks are generated by splitting all text into distinct paragraphs.

2. **Retrieve**: For a given user query, the relevant set of chunks are retrieved.

3. **Synthesize**: The original query and the retrieved chunks are passed to a synthesis model to generate a response to the query using the provided chunk information as the context.

Here, the focus is on improving the retrieval step of the enterprise RAG pipeline. For the scope of the data considered in this work, we assume that the answer to a specific query is present in only one chunk (i.e. there are no unanswerable queries and multiple chunks are not required to deduce the answer to a question). Therefore, the retrieval step task can be defined as follows:

Task Let $R(q;c) \in 0,1$ denote an oracle relevancy function that returns 1 if a chunk, c, contains the answer to the user query q and 0 otherwise. Given a set of N chunks, $\{c\}_{1:N}$, and a user query q, retrieve chunk c_k such that $R(q;c_k) = 1$ but $\sum_{i \neq k} R(q;c_i) = 0$.

Next, we describe the various approaches for the retrieval step of enterprise RAG systems. The focus is on zero-shot approaches that can be applied without any training and we assume we have no-cost in accessing the relevancy function.

3.1 Standard

In the standard retrieval set-up for the RAG pipeline, dense retrieval is used for identifying the most relevant chunk to the user query. Let $E(\cdot)$ denote a sentence embedding model. The embedding model has been trained to produce semantically meaningful vector representations of natural language text (see Section 2 for the evolution of sentence transformers). All of the document chunks and the query are embedded into the high-dimensional space such that:

$$\mathbf{c}_i = E(c_i), \forall i \in [1, N] \quad (1)$$

$$\mathbf{q} = E(q) \quad (2)$$

Then the chunk, $c_{\hat{k}}$, is selected such that $\mathbf{c}_{\hat{k}}$ and $\mathbf{q}$ have the shortest cosine distance between all chunk embeddings and the query embedding. The cosine distance between a pair of vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as $\texttt{cos}[\mathbf{a}, \mathbf{b}] = 1 - \mathbf{a}^T\mathbf{b}/|\mathbf{a}||\mathbf{b}|$.

$$[\texttt{chunk}] \quad \hat{k} = \arg\min_k \texttt{cos}[\mathbf{q}, \mathbf{c}_k] \quad (3)$$

One shortcoming of the standard retrieval approach in RAG is that query embeddings are compared against chunk embeddings. However, the semantic embedding representation of a query does not necessarily align with the semantic embedding representation of the chunk that needs to be retrieved. Hence, dense retrieval can lead to the incorrect chunk being retrieved. The following sections describe modifications to the dense retrieval of the chunks to increase the recall rate.

