Rank-Select-Compose: A Novel Retrieval Paradigm for Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved impressive results across various NLP tasks but remain constrained by hallucinations and the static nature of their pretrained knowledge. Retrieval-augmented generation (RAG) mitigates these limitations by retrieving external text to enhance response accuracy. However, traditional retrieval methods such as Dense Passage Retrieval (DPR) retrieve independent text chunks without considering their contextual relationships, limiting their effectiveness. In this work, we propose a novel **rank-select-compose** approach that improves retrieval quality by leveraging both text ranking and contextual selection. Our method first **ranks** text chunks using DPR and constructs a chunk-based graph that captures the relationships between chunks. The model then **selects** the most relevant chunks based on the current context. Finally, the model **composes** them into a context using chain-of-thought (CoT) and evaluates whether the retrieved information suffices to answer the query or requires further expansion. We empirically validate our rank-select-compose approach on multi-hop question-answering tasks, demonstrating significant improvements in retrieval efficiency and response quality, while reducing the overall context size. This hybrid retrieval strategy effectively balances the relevance of retrieved chunks and the conciseness of the retrieved context, advancing the state of RAG-based systems.

1 Introduction

Large language models (LLMs) are powerful tools that have demonstrated remarkable performance on a variety of NLP tasks such as commonsense reasoning, language understanding, and coding tasks (Lyu et al., 2019; Achiam et al., 2023; Team et al., 2024; Lozhkov et al., 2024), but they are restricted to the knowledge they are pretrained on and suffer from hallucination which limits their usefulness (Xu et al., 2024b). Retrieval-augmented Generation (RAG) augments LMs knowledge by retrieving information from external knowledge resources to mitigate hallucinations and enhance response quality (Lewis et al., 2020; Gao et al., 2023; Kasai et al., 2024). RAG works by first retrieving relevant information from a database given a query. The retrieved information is then incorporated into the LLM's input, allowing it to produce more accurate and contextually appropriate text.

Several retrieval methods are used for RAG, most of which rely on sparse word-based retrieval like BM25 or TF-IDF, on dense passage retrieval (DPR) that retrieves text chunks based on their embedding similarity to a given query, or a hybrid approach that combines the benefits of semantic and keyword search (Fan et al., 2024). The commonality of all these approaches is that they retrieve a set of top-k independent text chunks. That is, the retrieval of one chunk does not use any information about other chunks. However, **we hypothesize that utilizing the contextual relationship between different chunks can improve retrieval quality**. Important chunks that appear irrelevant to the query may be related to relevant chunks. Particularly in the case of long documents such as books, each text chunk's meaning is contextual and needs to be interpreted in the context of other chunks.

Further, the length of the retrieved chunks also plays a crucial role in the process of response generation. This is due to 1) the LLMs' limitation on maximum context length (Yu et al., 2024) and 2) its sensitivity to the placement of the answer within the context (Liu et al., 2024). Therefore, the retrieved knowledge must not only be **relevant**, but also **concise**. Conciseness is important due to the limited context windows of LLMs. Even if the entire document fits within the context window, the long length can still negatively impact performance if the answer is in the middle of the context (Liu et al., 2024) or due to the presence of irrelevant

information (Shi et al., 2023).

Dense Passage Retrieval (DPR) (Lyu et al., 2019) is one of the most extensively studied RAG techniques in the literature. It *ranks* text chunks in descending order of relevancy. While it effectively identifies the most relevant chunks for a given query, it overlooks the contextual relationships between chunks. In contrast, recent graph-based retrieval methods like Edge et al. (2024) allow models to capture complex relationships between chunks, but they face challenges in extracting relations and in determining which nodes to start traversal at.

Previous approaches to creating a graph representation of text have focused on entities as nodes and relations between entities as the edges (Edge et al., 2024; Mavromatis and Karypis, 2024). However, this approach relies on having a large corpus of text with redundancy in the relations. Even with state-of-the-art relation extractors, there is a trade-off between precision and recall meaning that either many low-quality relations will be extracted or only a few high-quality relations. In the former case, the created knowledge graph is too noisy to use, and in the latter case, the knowledge graph is too sparse.

