Unraveling and Mitigating Retriever Inconsistencies in Retrieval-Augmented Large Language Models

Anonymous ACL submission

Abstract

Although Retrieval-Augmented Large Language Models (RALMs) demonstrate their superiority in terms of factuality, they do not consistently outperform the original retrieval-free Language Models (LMs). Our experiments reveal that this example-level performance inconsistency exists not only between retrieval-augmented and retrieval-free LM but also among different retrievers. To understand this phenomenon, we investigate the degeneration behavior of RALMs and theoretically decompose it into four categories. Further analysis based on our decomposition reveals that the innate difference in knowledge sources and the unpredictable degeneration of the reader model contribute most to the inconsistency. Drawing from our analysis, we introduce Ensemble of Retrievers (EoR), a trainable framework that can adaptively retrieve from different knowledge sources and effectively decrease unpredictable reader errors. Our experiments on Open Domain Question Answering show that EoR substantially improves performance over the RALM with a single retriever by considerably reducing inconsistent behaviors.

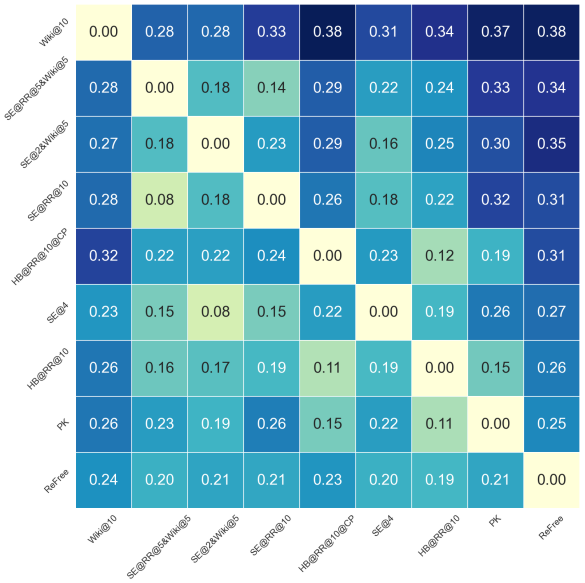

Figure 1: Retriever-to-Retriever Relative Win Ratio heatmap on Natural Questions with ChatGPT as LM. Each cell's number represents the proportion of questions answered incorrectly by the column retriever that was correctly answered by the row retriever. 0 represents all questions correctly answered by the row retriever can be correctly answered by the column retriever, which implies the column retriever consistently outperform the row retriever. See equation 1 for formal definition.

1 Introduction

Although Large Language Models (LLMs) have shown their superiority in many NLP tasks (OpenAI, 2023; Touvron et al., 2023), they are known to struggle with factual hallucinations (Mallen et al., 2023; Bang et al., 2023) and outdated parametric knowledge (Dhingra et al., 2022; Vu et al., 2023). Retrieval-Augmented Language Models (RALMs) as an ad-hoc technique have been proven to effectively alleviate these problems (Ram et al., 2023; Vu et al., 2023). In most RALM systems, a **retriever** takes the responsibility to retrieve relevant information from some external knowledge sources (e.g., Wikipedia dump (Lewis et al., 2020), search engine (Nakano et al., 2021), parametric knowledge (Yu et al., 2023)) and process them into text chunks to extract the most pertinent content for subsequent generation with a reader model (e.g., filtering, reranking (Liu et al., 2023b), compression (Xu et al., 2023)).

Although RALMs show their effectiveness at the corpus level (Gao et al., 2023b), retrieval-augmentation does not consistently promote the original retrieval-free LM but sometimes hurts its performance at the individual example level (Mallen et al., 2023; Yoran et al., 2023; Asai et al., 2023). In this work, we observe that variability in example-level performance exists not only between retrieval-augmented and retrieval-free LMs, but ac-

tually among different retrievers[1] (e.g. Figure 1). We call this observed phenomenon as **retriever inconsistency**. Specifically, we use open-domain question answering (ODQA;Chen et al., 2017) as our benchmark task and build 15 different retrievers by retrieving from different knowledge sources (search engine, Wikipedia dump and parametric knowledge) and implementing diverse processing methods (truncation, concatenation, reranking and compression). Our experiment reveals that, on average, more than 16% of questions incorrectly answered by one retriever can be corrected by an alternative retriever, and this result holds for all testing models and datasets.

