ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240755

R^4: Reinforced Retriever-Reorder-Responder for Retrieval-Augmented Large Language Models

Taolin Zhang[a,1], Dongyang Li[a,b,2], Qizhou Chen[a,b], Chengyu Wang[a,*], Longtao Huang[a], Hui Xue[a], Xiaofeng He[b,] and Jun Huang[a]**

[a]Alibaba Group, China
[b]East China Normal University, China

Abstract. Retrieval-augmented large language models (LLMs) leverage relevant content retrieved by information retrieval systems to generate correct responses, aiming to alleviate the hallucination problem. However, existing retriever-responder methods typically append relevant documents to the prompt of LLMs to perform text generation tasks without considering the interaction of fine-grained structural semantics between the retrieved documents and the LLMs. This issue is particularly important for accurate response generation as LLMs tend to "lose in the middle" when dealing with input prompts augmented with lengthy documents. In this work, we propose a new pipeline named "Reinforced Retriever-Reorder-Responder" (R^4) to learn document orderings for retrieval-augmented LLMs, thereby further enhancing their generation abilities while the large numbers of parameters of LLMs remain frozen. The reordering learning process is divided into two steps according to the quality of the generated responses: document order adjustment and document representation enhancement. Specifically, document order adjustment aims to organize retrieved document orderings into beginning, middle, and end positions based on graph attention learning, which maximizes the reinforced reward of response quality. Document representation enhancement further refines the representations of retrieved documents for responses of poor quality via document-level gradient adversarial learning. Extensive experiments demonstrate that our proposed pipeline achieves better factual question-answering performance on knowledge-intensive tasks compared to strong baselines across various public datasets. The source codes and trained models will be released upon paper acceptance.

1 Introduction

Recently, large language models (LLMs) have attracted extensive attention, which are typically pre-trained on large datasets and implicitly store substantial amounts of world or domain knowledge [33, 45]. However, LLMs are also prone to the hallucination problem, and thus, they may generate erroneous responses [49]. In contrast, retrieval-augmented LLMs [17, 10, 47, 35] retrieve knowledge from an external datastore when needed, thereby reducing hallucinations and increasing the knowledge coverage in responses.

In the literature, there are two major research aspects in this field: (1) `Datastore Indexing` [17, 10, 44, 48] and (2) `Document Retrieval` [35, 27]. For `Datastore Indexing`, these approaches utilize pre-trained models to generate static embeddings for documents, which are viewed as mounted external memory, and they leverage various semantic similarities to enhance indexing. For `Document Retrieval`, the system initially retrieves a collection of relevant documents based on the semantic relevance between the user query and the documents. Then, the LLMs concatenate these highly related documents in an unordered manner to the prompt input [4], which makes LLMs better at answering factual questions. These methods essentially organize the information related to the user query from the perspective of coarse-grained memory , ignoring the fine-grained relationships between retrieved documents and the knowledge mastery characteristics of LLMs [14, 22]. For instance, the ordering of the `top-K` retrieved documents can be further adjusted to enhance the performance of retrieval-augmented LLMs in answering questions more accurately, as illustrated in Figure 1.

In this paper, we propose the **R**einforced **R**etriever-**R**eorder-**R**esponder framework (**R**^{4}) to formalize a new retrieval-augmented generation (RAG) pipeline. To reorder the retrieved `top-K` documents and enhance the response effectiveness of the LLMs, we divide the reorder learning process into the following two steps:

Document Order Adjustment: Prior research indicates that LLMs have a better recall of information at the beginning and the ending positions of retrieved documents in prompts [14, 22]. Hence, we have developed a graph-based reinforcement module that dynamically adjusts the ordering of retrieved documents in the prompt, according to the reward scores of graph document nodes. This module assigns important documents related to the query to positions as close as possible to the beginning or the end of the prompt while moving less relevant documents towards the middle. Thus, the documents are continuously adjusted to better orderings through the iterative feedback.

Document Representation Enhancement: In cases of poorly generated responses, we enhance the representations of retrieved documents obtained from document graph learning, thereby avoiding the online computational burden of retrieving new documents for the prompt. Here, we employ gradient adversarial learning to capture the gradient of each token position, which is considered as enhanced semantics concatenated with the learned document representations. Subsequently, we apply a similar document order adjustment learning step with the aim of generating improved responses.

In the experiments, we compare our R^4 framework against various

* Co-corresponding Author. Email: chengyu.wcy@alibaba-inc.com
** Co-corresponding Author. Email: hexf@cs.ecnu.edu.cn
[1] Equal contribution.
[2] Equal contribution.

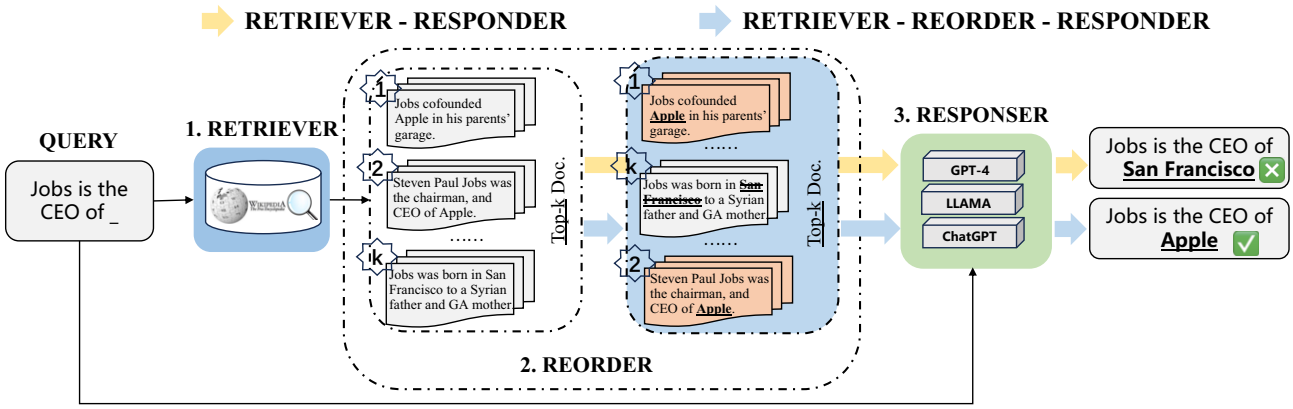

Figure 1: Different paradigms of retrieval-augmented approaches, including the traditional retriever-responder approach and our reinforced retriever-reorder-responder framework. Our pipeline emphasizes the importance of learning the key retrieved documents' ordering structure to assist LLMs in better addressing user queries related to factual knowledge (Best viewed in color).

baselines w.r.t. retrieval-augmented LLMs. The tasks include generative question-answering (QA), multi-choice QA, and dialogue-related tasks. Results show that R^4 significantly outperforms all baselines, thereby demonstrating the effectiveness of our approach.