Inspired by Huang et al. (2022), we instead employ a chunk-based graph representation that captures relationships between chunks without relying on traditional relation extraction methods. In this work, we propose a hybrid approach that addresses the limitations of both methods: DPR-retrieved **rank**ed chunks are used to identify the root nodes of a chunk-based graph, enabling selective retrieval of more contextually relevant chunks for a query. Building on the approach of Shao et al. (2023), we enable the LLM to **select** its context at each step of the retrieval process. The **Selector** expands the context by adding an additional chunk, while the **Composer** determines whether the current context is sufficient to answer the query or if more information is needed. To optimize this process, the Selector is prompted to provide multiple chunks, from which the most suitable one for expanding the context is selected. We also utilize chain-of-thought (CoT), making the Selector explain its rationale for choosing how to expand the context (Wei et al., 2022) and **compose** the final context for the Composer.

We empirically demonstrate the effectiveness of **rank-select-compose** on several multi-hop question-answering datasets with consistent quality improvements. Through ablation studies, we show that the proposed approach reduces the number of retrieved chunks while improving the retriever quality.

We summarize our findings as follows:

- Combining graph-based document representations with DPR results in greater performance than using either separately
- Allowing LLMs to *select* the context results in shorter contexts while improving performance
- Utilizing LLM explanations results in shorter contexts without sacrificing performance

Ablation studies demonstrate the benefits of our graph representation of text and that the improvements hold across different datasets, different embeddings, and different language models.

2 Related Work

RAG has been a popular method to deal with the hallucination and restriction to training data of LLMs. Although long context LLMs are gaining in popularity, RAG is still useful in cases where the context is sufficiently lengthy due to negative effects from irrelevant context and the position of the answer (Yu et al., 2024).

Previous RAG methods have largely relied on sparse or dense passage retrieval (Fan et al., 2024). However, RAG performance is heavily dependent on the quality of the retriever, giving incentives to improve the set of retrieved text chunks. One method for doing so is iterative retrieval. Li et al. (2023) iteratively uses a query to retrieve chunks, asks the LLM what information is missing from the retrieved context, and then generates a new query to gather the missing information. This process is repeated until the original query can be answered. Iterative retrieval is often combined with chain-of-thought (CoT) prompting to improve retrieval based on previous retrieval rounds. Trivedi et al. (2022a) iteratively retrieves chunks using a query, generates a sentence using the retrieved chunks through CoT, and then retrieves a new set of chunks using the last generated sentence. The iteration process ends either after a predefined number of steps or if the generated sentence contains "answer is". Shao et al. (2023) iteratively retrieves chunks using a query, generates a paragraph using the retrieved chunks through CoT, and then concatenates the generated paragraph to the query so that the retrieved chunks in the next iteration are more relevant.

Although superficially our method also utilizes iteration and CoT, Rank-Select-Compose is a retrieval method and so is compatible with these previous methods. Rather than using DPR to retrieve a new set of chunks at each iteration, it is possible to use Rank-Select-Compose instead.

Although DPR is the most commonly used retriever, recent works are beginning to use graphs for RAG. Edge et al. (2024) use a graph-based representation of text with entities as nodes and relations as edges, then create summaries of graph communities to answer questions. Mavromatis and Karypis (2024) utilize a GNN over a knowledge graph to retrieve answer candidates and the shortest paths between the question entities and the answer candidates. These are then used as input for an LLM-based generator.

Compared to Rank-Select-Compose, these methods utilize an entity and relation-centric graph, either using a pre-existing KG or creating one on the fly. However, using a pre-existing KG limits its usefulness for new and emerging entities and relations and creating one on-the-fly requires high-quality relation extraction which is still an open problem, requiring one to sacrifice either recall or precision. By adopting an entity- and chunk-centric approach, we are able to capture the relationship between entities and chunks in a much simpler but still useful way.

There are also several methods that seek to improve the retrieved contexts before passing them to the generator. For instance, there are multiple works that attempt to compress or summarize the context, either to remove excess information or to fit long contexts into the context window (Xu et al., 2024a). Other works rerank the retrieval results before generating an answer (Glass et al., 2022). These methods aim to reduce the context length and rearrange it post-facto. In contrast, Rank-Compose-Select reduces the context length and composes it during the retrieval process itself. These post-facto methods are also compatible and can be used to further shorten the context.