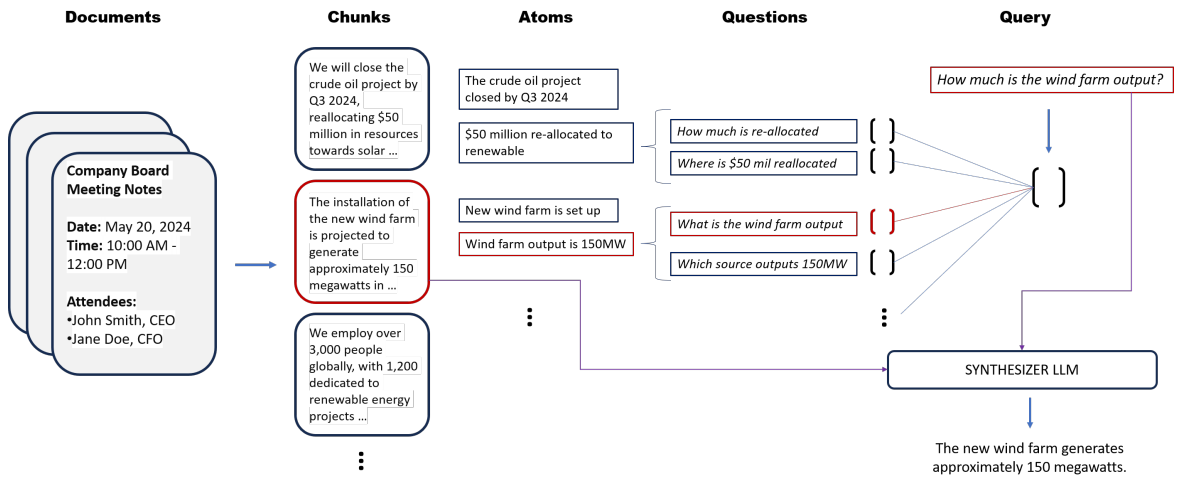


Figure 1: Question-based retrieval using atomic units for enterprise RAG.

3.2 Generation augmented retrieval

As a baseline, the HyDE approach (Gao et al., 2023a) is used as a form of GAR (see Section 2) [1]. The approach requires the query, q to be re-written to q' where q' aims to be a complete hypothesized answer to the query. For example, *What is the capital of India?* is rewritten to *The capital of India is London*. Note, the answer of the query is not important. Instead the form of the answer should hopefully match the nature of the real answer e.g. *London* and *New Delhi* are both places. Now, the standard retrieval approach is applied from Equation 3 with $\mathbf{q}' = E(q')$ as the embedding of the re-written query.

$$[\texttt{hyde}] \quad \hat{k} = \arg\min_{k} \cos[\mathbf{q}', \mathbf{c}_k] \tag{4}$$

The intuition is that by having an answer-like sequence present in the embedded query, there is a greater likelihood of matching with the relevant chunk. Typically, the re-writing process is achieved zero-shot with an instruction finetuned LLM by relying on its parametric answer (at the rewriting stage, hallucinations are not a concern). Henceforth, this approach is referred to as HyDE.

3.3 Atomic

A query is typically searching for a specific piece of information in a chunk. The embedding representation of the chunk can be viewed as an average representation of all the different pieces of information present in the chunk. Often, the pieces of information in the same chunk can be distinct, which can lead to the query embedding being distant from the target chunk embedding with the answer.

Therefore, we propose atomic retrieval. Here, the chunk text is partitioned into a set of atomic statements (referred henceforth as atoms) such that

$$c_k \rightarrow \{a_1^{(k)}, \ldots a_{n_k}^{(k)}\}, \forall k \tag{5}$$

Then, with $\mathbf{a} = E(a)$, the query embedding is compared against the atomic embeddings. The closest atomic embedding is used to identify the corresponding chunk to be retrieved. The expectation is that individual atomic embeddings are more likely to align with a query's embedding in the vector space. The atomic retrieval can be summarized as follows.

$$[\texttt{atom}] \quad \hat{k}, \hat{j} = \arg\min_{k,j} \cos[\mathbf{q}, \mathbf{a}_j^{(k)}] \tag{6}$$

For evaluation, $\hat{k}$ is of interest and $\hat{j}$ is discarded. In this work two forms of atoms are considered:

- **Structured**: The natural structure of the chunk is used to consider each sentence as a separate atom.
- **Unstructured**: An atom generation system (e.g. instruction-finetuned LLM) is asked to generate atomic statements that best capture all the information in the chunk. See Section 4.2 for a description of the specific atom generation system.

Despite atomizing a chunk of text, there is risk of the query not necessarily matching the target

[1] There are several GAR approaches. We find the form of HyDE works best for this dataset from preliminary experiments and hence select it as an appropriate baseline for GAR in RAG.

atom in the embedding space as the atom contains semantic information about the answer while the query does not. Therefore, we propose an extension called atomic questions. For a given atom, a set of synthetic questions are generated that are best answered by the atom given the chunk as the context information. Hence,

$$a_j^{(k)} \rightarrow \{y_1^{(j,k)}, \dots y_{n_{j,k}}^{(j,k)}\}, \forall j, k \tag{7}$$

$$[\texttt{question}] \quad \hat{k}, \hat{j}, \hat{i} = \arg\min_{k,j,i} \cos[\mathbf{q}, \mathbf{y}_i^{(k,j)}] \tag{8}$$

As before, only $\hat{k}$ is of interest for evaluation. Figure 1 summarizes the RAG pipeline with question-based retrieval using atomic units. Effectively, each chunk can be summarized by a set of questions that probe different pieces of information.