2 Related Work

Large Language Models. The rapid advancement of LLMs is precipitating a revolutionary transformation in NLP. In recent years, significant advancements in LLMs such as GPT-3 [3], ChatGPT [30] and GPT-4 [1] have yielded groundbreaking performance in various NLP tasks. Concurrently, there has been a surge in the development of open-source LLMs based on LLAMA [40] and other foundational models such as Alpaca [39], which can be fine-tuned for specific applications. These models have achieved breakthroughs across a spectrum of tasks and some have been integrated as commercial products in everyday workflows [50, 42].

Retrieval-augmented Models. Augmenting language models with relevant information retrieved from various knowledge sources has proven to be effective in improving performance on diverse NLP tasks, including language modeling [17, 24] and open-domain question answering [12]. Specifically, (1) a retriever first obtains a set of documents (i.e., sequences of tokens) from a corpus, based on the input query, and then (2) a language model integrates the retrieved documents as additional context to produce a final prediction. This retrieval approach can be incorporated into both encoder-decoder [12] and decoder-only models [17, 34, 31]. For instance, Atlas [12] fine-tunes an encoder-decoder model along with a retriever by treating documents as latent variables, while REALM [8] adapts a decoder-only architecture to include retrieved texts and pre-trains the model anew. These methods necessitate updating the parameters of the model through gradient descent, an approach unsuitable for black-box LLMs. Another strand of research on retrieval-augmented LLMs, such as kNN-LM [17] and TRIME [52], introduces a system that retrieves a set of tokens and interpolates between the LLM's next-token distribution and kNN distributions computed from the retrieved tokens during inference. Concurrent studies [23, 36] indicate that using a static retriever can enhance GPT-3's [3] performance in open-domain question answering.

However, existing retrieval-augmented LLMs tend to only utilize the external sources by directly appending them to the query, while overlooking the nuanced structural and semantic interplay between the order of the retrieved documents and the LLMs.

3 Methodology

3.1 Model Overview and Important Notations

We introduce three main modules of our R^4 framework for retrieval-augmented LLMs, namely `Retriever`, `Reorder`, and `Responder`. An overview of our framework is shown in Figure 3. The `Retriever` module retrieves the query related documents to enhance the semantic understanding abilities of LLMs for answering factual questions. To further utilize the characteristics of positions in prompts [22], we propose the `Reorder` module to adjust the positions of retrieved documents. Finally, we concatenate the query and the adjusted documents as the input for the following inference and training in the `Responder` module.

We state some basic notations as follows. The hidden representations of a collection of retrieved documents $(d_1, d_2, \ldots, d_n)$ are denoted as $(h_{d_1}, d_{d_2}, \ldots, h_{d_n})$ and $h_{d_i} \in \mathcal{R}^{d_1}$, where n is the number of the retrieved documents related to a user query and d_1 is dimension of the output representation of the dense encoder (i.e., BERT [6]). During the graph learning process, the hidden representations of pseudo document nodes $h_{ps} \in \mathcal{R}^{d_1}$ (which include three position types) are aggregated from the corresponding documents assigned to certain positions.

3.2 Retriever

In our implementation, we utilize the Dense Passage Retriever (DPR) [16] to retrieve relevant Wikipedia documents[3] in response to a user query. The DPR employs a dense encoder (e.g., BERT [6]) to encode texts into context-aware representations, retrieving the `top-K` documents whose embeddings are nearest to that of the query. The similarity between a document and the query is computed as: $sim(que, doc) = E_Q(que)^T \cdot E_D(doc)$ where que and doc represent the query and the document, respectively, and E_Q and E_D denote the corresponding encoders.

[3] https://www.wikipedia.org/

3.3 Reorder

In this section, we present our document reordering technique, which adjusts the ordering of documents in the input prompt. Ideally, pivotal information (i.e., Query-related Documents) should be foregrounded and placed at the beginning and end to augment the LLM's capacity to respond accurately to user queries [14, 22].

3.3.1 Document Order Adjustment

According to [22], retrieved documents can be organized into three positional segments: beginning, middle, and end, as the prompt's initial and final parts significantly influence the response's effectiveness. In this component, we employ document graph learning to derive representations for each document in the graph and categorize document nodes into positions of $\{beginning, mid, end\}$. Subsequently, a reinforcement-based mechanism is employed to refine the graph's structure. This enhances the coherence among document nodes in proximal positions with analogous relevance, and distances contrasting nodes with discrepant meanings.

(1) Heterogeneous Query-Document Graph Construction: The process for constructing a heterogeneous graph comprising queries and documents is depicted in Figure 2 and involves the two parts:

i) *Homogeneous Document Graph:* We start by fully connecting all retrieved documents, regardless of their positions. Next, we introduce positional pseudo document nodes for each position type. The representation of the node is the mean pooled representations of document nodes assigned to the position, which serves as a unified semantic hub.

ii) *Heterogeneous Query-Document Graph:* The query node is at the core of the heterogeneous graph. Each positional pseudo document node is connected to the query node to form the overall structure. Furthermore, to integrate the retrieved documents with the query, we establish additional connections from all document nodes to the query. These connections facilitate the learning of graph representations during the feedback training phase.