3 Rank-Select-Compose

In this section, we describe the rank-select-compose approach in detail. An overview of our approach is depicted in Figure 1. The first step retrieves a ranked list of chunks. Then, we select the most appropriate chunk for answering the query. Finally, we compose it into a context and determine whether it can answer the question or not. If not, then we repeat the selection and composition process until the context can.

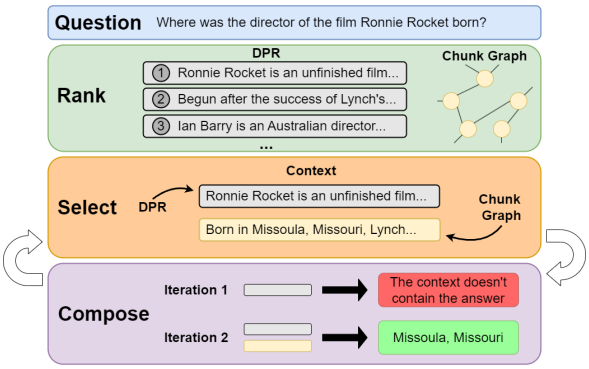

Figure 1: Overview of the basic operation of our model. Given a query, we first use Dense Passage Retrieval (DPR) to retrieve an initial ranked set of chunks and create a graph where chunks are nodes and edges are formed from the collocation of entities. We then perform multiple iterations of Select and Compose, where the context is expanded by asking the language model to select the best chunk from among DPR and the neighbors of the context chunks in the graph. This process is repeated until either the Composer generates an answer or a maximum number of iterations is reached.

3.1 Graph-Based Document Representation

We begin by outlining the construction of a chunk graph to facilitate the efficient selection of relevant text for a given query. The first step in building any knowledge graph (KG) is defining the graph's topology, which involves identifying the nodes and the edges between them. Typically, KGs consist of entities as nodes, with relationships between these entities represented as edges.

However, most corpora lack explicitly labeled relationships, requiring the use of extraction techniques to generate a knowledge graph from the text. In our case, attempts to extract relation tuples proved inadequate: either the recall was too low, failing to extract enough relationships per sentence to form a fully connected graph, or the precision was too low, resulting in an overabundance of incorrect relations.

Generally, entity extraction is simpler and yields higher recall and precision compared to relation extraction. Therefore, in constructing a graph representation of text, we rely on entity collocation to infer relationships, similar to Huang et al. (2022). Instead of explicitly extracting relations, we use the sentences where entities co-occur to indicate potential connections as depicted in Figure 2. This