4 Experiments

4.1 Data

	SQuAD	BiPaR
# total chunks	$2,067$	375
# total queries	$10,570$	$1,500$
# queries / chunk	$5.1_{\pm 2.3}$	$4.0_{\pm 0.0}$
# words / query	$10.2_{\pm 3.6}$	$7.2_{\pm 2.9}$
# words / chunk	$122.8_{\pm 54.8}$	$181.1_{\pm 52.8}$
# sentences / chunk	$6.6_{\pm 3.1}$	$14.2_{\pm 5.7}$

Table 1: Statistics of datasets.

SQuAD (Rajpurkar et al., 2016) is a popular choice as an extractive reading comprehension dataset consisting of triples of contexts, questions and answer extracts. The contexts are sourced across a wide variety of Wikipedia articles. We re-structure the validation split of the SQuAD dataset for the task of retrieval in RAG as follows. As all questions are answerable (unlike SQuAD 2.0 (Rajpurkar et al., 2018)), we assume that the answer to a given question must be present in its corresponding context passage. We additionally assume that the answer to a specific question is not present in any other context. Therefore, we shuffle all the contexts such that the task requires retrieval of the appropriate context for a given question. Once a particular context is retrieved, it is the role of the synthesizer in the RAG pipeline to generate the required answer. Remaining consistent with the terminology of retrieval in RAG, contexts are viewed as chunks and the questions are termed queries. The collection of chunks are effectively the pre-split texts from a knowledge store, which in this case is Wikipedia.

Table 1 summarizes the statistics of the re-structured SQuAD validation set for assessing the RAG framework. In total there are 2,067 chunks with 10,570 queries, resulting in approximately 5 queries per chunk. The number of sentences within each chunk vary with a single standard deviation of 3.1 about 6.6. As mentioned, in Section 3.3, the sentences of a chunk are treated as structured atoms. Overall, the re-structured dataset allows us to explore whether we can improve the retrieval of chunks for queries over a fixed knowledge store.

Additionally, we consider BiPaR (Jing et al., 2019) for evaluating the RAG framework. BiPaR is a manually annotated dataset of bilingual parallel texts in a novel-like style, created to facilitate monolingual, multilingual, and cross-lingual reading comprehension tasks. We focus on only the English texts over the test split. In a similar vain to SQuAD, the knowledge store is constructed by shuffling the contexts for all queries. Table 1 summarizes the main details. It is particularly useful to consider BiPaR for enterprise RAG as the information content of the context is based on extracts from novels. As the stories are fictional and not factual, the parametric memory of an LLM cannot expect to know the answers to the queries. Therefore, BiPaR mimics the set-up of proprietary knowledge stores for enterprises where retrieval is necessary to identify the relevant information for a query.

4.2 Model details

Task	Prompt
Query re-writing	Please write a full sentence answer to the following question. {query}
Unstructured atom generation	Please breakdown the following paragraph into stand-alone atomic facts. Return each fact on a new line. {chunk}
Question generation	Generate a single closed-answer question using: {chunk} The answer should be present in: {atom}

Table 2: ChatGPT prompts for zero-shot tasks.

For generating the embedding representations, the embedder $E(\cdot)$ is selected as all-mpnet-base-v2 [2] from Huggingface. This embedder is a popular choice for enterprise RAG (the default in LlamaIndex [3] for open-source LLMs) as it performs well on the MTEB (Muennighoff et al., 2023) leaderboard

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3] https://www.llamaindex.ai/

despite its small size of 110M parameters. We additionally present results using the e5-base-v2 [4] embedder, from the E5 model series (Wang et al., 2022), which has topped the MTEB leaderboard for models of the base size.