As training progresses, the edges among the nodes dynamically evolve, guided by the graph rewards.

(2) Node Initialization: We utilize BERT [6] to initialize the representations of document nodes. Given that each document is composed of multiple tokens, we adopt self-attentive pooling [21] over these tokens to aggregate their information into a single document representation, denoted as h_{d_i}. Subsequently, we arbitrarily cluster all document nodes into three categories (i.e., position types), corresponding to their assigned positions. To promote the graph learning propagation process between nodes stably [19], we construct a pseudo node in each positions. The representation for each positional pseudo node, h_{ps_τ}, is computed as the average of the hidden states of document nodes within the category. Specifically, $h_{ps_\tau} = Avg(h_{d_i}, \ldots, h_{d_j})$, where the document nodes $(d_i, \ldots, d_j)$ are uniformly associated with one of the position types $\tau \in \{beginning, mid, end\}$.

(3) Node Representation Learning: To facilitate the learning of these types, we adapt the R-GCN model [32]. Consider the hidden representation of the graph layer at layer $(l+1)$ as:

$$H^{(l+1)} = \sigma\left(\tilde{A} \cdot H^{(l)} \cdot W_A^{(l)} + H^{(l)} \cdot W_d^{(l)}\right) \tag{1}$$

where $\tilde{A} \in \mathbb{R}^{|\mathcal{N}_d| \times |\mathcal{N}_d|}$ represents the fully connected adjacency matrix including nodes and self-connections, with $\mathcal{N}_d$ being the set of all document nodes in the graph. $W_A^{(l)}$ and $W_d^{(l)}$ are the layer-specific trainable parameters, and σ denotes the activation function. The index l refers to the layer number. Given that query nodes connected to different neighboring documents can exhibit varying degrees of relevance, our model dynamically learns the adjacency matrix $\tilde{A}$. This learning process incorporates two types of attention mechanisms:

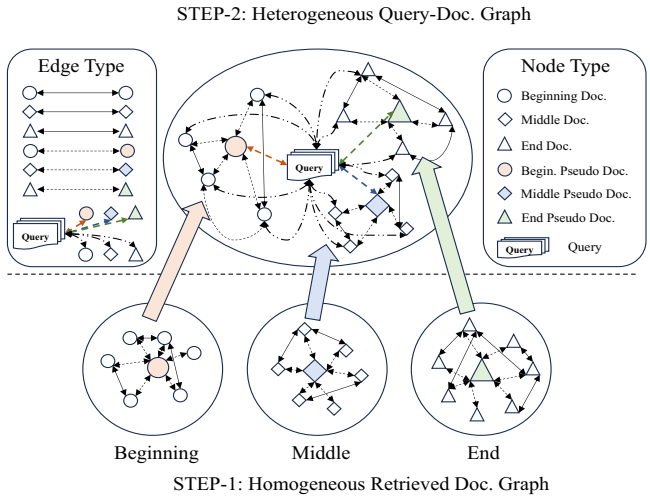

Figure 2: Illustration of the heterogeneous graph construction process between a query and retrieved documents within the R^4 framework.

- **Position-level Attention:** For a given query node n_q, this attention mechanism is designed to evaluate the significance of various document positions that neighbor n_q. We aggregate the hidden states of all neighboring document nodes n'_d assigned a specific positional document type τ into their composite representation $h_\tau = \sum_{n'_d} h_{n'_d}$. The attention scores for the three distinct positional node types are then computed based on their relation to the query node, calculated as follows:

$$\alpha_\tau = \sigma\left(\mu_\tau^T \left[h_{n_d} W_{n_d} \oplus h_\tau W_\tau\right]\right) W_\alpha + b_\alpha \tag{2}$$

where μ_τ^T represents the attention vector specific to positional type τ, and $\oplus$ denotes concatenation. The matrices W_{n_d}, W_τ, W_α, and the bias term b_α are trainable parameters. The normalized attention scores α_τ for each position type are given by:

$$\alpha_\tau = \frac{\exp(\alpha_\tau)}{\sum_{\tau' \in \Gamma} \exp(\alpha_{\tau'})},$$

with $\Gamma = \{beginning, mid, end\}$.

- **Node-level Attention:** Given a specific query node n_q, the importance of neighboring document nodes n'_d varies across the different position types τ. To address this, we compute the attention at the node level:

$$\beta_{n_q n'_d} = \sigma\left(\nu_{n_q}^T \cdot \alpha_\tau \left[W_{n_q} h_{n_q} \oplus W_{n'_d} h_{n'_d}\right]\right) \tag{3}$$

where $\nu_{n_q}^T$ denotes the node-level attention vector, and α_τ is the weight assigned to document positions by the previous attention mechanism. $\nu_{n_q}^T$, W_{n_q}, and $W_{n'_d}$ are learnable parameters. The normalized weights for the document nodes are computed as:

$$\beta_{n_q n'_d} = \frac{\exp\left(\beta_{n_q n'_d}\right)}{\sum_{n'_d \in \mathcal{N}_d} \exp\left(\beta'_{n_q n'_d}\right)} \tag{4}$$