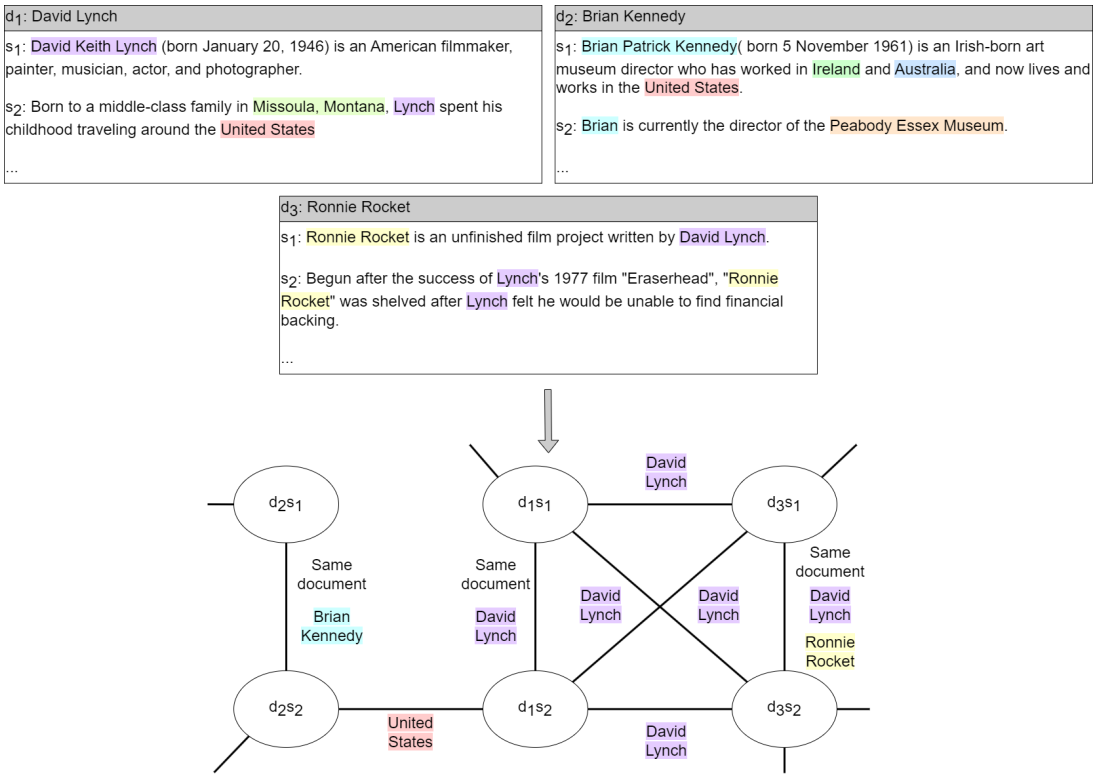

Figure 2: Overview of chunk graph creation. Given a set of documents, we first perform chunking and then entity extraction. Using the chunks, we construct a graph where nodes are chunks and edges exist between chunks that come from the same document or share an entity.

approach allows us to build a high-recall graph while avoiding the introduction of erroneous relationships between entities. Ultimately, we create a chunk-based graph, where the nodes represent chunks of text and an edge connects two nodes if they either share at least one entity or originate from the same document.

We also note that the granularity of chunks significantly impacts graph completeness. When large chunks, such as entire documents, are used as nodes, the graph loses specificity and suffers from weak semantic connections. On the other hand, using smaller chunks, like individual sentences, can introduce ambiguities in relationships, often leading to a sparser graph. These ambiguities arise primarily from the presence of pronouns, as we rely on shared entities between chunks to establish relationships. To address this, we apply coreference resolution to replace pronouns in the documents. In Section 5.1.3, we demonstrate that replacing pronouns with their referents helps preserve context and creates a denser graph, resulting in improved performance when using the graph for retrieval.

Another challenge is determining the starting point for graph traversal. In knowledge graphs where entities are used as nodes and the relationships between them are edges, a typical approach is to extract entities from the question and use those as the initial nodes for traversal (Zhang et al., 2022). From there, the context can be expanded further. However, this method may be less effective if the question contains few or no entities, or if other parts of the question, such as the predicate, are more important than the entity itself. Therefore, in this case, the starting node is defined as the highest-ranked document according to the Dense Passage Retrieval (DPR) method.

3.2 Selection and Composition

Up to this point, we have represented all our documents using a graph structure and generated a ranked list of highly relevant documents through DPR. In this section, we will focus on the process of content selection and composition for the LLM to produce the final answer.

One key limitation of current selection strategies is that they rely on fixed criteria for semantic similarity to select a predetermined number of top-k relevant documents for each query. This approach does not account for varying levels of query diffi-

culty, which can affect the quality and relevance of the selected documents. Further (Liu et al., 2024) and Shi et al. (2023) demonstrated that both the length of the context and the position of the answer within the context can influence the generation accuracy of LLMs.

Instead of manually selecting the context for the LLM, we allow the LLM to choose its own context. During the selection process, the **Selector** expands the context incrementally by adding one chunk at a time. To select context, the Selector is presented with several chunks and decides which one is most suitable for expanding the context. The available chunks for selection are the top k-ranked chunks from DPR, as well as chunks within a 1-hop distance from the currently retrieved chunks in the chunk-based graph. Figure 1 shows an example of selection, where different chunks are retrieved from different sources on different iterations. For each chunk, information about its relationship to the existing context is provided such as sharing an entity or being from the same document as a chunk already included. Figure 2 shows examples of different edges between chunks. The **Composer** then determines whether the current context is sufficient to answer the question or if further selection is needed.