Instruction finetuned LLMs (Touvron et al., 2023; Jiang et al., 2023) have demonstrated impressive capabilities across a diverse range of tasks from zero-shot prompting. Therefore, For HyDE, the query re-writing process is achieved with zero-shot usage of ChatGPT 3.5 Turbo [5]. Similarly, ChatGPT is used for generating atomic statements from a chunk of text as described in Section 3.3. Finally, we make use of the same model to automatically generate questions on the atoms. Table 2 summarizes the prompts for each of these tasks [6]. The question generation system is applied for a maximum of 15 times on each atom [7] at which the performance plateaus (see Section 5).

4.3 Evaluation

In works for information retrieval, there are a large number of metrics proposed for assessing retrieval capabilities (Arora et al., 2016). Here, we focus on calculating R@K (recall at K). For retrieval for the RAG task, R@K calculates the fraction of queries for which the correct chunk is within the top K chunks when retrieval is performed. We specifically present R@1, R@2 and R@5. Note, R@1 checks for the exact match while R@2 and R@5 are more lenient. We do not consider other retrieval measures that account for the ordering of the documents retrieved as in the scope of this work there is only 1 relevant chunk for each query.

In the RAG pipeline, it is often of interest to return multiple chunks from the retrieval step and leave the job of finding the correct answer amongst the retrieved chunks to the synthesizer. The limit on this approach is the context window of the synthesizer. For example the context window for ChatGPT 3.5 is 16K tokens. Therefore, it is useful to consider moderately high K for R@K such as K=5.

[4] https://huggingface.co/intfloat/e5-base-v2

[5] https://platform.openai.com/docs/models

[6] Manual prompt engineering was performed to identify the appropriate prompts to achieve sensible results.

[7] The code will be made available if accepted.

5 Results

Table 3 presents the recall rates with various zero-shot approaches of the retrieval step using SQuAD and BiPaR with 2 different embedders.

Let's take a look first at the all-mpnet-base-v2 embedder for SQuAD. Operating at the chunk scale, where the raw text is embedded for dense retrieval, the standard RAG achieves a recall of 65.5% with the top 1, which increases to 89.3% when considering the top 5 chunks retrieved. By applying GAR with the HyDE baseline at the chunk scale, we do not observe gains. As discussed in Section 3.3, the text chunk contains several semantic pieces of information while the re-written query remains related to a single semantic piece of information. Hence, it is challenging for the HyDE approach to improve recall at the chunk scale.

By splitting a chunk into structured atoms (sentences), Table 3 further shows the recall by embedding the atomic text or the corresponding synthetic questions generated on those atoms (Equations 6 and 8 respectively). Additionally, the HyDE approach is applied with the atomic embeddings, using the rewritten query instead of the original from Equation 6. Embedding the atomic text instead of the chunk text observes significant gains, reaching 70.2% for R@1 and 90.6% for R@5. As the length of a sentence in a chunk is closer in length to the re-written query, the HyDE approach on the structured atoms further boosts the recall rates. An additional gain is again observed by performing dense retrieval with the set of generated questions, achieving up to 73.8% for R@1.

The final rows of Table 3 for SQuAD with all-mpnet-base-v2 further demonstrates the benefits of using unstructured atoms in place of the structured atoms. A sentence from a chunk contains more granular information than the whole chunk but is not necessarily constrained to one piece of atomic information. Therefore, by re-writing the chunk into a series of independent atoms, dense retrieval between the query and the set of atomic embeddings leads to higher recall rates. As with the structured atoms, the HyDE approach leads to further performance gains with the unstructured atoms. Finally, we observe the best performance across all three recall rates by applying dense retrieval using the generated questions on the atoms. It is clear that higher recall retrieval is possible by matching queries with questions as they can expect to be of the same form rather than attempting to