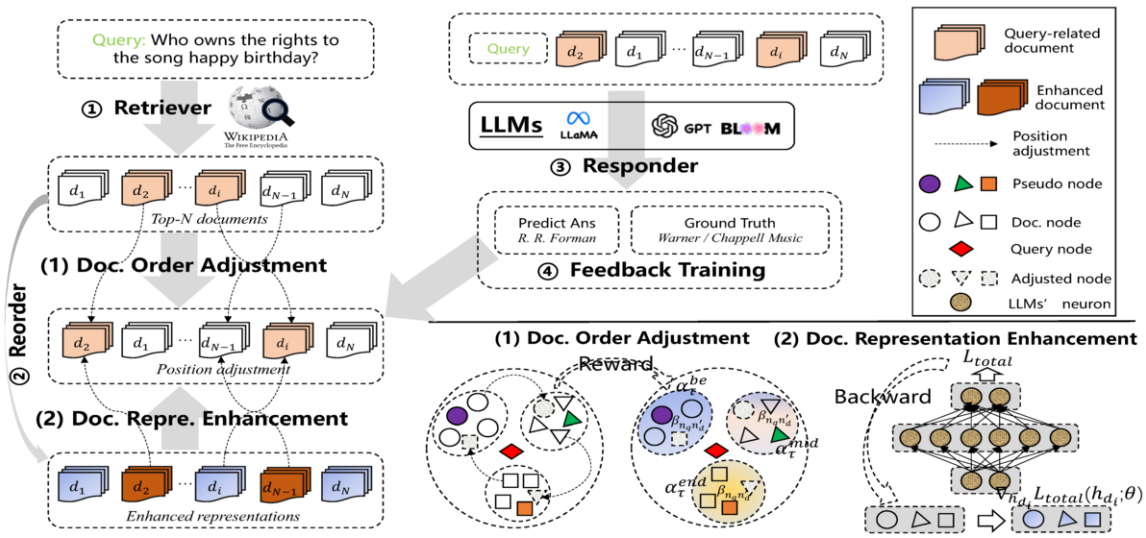

Figure 3: Model Overview. `Document Order Adjustment`: document positions are dynamically adjusted in the clusters according to the feedback. `Document Representation Enhancement`: document representations are updated by the weight gradient of the training loss (Best viewed in color).

Finally, we incorporate the aforementioned attention mechanisms to refine the adjacency matrix $\tilde{A}$ in Equation 1. Specifically, the element located at the intersection of the n_q-th row and n'_d-th column within $\tilde{A}$ is substituted with $\beta_{n_q n'_d}$. Through this process, graph node representations, denoted by h_{d_i}, are obtained via the `Node Representation Learning`. The representations encapsulate the contextual information distilled through both position-level and node-level attention, thereby enabling a more nuanced understanding of the document-query relationships within our model.

(4) Reinforced Order Adjustment: We employ a reinforcement learning (RL) strategy aimed at dynamically adjusting the node positions within the graph according to the generative quality of answers. Specifically, our RL method involves manipulating the node distribution, essentially pushing and pulling nodes across different positional categories, guided by the interaction with the pseudo node of each set. This approach allows for an adaptive reorganization of nodes, optimizing the arrangement based on the latent learning preferences of the LLM, and reinforcing the model's ability to prioritize critical information effectively.

State: We extract the hidden representation for each document in the graph, denoted as h_{d_i}. Due to the possibility of adjusting each node in the graph to other positions, these representations couple with the aggregated representations of pseudo nodes, constitute the state S_i:

$$S_i = [h_{d_i} \oplus h_{ps_{\text{be}}} \oplus h_{ps_{\text{mid}}} \oplus h_{ps_{\text{end}}}] \tag{5}$$

Policy: The RL policy, π_θ, delineates a probabilistic approach towards predicting the most strategic document position. It utilizes the current state S_i to propose an action a_i, expressed as:

$$\pi_\theta(a_i \mid S_i) = P(a_i \mid S_i) \tag{6}$$

where θ indicates trainable parameters.

Action: The RL action entails selecting a suitable document position, informed by the corresponding pseudo node. The choice for the i-th document node is represented by a one-hot vector:

$$a_i = \{1, \cdots, 0\} \in \mathbb{R}^3, \quad a_i \sim \pi_\theta(a_i \mid S_i) \tag{7}$$

with 1 signifying the document's affiliation to a selected positional set, and 0 indicating non-affiliation.

Reward: Our objective is to enhance the model's proficiency in accurately allocating documents to positions. The reward is quantified by the semantic similarity between the document node and the pseudo node selected by the action: $r_i = \text{sim}(h_{d_i}, h^a_{ps_\tau})$ where $h^a_{ps_\tau}$ is the action-selected pseudo node's representation. The cumulative reward is determined as $R_{\text{cum}} = \sum_{i=1}^{N} r_i$, leading to the objective:

$$J_\theta = \mathbb{E}_{s_i, a_i, r_i}\left[\sum_{i=1}^{N} r_i\right] \tag{8}$$

We employ the policy gradient method [37] through the implementation of the REINFORCE algorithm [46], augmented with a baseline mechanism [43] to optimize our RL objective efficiently.

3.3.2 *Document Representation Enhancement*

Occasionally, the R^4 pipeline may not align well with ground-truth responses, prompting a need for further optimization of document representations. To achieve this without incurring significant computational overhead, we leverage already retrieved documents for semantic enhancement rather than fetching new documents from the datastore. We begin by computing the document-level gradient g of our total loss function $\mathcal{L}_{total}$ with respect to each document's hidden representation h_{d_i}, and the model parameters θ:

$$g_i = \nabla_{h_{d_i}} \mathcal{L}_{total}(h_{d_i}; \theta) \tag{9}$$

This gradient is then used to generate a corresponding gradient perturbation, pt_{adv_i}, through differentiation: $pt_{adv_i} = \epsilon \cdot \frac{g_i}{\|g_i\|}$ where $\|g_i\|$ denotes the norm of gradient g_i, and ϵ is a hyper-parameter configuring the scale of this norm. Consequently, the enhanced document representation is formulated as:

$$\widetilde{h}_{d_i} = h_{d_i} \oplus pt_{adv_i} \tag{10}$$

which employs element-wise addition between vectors. These refined document vectors are then reintegrated into the `Document Order Adjustment` module for improved fine-tuning, ordered based on their rewarded positions r_i.

Table 1: General results of our R^4 model over the public datasets. T-tests demonstrate the improvements of our work are statistically significant with p<0.05.