This process of Selection and Composition repeats until the Composer either opts to answer the question using the current context or until a predetermined number of chunks have been selected. In addition to selecting the next chunk, the Selector is also tasked with generating an explanation for why that particular chunk was chosen, following a method similar to Chain of Thought (CoT) reasoning (Wei et al., 2022). These explanations along with the composed content are included in the context and passed to the generator to generate a response. In Section 5.1.2, we demonstrate that the inclusion of explanations helps the language model better reason about which contexts are necessary to answer the question, resulting in a shorter retrieved context.

4 Experiments

4.1 Datasets and Evaluation Metrics

To demonstrate the power of graph representations of text and the incremental selection of the appropriate context for answering the question, we perform experiments using the multi-hop QA datasets 2WikiMultiHopQA (Ho et al., 2020) (designed to be an improvement on the HotPotQA dataset (Yang et al., 2018)) and MuSiQue (Trivedi et al., 2022b). 2WikiMultiHopQA and MuSiQue consist of 2-hop questions derived from Wikipedia articles. For each question, a set of documents is provided. The labels for each question consist of both a gold answer and which sentences from the documents support the answer. For each dataset, we limited ourselves to the first 750 questions in the development set.

Comparatively, MuSiQue is more difficult than 2WikiMultiHopQA due to its longer documents, with an average of 2160 tokens among all the provided documents compared to 2WikiMultiHopQA's 883 tokens. Additionally, roughly one-third of the questions in 2WikiMultiHopQA are multiple-choice between two choices, while all questions in MuSiQue are open-ended.

We evaluate performance using the metrics provided by Ho et al. (2020). These metrics include an exact match, recall, precision, and F1 score for how much the tokens of the predictions match the tokens in the gold answers.

4.2 Baselines

DPR This approach follows the standard Dense Passage Retrieval (DPR) technique, where a hybrid search method (combining both sparse and dense retrieval) with a reranker is employed to retrieve the top-k documents for a given query. In all our experiments, we set k=10.

Oracle To establish the upper bound on quality and eliminate the generator's influence on results, we provide all the gold documents that answer the query as the context for the generator.

RSC Rank-Select-Compose: Our approach as described in Section 3. The Selector selects a chunk either from the top 10 results of the DPR or from the neighbors of previously selected chunks in the chunk graph. The Composer then determines whether the current context is sufficient to answer the question or if another round of selection is needed.

RSC-DPR Rank-Select-Compose with DPR only. Includes use of the Selector for selecting how to expand the context but only from among the chunks ranked in the top-k by DPR. The Composer then decides whether the context is sufficient to answer the question or if another round of selection is necessary.

RSC-Graph Rank-Select-Compose with the chunk graph only. The starting node in the chunk graph is whichever chunk was ranked highest by DPR. The

Method	DPR Embed	2WikiMultiHopQA					MuSiQue				
		EM ↑	P ↑	R ↑	F1 ↑	AConL ↓	EM ↑	P ↑	R ↑	F1 ↑	AConL ↓
Oracle		60.00	66.27	68.41	66.47	86.19	35.47	47.29	48.12	46.44	225.79
DPR	jina	37.20	43.70	44.29	43.46	363.55	12.93	22.91	21.32	21.04	416.23
RSC-DPR	jina	30.27	36.88	37.00	36.50	161.09	9.60	18.44	16.22	16.54	168.81
RSC-Graph	jina	36.53	44.00	45.59	44.12	**104.05**	**17.07**	**26.98**	**25.40**	**25.26**	**126.75**
RSC	jina	**38.53**	**46.75**	**47.94**	**46.57**	112.49	16.13	25.88	24.39	24.22	132.68
DPR	instruct	**39.20**	**46.02**	**46.89**	**45.84**	339.53	14.53	23.86	22.29	22.24	378.29
RSC-DPR	instruct	31.47	38.95	39.37	38.54	152.70	10.67	19.33	17.22	17.41	152.29
RSC-Graph	instruct	35.87	43.37	44.69	43.31	**103.80**	16.40	27.18	25.55	25.30	**117.86**
RSC	instruct	36.40	44.43	46.09	44.46	107.55	**16.80**	**27.71**	**25.60**	**25.60**	123.73

Table 1: Evaluation of different methods on the first 750 examples of the development sets of multi-hop QA datasets 2WikiMultiHopQA and MuSiQue using Llama3. EM is the exact match accuracy. AConL is the average context length in tokens used to generate the answer. EM, P, R, and F1 Score are computed using the evaluation script provided by Ho et al. (2020). Here **BOLD** represents the best score and the underline represents the second best

Selector selects how to expand the context but only from among the chunks within 1-hop of either the starting node or a node already selected to be part of the context. The Composer then decides whether the context is sufficient to answer the question or if another round of selection is necessary.