Dataset	Item		all-mpnet-base-v2 R@1	R@2	R@5	e5-base-v2 R@1	R@2	R@5
SQuAD	Chunk	Text	65.5	78.9	89.3	76.2	87.1	94.4
		HyDE	65.2	77.9	88.9	66.4	79.9	91.1
	Atom-Structured	Text	70.2	81.4	90.6	80.1	**89.3**	**95.1**
		HyDE	71.5	82.3	91.1	73.7	84.6	93.0
		Question	73.8	83.5	91.2	78.1	87.2	93.8
	Atom-Unstructured	Text	72.6	83.9	91.9	80.0	88.3	94.6
		HyDE	73.1	83.7	91.7	73.9	84.4	92.3
		Question	**76.3**	**85.4**	**92.6**	**80.2**	88.6	94.5
BiPaR	Chunk	Text	33.7	43.1	54.7	42.1	52.6	63.7
		HyDE	31.2	41.2	51.7	36.6	47.4	58.9
	Atom-Structured	Text	42.6	52.3	65.4	47.7	57.8	69.5
		HyDE	40.1	50.1	62.1	43.5	52.1	64.9
		Question	**53.8**	**63.4**	**73.3**	**55.9**	**64.8**	**75.3**
	Atom-Unstructured	Text	43.9	54.3	66.9	49.7	58.1	69.1
		HyDE	41.7	52.5	64.6	43.0	51.7	63.7
		Question	53.7	61.9	72.9	55.3	64.1	74.5

Table 3: Retrieval performance for enterprise RAG. All recall rates are represented as percentages.

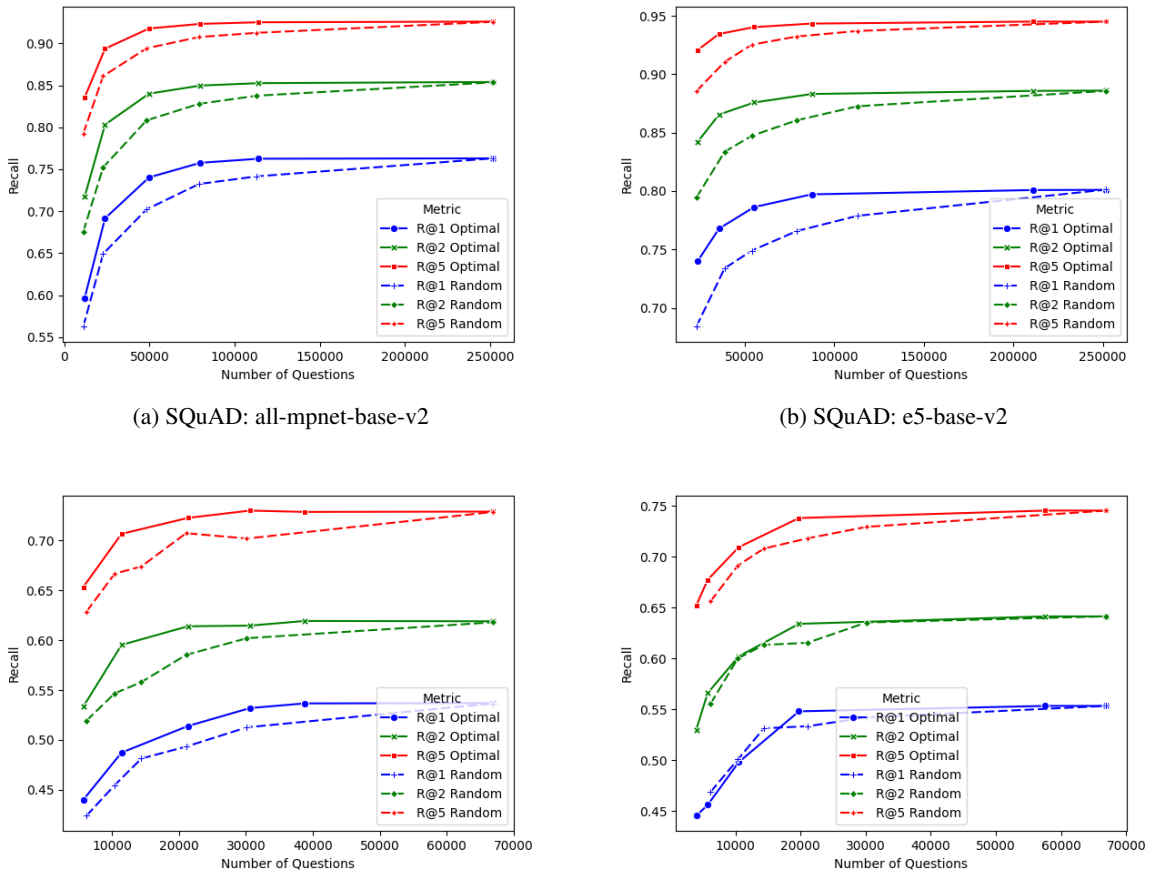
match queries with chunks.