Dataset	Model	Rogue-1 10	Rogue-1 15	Rogue-1 20	Rogue-1 Avg.	Bleu-4 10	Bleu-4 15	Bleu-4 20	Bleu-4 Avg.
NQ	REALM	32.3	35.6	36.7	$34.9_{(\pm 0.4)}$	7.08	7.22	7.23	$7.18_{(\pm 0.24)}$
	ICR	40.4	41.7	42.0	$41.4_{(\pm 0.3)}$	7.59	7.61	7.68	$7.63_{(\pm 0.21)}$
	REPLUG	41.9	43.8	44.6	$43.4_{(\pm 0.2)}$	7.46	7.48	7.65	$7.53_{(\pm 0.20)}$
	Selfmem	42.6	43.5	45.9	$44.0_{(\pm 0.3)}$	7.51	7.63	7.79	$7.64_{(\pm 0.16)}$
	SELF-RAG	42.7	45.4	46.2	$44.8_{(\pm 0.4)}$	7.25	7.72	7.87	$7.61_{(\pm 0.27)}$
	FILCO	39.2	41.1	45.3	$41.9_{(\pm 0.1)}$	7.36	7.44	7.89	$7.56_{(\pm 0.13)}$
	LongLLMLingua	38.1	42.2	44.5	$41.6_{(\pm 0.3)}$	7.15	7.29	7.26	$7.23_{(\pm 0.14)}$
	Ours	**44.7**	**46.5**	**47.5**	**46.2**$_{(\pm 0.1)}$	**7.93**	**8.14**	**8.63**	**8.2**$_{(\pm 0.08)}$
TriviaQA	REALM	22.6	23.9	24.2	$23.6_{(\pm 0.3)}$	6.41	6.55	6.82	$6.59_{(\pm 0.19)}$
	ICR	25.2	26.5	26.8	$26.2_{(\pm 0.2)}$	7.59	7.60	7.78	$7.66_{(\pm 0.09)}$
	REPLUG	27.0	27.3	28.1	$27.5_{(\pm 0.2)}$	7.47	7.66	7.83	$7.65_{(\pm 0.15)}$
	Selfmem	26.1	26.4	27.9	$26.8_{(\pm 0.4)}$	7.65	7.72	7.89	$7.75_{(\pm 0.24)}$
	SELF-RAG	26.8	27.1	27.6	$27.2_{(\pm 0.1)}$	7.73	7.87	7.95	$7.85_{(\pm 0.07)}$
	FILCO	27.5	27.7	28.0	$27.7_{(\pm 0.3)}$	7.80	7.83	8.01	$7.88_{(\pm 0.14)}$
	LongLLMLingua	26.9	27.5	27.9	$27.4_{(\pm 0.1)}$	7.92	7.84	8.13	$7.96_{(\pm 0.11)}$
	Ours	**28.8**	**28.9**	**29.3**	**29.0**$_{(\pm 0.2)}$	**8.29**	**8.52**	**8.74**	**8.51**$_{(\pm 0.11)}$
MultiDoc2Dial	REALM	20.2	21.6	22.4	$21.4_{(\pm 0.5)}$	7.60	7.96	8.01	$7.86_{(\pm 0.36)}$
	ICR	20.8	22.7	24.3	$22.6_{(\pm 0.4)}$	8.29	8.42	8.74	$8.48_{(\pm 0.25)}$
	REPLUG	21.5	23.2	23.8	$22.8_{(\pm 0.2)}$	8.37	8.53	8.69	$8.53_{(\pm 0.13)}$
	Selfmem	22.7	23.6	24.1	$23.5_{(\pm 0.4)}$	8.43	8.66	8.85	$8.65_{(\pm 0.21)}$
	KnowledGPT	21.6	23.9	24.4	$23.3_{(\pm 0.2)}$	8.64	8.81	8.90	$8.78_{(\pm 0.15)}$
	SELF-RAG	22.1	24.3	24.6	$23.7_{(\pm 0.3)}$	8.57	8.76	9.03	$8.79_{(\pm 0.19)}$
	FILCO	23.4	24.5	24.9	$24.3_{(\pm 0.2)}$	8.91	9.05	9.22	$9.06_{(\pm 0.17)}$
	LongLLMLingua	23.1	24.9	25.2	$24.4_{(\pm 0.1)}$	8.84	9.23	9.16	$9.08_{(\pm 0.23)}$
	Ours	**25.2**	**26.1**	**25.9**	**25.7**$_{(\pm 0.1)}$	**9.38**	**9.67**	**9.97**	**9.67**$_{(\pm 0.04)}$
CMU DoG	REALM	9.6	10.3	10.7	$10.2_{(\pm 0.5)}$	6.22	6.37	6.40	$6.33_{(\pm 0.29)}$
	ICR	12.0	12.4	12.8	$12.4_{(\pm 0.3)}$	7.14	7.62	7.75	$7.50_{(\pm 0.25)}$
	REPLUG	12.5	12.9	13.1	$12.8_{(\pm 0.4)}$	7.46	7.79	7.84	$7.70_{(\pm 0.21)}$
	Selfmem	12.7	13.1	13.4	$13.1_{(\pm 0.3)}$	7.75	7.90	7.92	$7.86_{(\pm 0.17)}$
	KnowledGPT	13.9	14.5	14.6	$14.3_{(\pm 0.2)}$	8.01	8.05	8.23	$8.10_{(\pm 0.15)}$
	SELF-RAG	13.3	13.6	14.2	$13.7_{(\pm 0.3)}$	7.89	8.04	8.19	$8.04_{(\pm 0.11)}$
	FILCO	13.8	14.0	14.4	$14.1_{(\pm 0.2)}$	8.03	8.18	8.25	$8.15_{(\pm 0.09)}$
	LongLLMLingua	13.2	13.5	14.1	$13.6_{(\pm 0.1)}$	7.84	8.25	7.95	$8.01_{(\pm 0.06)}$
	Ours	**14.8**	**14.9**	**15.6**	**15.1**$_{(\pm 0.3)}$	**8.87**	**9.10**	**8.96**	**8.98**$_{(\pm 0.22)}$