We allow each RSC method to run for a maximum of 10 iterations.

4.3 Experimental Settings

- LLM: We use Meta-Llama-3-8b-Instruct (Llama3) (AI@Meta, 2024) as our primary large language model.
- Hardware: We run all experiments on a Quadro 5000 RTX.
- Entity Extractor:We use spaCy's EntityRecognizer (M. Honnibal and Boyd, 2020).
- Coref Resolution: We use AllenNLP's coref-spanbert-large-2021.03.10 model (Gardner et al., 2017).
- Chunk Granularity: Single sentence.
- DPR Settings: For DPR, we use a hybrid retriever of a BM25 retriever and a semantic retriever with equal weight. For semantics, we experimented with different sentence embedding models such as jina-embeddings-v2-base-en (jina) (Günther et al., 2023) and instructor-large (instruct) (Su et al., 2022).

5 Results

We summarize our findings in Table 1. Across both datasets, RSC consistently ranks either first or second compared to other RSC variants and DPR. In nearly all cases, RSC outperforms DPR, regardless of the dataset or the DPR embeddings used. The only exception is with 2WikiMultiHopQA using instruct embeddings, where DPR shows slight gains over RSC. We hypothesize that these gains for DPR may be due to the simplicity of the 2WikiMultiHopQA dataset combined with more powerful embeddings. However, as dataset complexity increases, such as with the MuSiQue dataset, RSC surpasses DPR by 3.4 absolute F1 points.

A key observation from these experiments is the average content length of the context for generation model. For the 2WikiMultiHopQA dataset, RSC achieved a 3x reduction in average content length compared to DPR, regardless of the embeddings. This reduction was even more pronounced with the more complex MuSiQue dataset, where RSC produced over a 4x reduction in average content length. The substantial reduction in average content length achieved by RSC, particularly for more challenging datasets like MuSiQue, **highlights its efficiency in retrieving concise yet informative contexts**. These findings suggest that **RSC offers a robust and scalable solution for multi-hop question-answering tasks**, particularly when handling large or complex datasets.

Concise but Relevant Content: As shown in Table 1, RSC outperforms DPR while using a shorter average content length. According to Shi et al. (2023), reducing context length can enhance answer generation by eliminating irrelevant information. To investigate whether RSC's advantage over DPR is solely due to selecting a more concise con-

Method	2WikiMultiHopQA						MuSiQue					
	EM ↑	P ↑	R ↑	F1 ↑	ANCh ↓	AConL ↓	EM ↑	P ↑	R ↑	F1 ↑	ANCh ↓	AConL ↓
Limited-k DPR	36.27	42.39	43.67	42.32	3.23	119.30	7.33	13.82	12.44	12.39	3.47	141.73
RSC	**38.53**	**46.75**	**47.94**	**46.57**	3.23	112.49	**16.13**	**25.88**	**24.39**	**24.22**	3.47	132.68

Table 2: **Context size Vs. Retrieval quality**. Here, ANCh represents the average number of retrieved chunks and AConL the average context length. Limited-k DPR is DPR where the number of top-ranked chunks retrieved by DPR are limited to be the same number of chunks retrieved by RSC. All numbers are obtained using jina embeddings.

Method	2WikiMultiHopQA						MuSiQue					
	EM ↑	P ↑	R ↑	F1 ↑	ANCh ↓	AConL ↓	EM ↑	P ↑	R ↑	F1 ↑	ANCh ↓	AConL ↓
RSC	38.53	**46.75**	**47.94**	**46.57**	**3.23**	**112.49**	**16.13**	**25.88**	**24.39**	**24.22**	**3.47**	**132.68**
RSC w/o Explanation	**38.80**	46.61	47.34	46.33	3.85	133.70	15.07	25.23	23.45	23.35	3.89	148.39

Table 3: Ablation study demonstrating the effect of allowing the **Selector** to explain why it chose a particular sentence to expand the context. All numbers are obtained using jina embeddings.

text, we conducted an experiment where DPR was restricted to retrieve the same number of chunks as RSC for each example. Specifically, if RSC selects k chunks before answering, we limited DPR to use only the top-k ranked chunks for its response.