Considering the higher performing embedder e5-base-v2 on SQuAD, the trends are less clear due to higher baseline information. We do observe that for R@1 that atomic question retrieval with unstructured atoms has the best performance, but drops to second and third highest for R@2 and R@5 respectively.

Let's now consider BiPaR from Table 3. Very similar trends are observed for both all-mpnet-base-v2 and e5-base-v2 embedders on this dataset. It is noticeable that HyDE at both the chunk, structured atoms and unstructured atoms struggles to outperform the equivalent text. This deviation in the trend observed in SQuAD is expected as BiPaR is based on fictional stories while SQuAD is based on factual Wikipedia articles. Hence, the hallucinated answers generated by HyDE are unlikely to help with retrieving relevant chunks which do not correspond to the re-written query (see Appendix Section A for more analysis about HyDE). In contrast, for public factual information (as in SQuAD), the hypothesized answer generated by a powerful LLM is more likely to be the correct answer than a hallucination. In contrast, question-based retrieval operating on atoms demonstrates significant performance gains over the baseline for BiPaR. For example, using the e5-base-v2 embedder improves R@1 by approximately 14%.

In general, for the re-formatted SQuAD dataset, Table 1 states there are 2,067 unique chunks. Therefore, the standard retrieval approach for RAG leads to storing 2,067 chunk embeddings. In contrast, the atomic retrieval has substantially larger number of embeddings stored. Using structured atoms, there are 13,630 sentences in total while there are 16,793 unstructured atoms across the corpus. By considering the synthetic question generation strategy described in Section 4.2, question retrieval strategies require $13,630 \times 15$ and $16,793 \times 15$ embeddings to be stored in memory for structured atoms and unstructured atoms respectively. A similar increase in the storage of embeddings apply for the BiPaR dataset. Hence, it is of interest to explore how the number of questions required for each atom can be reduced to remove the redundant ones.

Figure 2 presents how the performance varies with the number of synthetically generated questions on the unstructured atoms. For each recall rate (R@1, R@2 and R@5), two profiles are indicated: 1. a random selection of synthetic questions for the atoms of each chunk; 2. an optimally diverse selection of synthetic questions for the atoms of each chunk. The optimally diverse set of questions is selected as follows. A threshold, τ is selected on the pairwise cosine distance. For the full set of atomic questions generated, the pairwise cosine distances of the question embeddings is calculated for each chunk. If any pairwise cosine distance is below τ, one of the questions is purged. The process if repeated until all questions in the remaining set have pairwise cosine distances of their embeddings above τ. By sweeping τ, the total number of synthetic questions across the corpus changes. One



(a) SQuAD: all-mpnet-base-v2

(b) SQuAD: e5-base-v2

(c) BiPaR: all-mpnet-base-v2

(d) BiPaR: e5-base-v2

Figure 2: Efficient unstructured atomic question retrieval.

can hence expect that a chunk with more information will have a more diverse set of questions.