		Accuracy 10	Accuracy 15	Accuracy 20	Avg.
MMLU	REALM	84.7	85.3	85.3	$85.1_{(\pm 0.5)}$
	ICR	86.2	86.9	87.1	$86.7_{(\pm 0.3)}$
	REPLUG	85.3	86.8	87.6	$86.6_{(\pm 0.3)}$
	Selfmem	87.5	87.9	88.4	$87.9_{(\pm 0.4)}$
	SELF-RAG	87.3	88.5	88.7	$88.2_{(\pm 0.2)}$
	FILCO	88.1	88.4	89.2	$88.6_{(\pm 0.4)}$
	LongLLMLingua	88.0	86.5	88.7	$87.7_{(\pm 0.2)}$
	Ours	**90.5**	**90.4**	**90.3**	**90.4**$_{(\pm 0.1)}$

3.4 Responder

Document positions are dynamically adjusted leveraging the current state of the graph network. Thus, the prompt construction process involves concatenating the query with retrieved documents, taking into account their dynamically adjusted positions. Given a query que and documents positioned as $(d_{begin1}, \cdots, d_{end1}, d_{end2})$, the inputs are formulated and presented to the LLM $\mathcal{M}$ as:

$$Ans = \mathcal{M}(que, d_{begin1}, \cdots, d_{end1}, d_{end2}) \tag{11}$$

where Ans represents the LLM's output prediction.

3.5 Feedback Training

To quantify the semantic disparities between predicted outcomes and ground truth, we employ the BLEU [25] and ROUGE [20] metrics. This comparison informs the optimization of future prompts, with the objective of enhancing model performance. Specifically, the BLEU score s_{BLEU} serves as a loss coefficient, guiding the adjustment of the reward optimization direction. This mechanism is encapsulated in the total loss function, defined as:

$$\mathcal{L}_{\text{total}} = s_{\text{BLEU}} \cdot J_\theta - f_{dis}(Ans, Gt) \tag{12}$$

where J_θ signifies the RL loss, and f_{dis} is the Lipschitz distance [38], a measure designed for comparing string data [4].Gt denotes the ground-truth answer string.

[4] This similarity calculation method can also be used as an alternative to other string type distance algorithms

Table 2: The comparison of different retriever types in terms of Rouge-1 (%) and Accuracy (%).

Retriever Type	Model	NQ Rouge-1	TriviaQA Rouge-1	MMLU Accuracy	MultiDoc2Dial Rouge-1	CMU DoG Rouge-1	Avg.
Sparse	TF-IDF [28]	21.9	11.4	70.9	18.8	9.4	$26.5_{(\pm 0.4)}$
	BM25 [29]	32.3	12.6	75.4	20.5	10.2	$30.2_{(\pm 0.2)}$
Dense	Spider [26]	41.8	27.9	80.3	24.7	13.7	$37.7_{(\pm 0.4)}$
	Contriever [11]	40.7	28.2	87.1	25.1	14.5	$39.1_{(\pm 0.3)}$
	DPR [16]	46.5	28.3	89.7	25.0	14.8	$40.9_{(\pm 0.1)}$

Table 3: Ablation study in terms of Rouge-1 (%) and Accuracy (%).

	NQ Rouge-1	MMLU Accuracy	MultiDoc2Dial Rouge-1
Ours	46.2	90.4	25.7
- Graph Doc. Learning	44.7	87.2	23.3
- Doc. Enhancement	43.4	87.6	23.1
- Rein. Order Adjustment	42.9	86.2	22.1

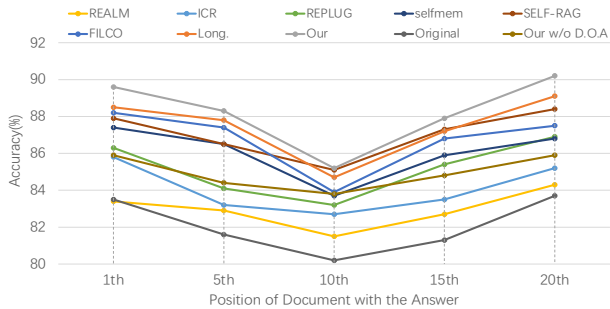

Figure 4: Results w.r.t key retrieved document positions.

4 Experiments

4.1 Datasets

We evaluate our model using several datasets, classified into three types: Generative QA, Multi-choice QA, and Dialogue.

Generative QA: Natural Questions (NQ) [18] focuses on real user questions and includes short answer types, which are used in our experiments. **TriviaQA** [15] is known for its complexity due to syntactic and lexical variability between questions and answers.

Multi-choice QA: Massive Multitask Language Understanding (MMLU) [9] is a comprehensive benchmark with samples from various domains and a wide range of difficulty levels.

Dialogue: MultiDoc2Dial [7] is a document-grounded dialogue dataset sourced from realistic scenarios across four distinct domains. **CMU DoG** [53] centers on textual conversations with topics derived from popular movies outlined in Wikipedia articles.

4.2 Baselines and Experimental Settings

We compare the efficacy of our model against a set of baseline models. **REALM** [8] is BERT-style Transformer model augmented with a knowledge retriever that sources relevant text from Wikipedia. **In-Context RALM (ICR)** [27] improves model performance using an existing retrieval tool, without additional language model training. **REPLUG** [35] enhances a language model treated as a black box by appending it with externally retrieved documents. **Selfmem** [5] implements a dynamic, unbounded memory selection mechanism to empower generative models. **KnowledGPT** [51] introduces knowledge-grounded dialogue generation that integrates knowledge selection and response generation phases. **SELF-RAG** [2] utilizes reflection tokens to evaluate retrieval requirements and assess the quality of retrieved content. **FILCO** [41] utilizes a trained filter to refine the retrieval context based on criteria such as string inclusion, lexical overlap, and conditional cross-mutual information. **LongLLMLingua** [14] proposes a question-aware coarse-to-fine compression technique to concentrate key information within prompts and implements document reordering to mitigate information loss.

We utilize Llama2 (7B) [40] as the foundation for our model. The results reported are the average across multiple runs, each initialized with a different random seed. In our experiments, the hyperparameter N varies within the set {10, 15, 20}. We impose a maximum token length constraint of 512 for any individual document. The gradient perturbation hyper-parameter ϵ is fixed at 2. Our model's graph architecture is composed of three layers, with the graph document node set $\mathcal{N}_d$ comprising N document nodes alongside three pseudo nodes corresponding to distinct positional types. During inference, we configure the temperature hyper-parameter to 0.6 and set the top-p sampling parameter to 0.9, facilitating controlled text generation that balances creativity and coherence.