We present this analysis in Table 2, where ANCh refers to the average number of retrieved chunks and AConL to the average context length in tokens. As seen, both methods result in similar context lengths across the datasets. However, RSC achieves an absolute gain of 4 F1 points on the 2WikiMultiHopQA dataset and a remarkable 12-point improvement on the more complex MuSiQue dataset. These results demonstrate that **RSC not only selects concise contexts but also ranks more relevant information higher than DPR**, leading to better performance.

5.1 Ablation Studies

In this section, we will discuss the impact of each design choice.

5.1.1 Impact of Embeddings

We explored whether the performance gains of RSC are tied to the specific embeddings used. As shown in Table 1, RSC consistently achieves similar or better performance compared to DPR, regardless of whether jina or instruct embeddings are used while maintaining a much shorter context length. Furthermore, the trend of RSC-Graph outperforming RSC-DPR persists across different embeddings, indicating that the **improvements offered by RSC are robust and not dependent on the choice of embeddings**.

5.1.2 Impact of Explanation

Our Selector works by selecting the context to add to the context. In the process, it also generates an explanation for why that chunk is necessary to answer the question. In Table 3 we show the impact of explanations on the RSC quality. First, **explanations do result in better context selection**. For both the datasets we observe a slight but consistent gain with explanations. Second, explanation seems to help the Composer understand the context better and therefore answer the question with a shorter context.

5.1.3 Impact of Coreference Resolution

As discussed in Section 3, to reduce the sparsity of chunk graphs, we replace pronouns with their corresponding entities using coreference resolution. To assess the impact of coreference resolution on performance, we reran the experiments from Table 1 without applying coreference resolution, using both the 2WikiMultiHopQA and MuSiQue datasets.

The results in Table 4 show that **removing coreference resolution negatively affects the performance of both RSC and RSC-Graph**, while slightly improving DPR's performance. The decreased performance of DPR may be due to error propagation from coreference resolution.

5.1.4 Impact of LLM

We run some additional experiments using SOLAR-10.7B-Instruct-v1.0 (Solar) (Kim et al., 2023, 2024) to test whether our results hold across different

Method	DPR Embed	2WikiMultiHopQA w/o Coref					MuSiQue w/o Coref				
		EM ↑	P ↑	R ↑	F1 ↑	AConL ↓	EM ↑	P ↑	R ↑	F1 ↑	AConL ↓
DPR	jina	**38.93**	44.82	**45.46**	**44.74**	321.52	13.87	22.95	21.32	21.22	368.55
RSC-DPR	jina	31.47	37.67	37.45	37.17	130.83	8.40	16.52	14.55	14.83	140.85
RSC-Graph	jina	35.73	44.07	44.92	43.72	**88.99**	14.53	24.61	22.96	22.92	**116.31**
RSC	jina	37.07	**44.88**	45.17	44.36	99.28	**15.20**	**25.48**	**23.95**	**23.80**	116.50
DPR	instruct	**38.27**	**44.74**	**45.56**	**44.75**	301.34	14.53	23.91	22.18	22.20	333.91
RSC-DPR	instruct	32.00	38.93	38.82	38.43	121.90	8.80	17.12	14.61	15.18	127.71
RSC-Graph	instruct	35.07	42.73	43.83	42.51	**86.04**	**15.73**	**25.65**	**23.62**	**23.80**	**106.77**
RSC	instruct	37.20	44.57	45.15	44.20	94.88	13.33	23.08	21.39	21.37	109.31

Table 4: Evaluation results on the 2WikiMultiHopQA and MuSiQue datasets when no coreference resolution is applied.