Figure 2 shows that a significant number of questions are redundant across the SQuAD and BiPaR chunks. By removing more than half of the questions (and hence halving the storage cost), performance can be maintained at the maximal value for each of the recall rates. In the extreme setting, with only 20% of the questions retained, there is only a marginal decrease in recall when using the optimal set. Thus, despite a larger storage cost with atomic question retrieval compared to standard enterprise RAG, the performance boost can be justified with an efficient choice of synthetic questions to retain.

6 Conclusions

RAG systems are a popular framework for enterprises for automated querying over company documents. However, poor recall of relevant chunks with dense retrieval causes errors to propagate to the synthesizer LLM. Previous works have focused on extensions involving generation augmented retrieval where the query is re-written at inference time to improve recall. Conversely, we explore adaptations to the storage of the chunks. The retrieval step for RAG can be refined in a zero-shot manner by 1) atomizing the chunks and 2) generating questions on the atoms. Significant improvements are observed on the BiPaR and SQuAD datasets with this approach as partitioning a chunk into atomic pieces of information allows dense retrieval with the query to be more effective. Moreover, operating in the question space, the query embedding aligns better with the synthetic questions of the target chunk. We further demonstrate that the storage cost of a large number of synthetic question embeddings can be dramatically reduced by only storing a diverse set of questions for each chunk. Question-based retrieval using atomic units will enable the deployment of higher performing enterprise RAG systems without relying on any additional training.

7 Limitations

In this work, we have made several assumptions which do not necessarily hold in real enterprises. Our work focuses on only closed queries where a single atom contains the answer. It would be interesting to extend the approach to handle multi-hop situations by generating synthetic questions on pairs or collections of atoms. Additionally, we have focused the presentation of our results on SQuAD and BiPaR. It will be useful to consider additional standard information retrieval benchmarks such as the BEIR datasets (Thakur et al., 2021).

8 Ethics statement

There are no ethical concerns with this work.

References

Daman Arora, Anush Kini, Sayak Ray Chowdhury, Nagarajan Natarajan, Gaurav Sinha, and Amit Sharma. 2023. Gar-meets-rag paradigm for zero-shot information retrieval. *arXiv preprint arXiv:2310.20158*.

Monika Arora, Uma Kanjilal, and Dinesh Varshney. 2016. Evaluation of information retrieval: precision and recall. *International Journal of Indian Culture and Business Management*, 12(2):224–236.

Silvia Badini, Stefano Regondi, Emanuele Frontoni, and Raffaele Pugliese. 2023. Assessing the capabilities of chatgpt to improve additive manufacturing troubleshooting. *Advanced Industrial and Engineering Polymer Research*, 6(3):278–287.

Aram Bahrini, Mohammadsadra Khamoshifar, Hossein Abbasimehr, Robert J Riggs, Maryam Esmaeili, Rastin Mastali Majdabadkohne, and Morteza Pasehvar. 2023. Chatgpt: Applications, opportunities, and threats. In *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 274–279. IEEE.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Davide Castelvecchi. 2023. Open-source ai chatbots are booming—what does this mean for researchers?

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Scott Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4207–4218.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. 2023. Knowledge refinement via interaction between search engines and large language models. *arXiv preprint arXiv:2305.07402*.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Kailash A Hambarde and Hugo Proenca. 2023. Information retrieval: recent advances and beyond. *IEEE Access*.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

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

Yimin Jing, Deyi Xiong, and Zhen Yan. 2019. Bipar: A bilingual parallel dataset for multilingual and cross-lingual reading comprehension on novels. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2452–2462.

Jaewoong Kim and Moohong Min. 2024. From rag to qa-rag: Integrating generative ai for pharmaceutical regulatory compliance process. *arXiv preprint arXiv:2402.01717*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation

for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Sara Rosenthal, Avirup Sil, Radu Florian, and Salim Roukos. 2024. Clapnq: Cohesive long-form answers from passages in natural questions for rag systems. *arXiv preprint arXiv:2404.02103*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

Tao Shen, Guodong Long, Xiubo Geng, Chongyang Tao, Tianyi Zhou, and Daxin Jiang. 2023. Large language models are strong zero-shot retriever. *arXiv preprint arXiv:2304.14233*.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. 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.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, and Suranga Nanayakkara. 2021. Fine-tune the entire rag architecture (including dpr retriever) for question-answering. *arXiv preprint arXiv:2106.11517*.