4.3 General Results

This section presents a comparative evaluation of the R[4] framework. We employ ROUGE-1 and BLEU-4 as evaluation metrics to report the model performance for generative QA. Accuracy as the primary metric for multi-choice QA. In evaluating our framework, we investigate varying numbers of retrieved documents—including 10, 15, and 20—to assess their influence on the outcomes. We ensure an equitable comparison by employing the same LLAMA2 backbone [40] across different baselines. Table 1 illustrates that: (1) An increased count of retrieved documents correspondingly enhances the performance of both our model and the baselines, suggesting that providing LLMs with more query-related documents is an effective tactic to better fulfill user queries. (2) Our model exhibits notable advancements in performance compared to other retrieval-augmented LLMs. The performance gains can be attributed primarily to our methodological enhancements, specifically, the strategic ordering of document learning and enhanced document representations.

4.4 Detailed Analysis

4.4.1 The Influence of Answer Positions

Our analysis targets the reordering efficacy by examining pivotal positions of key retrieved documents. Utilizing MMLU [9], we assess our fully equipped R[4] model against a baseline retriever-responder (Original) [5], with both methods employing the top-20 documents. In

[5] The baseline retriever-responder method retrieves the `top-K` documents and concatenates them with the query, enabling LLMs to generate a response without order adjustment.

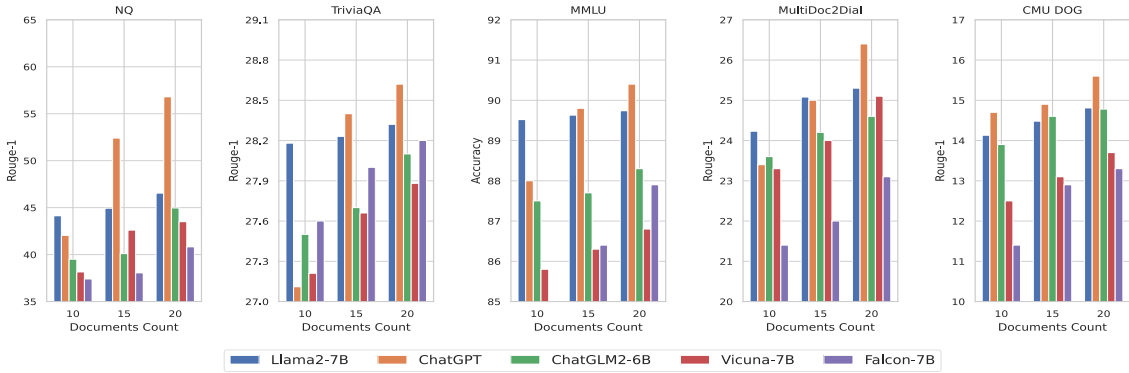

Figure 5: Comparison between different LLM backbones of our R^4 pipeline with various retrieved document counts.

our variant, we focus on analyzing document placements in positions 1, 5, 10, 15, and 20, contrasting with the dynamic determination of document positioning by the complete R^4 model. For various baselines, we randomly shuffle the order of retrieved documents and then place the key documents in their corresponding positions.

Observation from Figure 4 reveal that: (1) Our variant's performance aligns with that of the baseline, emphasizing the significance of beginning and ending document positions for LLMs' comprehension. Nonetheless, the R^4 model still outperforms the baseline due to representation enhancement. (2) The baselines also exhibit a position sensitive phenomenon that the position at the beginning and end is more effective than the middle position. (3) The R^4 model demonstrates a steady and robust output irrespective of the initial positions of key documents. This substantiates the notion that the ordering and optimization of documents inherently bolster the capacity of LLMs in addressing user queries in RAG systems.

4.4.2 *The Influence of Different Retrievers*

To evaluate the impact of retrieval mechanisms, we implemented benchmarks using both dense and sparse retrievers. Sparse retrievers, such as TF-IDF [28] and BM25 [29], are grounded in token frequency-centric metrics, contrasting with dense retrievers represented by Spider [26], Contriever [11], and DPR [16]. Findings presented in Table 2 support the following conclusions: (1) Dense retrievers surpass sparse alternatives, showcasing superior performance and consistency across tasks. (2) Among dense retrieval models, those employing direct negative sampling techniques, such as DPR [16], notably improve query document discrimination, which is achieved through contrastive learning [13].

4.4.3 *The Influence of Different Backbones*

The advent of ChatGPT [30] has ushered in an era of advanced LLMs. In our study, we probe the adaptability of the R^4 framework using a range of LLMs as underlying backbones while standardizing all other facets. Furthermore, we examine how the quantity of retrieved documents—specifically sets of 10, 15, and 20—affects model performance. Insights derived from Figure 5 indicate that: (1) The performance of our approach improves in tandem with the quantity of retrieved documents across all LLM backbones. This pattern underlines the resilience of the R^4 framework to changes in document volume. (2) While it is evident that performance gains diminish as the number of retrieved documents increases, ChatGPT exhibits remarkable stability across different document counts.

4.4.4 *Ablation Study*

An ablation study in Table 3 dissects our model's performance enhancers by individually retracting key components: graph document learning, document enhancement, and reinforced order adjustment. This investigation pivots on QA and dialogue tasks, focusing on datasets which extensively feature background knowledge. In the experiments, removing the graph document learning component reverts document representation to BERT's original embeddings. The absence of document enhancement halts the enhancement feedback loop. Without reinforced order adjustment, the training loss is solely predicated on minimizing the Lipschitz distance [38]. Our observations establish that: (1) Excluding the reinforced order adjustment incurs a substantial performance degradation, accentuating its pivotal role in refining document ordering. (2) Withdrawing any component impairs the model's effectiveness, validating each element's contribution to the synergistic pipeline that bolsters semantic understanding of user queries in RAG.