Method	2WikiMultiHopQA				
	EM	P	R	F1	AConL
DPR	31.60	35.74	39.39	36.14	417.80
RSC-DPR	37.07	44.12	45.57	44.02	138.97
RSC-Graph	38.00	45.51	48.19	45.81	**97.77**
RSC	**43.33**	**49.77**	**52.17**	**49.99**	109.81

Table 5: Evaluation results on the 2WikiMultiHopQA dataset using the Solar LLM and Jina embeddings.

LLMs. As shown in table 5, RSC achieves a higher performance than DPR with a lower context length on the 2WikiMultiHopQA dataset. Thus, **the gains from RSC are independent of the chosen LLM**.

5.2 Analysis

We perform additional analysis of cases where DPR and RSC fail in 2WikiMultiHopQA to see if there are patterns to the failure cases. We find that from a sample of 25 cases, in 5 of those cases DPR was unable to answer the question despite having the gold sentences in its context. In these cases the Oracle was able to answer the question with just the gold sentences, indicating that the issue may lie with the context being too long or being in the middle of the context as described by (Liu et al., 2024) and Shi et al. (2023).

We also observe that the questions answered correctly and incorrectly by RSC, RSC-DPR, and RSC-Graph in 2WikiMultiHopQA diverge. In 26.8% of questions, when RSC-DPR answers correctly or incorrectly RSC-Graph does the opposite. Between RSC-DPR and RSC, they differ 31.5% of the time. Between RSC-Graph and RSC, they differ 32.4% of the time. Additionally, there are some questions that RSC answers incorrectly that RSC-DPR and RSC-Graph answer correct. This indicates that despite RSC being able to choose from the chunks available to RSC-DPR and RSC-Graph, its retrieval strategy differs from both of them.

One potential concern is data contamination, whereby the language model correctly answers questions because the dataset is within the training data. To test for this, we asked the language model to answer the questions in 2WikiMultiHopQA and MuSiQue without any context at all. For 2WikiMultiHopQA, we found that Llama3 achieves an F1 score of 26.49. This may be because roughly one-third of the questions in 2WikiMultiHopQA are multiple-choice questions between two choices. Additionally, Solar achieves comparable performance with all methods but only achieves an F1 score of 4.03, mostly because unlike Llama3 which guesses on almost all questions, Solar mostly answers "I don't know". On MuSiQue, both models perform much worse, with Llama3 achieving an F1 score of 0.45 and Solar achieving 0.46. Because of these results, we believe that data contamination is not an issue for these datasets.

6 Conclusion

In this paper, we introduce **Rank-Select-Compose**, a novel retrieval paradigm that allows the language model to select how to expand the context using both DPR and a graph representation of the text. We demonstrate greater performance than DPR on the multi-hop QA datasets 2WikiMultiHopQA and MuSiQue while retrieving a shorter context. As a retrieval method, this work is compatible with many other RAG methods that seek to improve the context post-retrieval, such as those that perform iterative retrieval or context compression. We hope this work inspires future research on how to better structure and retrieve text for RAG.

Limitations

In this work, we used relatively small, open-source models. A model with more parameters may have led to improved performance. Further, we do not perform any particular fine-tuning of the language model or any in-depth prompt engineering. There may be greater gains if one is willing to fine-tune for the tasks of selecting chunks for context expansion or for generating explanations for why a particular chunk was chosen.

References

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

AI@Meta. 2024. Llama 3 model card.

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

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Jie Huang, Kerui Zhu, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. Deer: Descriptive knowledge graph for explaining entity relationships. *arXiv preprint arXiv:2205.10479*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. 2024. Realtime qa: what's the answer right now? *Advances in Neural Information Processing Systems*, 36.

Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. 2024. sdpo: Don't use your data all at once.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2023. Solar 10.7b: Scaling large language models with simple yet effective depth up-scaling.

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.

Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. 2023. Llatrieval: Llm-verified retrieval for verifiable generation. *arXiv preprint arXiv:2311.07838*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*.

He Lyu, Ningyu Sha, Shuyang Qin, Ming Yan, Yuying Xie, and Rongrong Wang. 2019. Advances in neural information processing systems. *Advances in neural information processing systems*, 32.

S. Van Landeghem M. Honnibal, I. Montani and A. Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

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. *arXiv preprint arXiv:2305.15294*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022b. musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

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

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. *arXiv preprint arXiv:2202.13296*.