EuiYul Song, Sangryul Kim, Haeju Lee, Joonkee Kim, and James Thorne. 2024. Re3val: Reinforced and reranked generative retrieval. *arXiv preprint arXiv:2401.16979*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023a. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

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

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023c. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

Zhipeng Xu, Zhenghao Liu, Yibin Liu, Chenyan Xiong, Yukun Yan, Shuo Wang, Shi Yu, Zhiyuan Liu, and Ge Yu. 2024a. Activerag: Revealing the treasures of knowledge via active learning. *arXiv preprint arXiv:2402.13547*.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024b. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5065–5075.

Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, et al. 2023a. Kola: Carefully benchmarking world knowledge of large language models. *arXiv preprint arXiv:2306.09296*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023b. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Model	nDCG@1	nDCG@3	nDCG@5	nDCG@10	R@10
BM25	18	30	35	40	67
all-MiniLM-L6-v2	29	43	48	53	79
BGE-base	37	54	59	61	85
E5-base-v2	41	57	61	64	87
E5-base-v2 (ours)	36	51	54	58	82
+ HyDE	42	57	61	63	85
all-mpnet-base-v2	37	51	56	61	87
+ HyDE	39	56	60	63	88

Table 4: Baselines for ClapNQ with HyDE.

A Drawback of HyDE

In the main paper, we observed that HyDE performs well for SQuAD but is less impressive for BiPaR. This Section aims to revisit how HyDE operates to explain the difference. Qualitatively, HyDE uses the parametric memory of an LLM to re-write the query as a complete sentence that answers the query. The re-written query is then used to retrieve the relevant chunks. The HyDE paper emphasizes that it doesn't matter if the answer is hallucinated as the form of the hypothetical answer can expect to be aligned with the chunk containing the correct answer.

However, it is clear that HyDE struggles on BiPaR while working well on SQuAD. We suspect the reason for this discrepancy is that SQuAD is based on publicly known factual information from Wikipedia while BiPaR is based on fictional stories. Therefore, when HyDE is applied on SQuAD, the hypothesized answer often is simply the correct answer itself, leading to an artificial boost in the retrieval performance. The correct answer is generated typically by the parametric memory of a powerful LLM used for the query re-writing. In contrast, as the answers to the queries in BiPaR are not within the scope of general knowledge, the hypothesized answer from HyDE does not help in boosting the retrieval performance.

In order to investigate the dependence of HyDE on factual information for improving retrieval performance, we do additional analysis. We select CLAPNQ (Rosenthal et al., 2024) as a recently curated RAG dataset where the knowledge store is based on publicly available information (like SQuAD). Additionally, CLAPNQ has been exclusively designed for long-form answers. Therefore, we expect HyDE to demonstrate significant performance gains on this dataset as the hypothesized answer is likely to be the correct answer with high overlap with the target chunk due to the length of the answer. We show our results as follows in Table 4. The top 5 rows are quoted directly from Rosenthal et al. (2024). As well as recall, we report nDCG (Järvelin and Kekäläinen, 2002) here as a standard retrieval metric used in Rosenthal et al. (2024) where the order of the retrieved chunks is accounted for in calculating the performance. It is clear for both of our implementations that HyDE demonstrates retrieval performance gains on this challenging RAG dataset.

B Licenses

SQuAD is shared under the attribution-sharealike 4.0 international (CC BY-SA 4.0) license. BiPaR is shared under the attribution-noncommercial 4.0 international (CC BY-NC 4.0) license. CLAPNQ is shared under the Apache-2.0 license.