5 Conclusion

In this study, we introduced the R^4 pipeline, a novel framework designed to refine the RAG framework. Central to the R^4 pipeline are two innovative mechanisms: the document order adjustment and document representation enhancement learning modules. Our empirical evaluation across a variety of knowledge-intensive scenarios illustrates the R^4 pipeline's superior performance, facilitating a more nuanced understanding and organization of retrieved documents. This study's implications suggest that through targeted adjustments in document ordering and representation, we can further harness the potential of RAG in responding to complex queries, paving the way for advancements in automated QA systems.

References

[1] Gpt-4 technical report. In *OpenAI*, OpenAI. 2023.

[2] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *CoRR*, abs/2310.11511, 2023.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] D. Cheng, S. Huang, J. Bi, Y. Zhan, J. Liu, Y. Wang, H. Sun, F. Wei, D. Deng, and Q. Zhang. UPRISE: universal prompt retrieval for improving zero-shot evaluation. *CoRR*, abs/2303.08518, 2023.

[5] X. Cheng, D. Luo, X. Chen, L. Liu, D. Zhao, and R. Yan. Lift yourself up: Retrieval-augmented text generation with self memory. *CoRR*, abs/2305.02437, 2023.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

[7] S. Feng, S. S. Patel, H. Wan, and S. Joshi. Multidoc2dial: Modeling dialogues grounded in multiple documents. In *EMNLP*, pages 6162–6176, 2021.

[8] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[9] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[10] G. Izacard, P. S. H. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave. Few-shot learning with retrieval augmented language models. *CoRR*, abs/2208.03299.

[11] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022, 2022.

[12] G. Izacard, P. S. H. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43, 2023.

[13] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *CoRR*, abs/2011.00362, 2020.

[14] H. Jiang, Q. Wu, X. Luo, D. Li, C. Lin, Y. Yang, and L. Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *CoRR*, abs/2310.06839, 2023.

[15] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611, 2017.

[16] V. Karpukhin, B. Oguz, S. Min, P. S. H. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781, 2020.

[17] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020.

[18] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019.

[19] Y. Li, J. Yin, and L. Chen. Informative pseudo-labeling for graph neural networks with few labels. *Data Min. Knowl. Discov.*, 37(1):228–254, 2023.

[20] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004.

[21] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[22] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *CoRR*, abs/2307.03172, 2023.

[23] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, pages 9802–9822, 2023.

[24] S. Min, W. Shi, M. Lewis, X. Chen, W. Yih, H. Hajishirzi, and L. Zettlemoyer. Nonparametric masked language modeling. In *ACL*, pages 2097–2118, 2023.

[25] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[26] O. Ram, G. Shachaf, O. Levy, J. Berant, and A. Globerson. Learning to retrieve passages without supervision. In *NAACL*, pages 2687–2700, 2022.

[27] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham. In-context retrieval-augmented language models. *CoRR*, abs/2302.00083, 2023.

[28] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *ICML*, number 1, pages 29–48. Citeseer, 2003.

[29] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

[30] K. I. Roumeliotis and N. D. Tselikas. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192, 2023.

[31] O. Rubin, J. Herzig, and J. Berant. Learning to retrieve prompts for in-context learning. In *NAACL*, pages 2655–2671, 2022.

[32] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.

[33] M. Shanahan. Talking about large language models. *CoRR*, abs/2212.03551, 2022.

[34] W. Shi, J. Michael, S. Gururangan, and L. Zettlemoyer. Nearest neighbor zero-shot inference. In *EMNLP*, pages 3254–3265, 2022.

[35] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652, 2023.

[36] C. Si, Z. Gan, Z. Yang, S. Wang, J. Wang, J. L. Boyd-Graber, and L. Wang. Prompting GPT-3 to be reliable. In *ICLR*, 2023.

[37] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, pages 1057–1063, 1999.

[38] K. Suzuki and Y. Yamazaki. Non-separability of the lipschitz distance. *Pacific Journal of Mathematics for Industry*, 7(1), mar 2015.

[39] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. 2023.

[40] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[41] Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig. Learning to filter context for retrieval-augmented generation. *CoRR*, abs/2311.08377, 2023.

[42] Z. Wang, W. Wang, Z. Li, L. Wang, C. Yi, X. Xu, L. Cao, H. Su, S. Chen, and J. Zhou. Xuat-copilot: Multi-agent collaborative system for automated user acceptance testing with large language model. *CoRR*, abs/2401.02705, 2024.

[43] L. Weaver and N. Tao. The optimal reward baseline for gradient-based reinforcement learning. In *UAI*, pages 538–545, 2001.

[44] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

[45] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[46] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, pages 229–256, 1992.

[47] M. Yasunaga, A. Aghajanyan, W. Shi, R. James, J. Leskovec, P. Liang, M. Lewis, L. Zettlemoyer, and W. Yih. Retrieval-augmented multimodal language modeling. In *ICML*, pages 39755–39769, 2023.

[48] S. Ye, D. Kim, S. Kim, H. Hwang, S. Kim, Y. Jo, J. Thorne, J. Kim, and M. Seo. FLASK: fine-grained language model evaluation based on alignment skill sets. *CoRR*, abs/2307.10928, 2023.

[49] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, L. Wang, A. T. Luu, W. Bi, F. Shi, and S. Shi. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219, 2023.

[50] Y. Zhang, A. Maezawa, G. Xia, K. Yamamoto, and S. Dixon. Loop copilot: Conducting AI ensembles for music generation and iterative editing. *CoRR*, abs/2310.12404, 2023.

[51] X. Zhao, W. Wu, C. Xu, C. Tao, D. Zhao, and R. Yan. Knowledge-grounded dialogue generation with pre-trained language models. In *EMNLP*, pages 3377–3390, 2020.

[52] Z. Zhong, T. Lei, and D. Chen. Training language models with memory augmentation. In *EMNLP*, pages 5657–5673, 2022.

[53] K. Zhou, S. Prabhumoye, and A. W. Black. A dataset for document grounded conversations. In *EMNLP*, pages 708–713, 2018.