To further investigate the reasons behind retriever inconsistency, we theoretically show that RALM's degenerate behaviors on ODQA under regular conditions can be divided into four categories: Retriever Error, Extraction Error, Hallucination Error, and Lucky Guess. We further empirically investigate the example-level error occurrence behaviors of these four types of errors and the results indicate a ubiquitous inconsistent pattern across retrievers for every error type, which collectively contributes to the retriever inconsistency.

Our further analysis reveals that the innate difference in knowledge sources, such as the absence of post-2018 information in the 2018 Wiki dump or search engine's deficiency in understanding tricky queries, and the inevitable and unpredictable degenerated behavior of the reader model, such as hallucination or the weak robustness to irrelevant context, serve as the main reason of retriever inconsistency.

Inspired by our analysis, we propose Ensemble of Retrievers (EoR), a trainable framework that first samples from RALM with different retrievers and then rejects based on a voting mechanism that measures similarity between answers. Not only can EoR reduce retriever errors by adaptively retrieving from the most appropriate knowledge source, but effectively reduce errors caused by unpredictable degradation of the reader model by comparing answers from different retrievers, based on our observation of inconsistent model error behavior and the intuition that incorrect answers vary while correct answers are always similar. By introducing controlling parameters in our framework, we can easily construct an optimization problem which can be solved by a heuristic search algorithm, and automatically search for the optimal retriever pool used for sampling. Our framework is compatible with any LLM and does not require any training on them. Experiment shows that EoR can effectively improve the performance consistency compared to RALM with a single retriever, thereby improving the corpus performance on ODQA.

[1]We distinguish different retrievers by their knowledge sources and text processing methods. We consider two retrievers equal if and only if their retrieved text chunks are exactly the same for the same query. Retrieval-free could be thought of a singular retriever which reads the query and outputs an empty set.

2 Retrievers Are Inconsistent

To investigate the example-level inconsistent behavior across retrievers and the reasons behind it, we adopt the single-hot short-form ODQA task, which consists of factual questions with short and clear answers, as our benchmark task. The straightforward task format and reliable automatic evaluation metrics (Kamalloo et al., 2023) enable us to quickly and accurately evaluate the correctness of model responses and retrieved documents, facilitating in-depth theoretical and empirical analysis.

2.1 Experimental Setup

We adopt the zero-shot in-context RALM (Ram et al., 2023) which directly prepends the retrieved documents to the input query based on the prompt template (see Appendix C.2). This naive yet efficient framework have been widely used in recent works (Gao et al., 2023b). We employ Llama2-chat$_{7B,\ 13B}$ (Touvron et al., 2023) and ChatGPT[2] as our base LM and perform greedy-search on all response generations to reduce the hallucination brought by sampling and guarantee reproducibility.

Retrievers: We characterize retrievers by their individual knowledge sources and the diverse knowledge processing methods. We adopt three different knowledge sources: Search Engine (**SE**), Wikipedia (**Wiki**) and model generated parametric knowledge (**PK**). Specifically, we choose Google as our search engine and directly forward the original query to the Google Search API;[3] We implement DPR (Karpukhin et al., 2020), which use English Wikipedia dump from Dec. 20, 2018 as the documents source, to retrieve from Wikipedia; For parametric knowledge, we follow GenRead (Yu et al., 2023) to directly prompt the base LM to generate background documents to answer the

[2]gpt-3.5-turbo-instruct
[3]https://serper.dev/

query.

As for knowledge processing methods, we adopt four main operations: truncation, concatenation, reranking and compression. Truncation here particularly refer to select top-k text chunks from sorted text list,[4] denoted by "**@k**"; Concatenation here specifically refers to concatenate text from different sources, denoted by "**&**". Particularly, we use Hybrid (**HB**) to represent the concatenation of text chunks from all three original knowledge sources; For reranking (denoted by "**@RR**"), We adopt WebGLM's (Liu et al., 2023b) reranking model, a Contriever (Izacard et al., 2022) model re-trained with model extracted data. It is reported with better performance than vanilla Contriever; In the case of Compression (denoted by "**@CP**"), we directly prompt the base LM to summarize the input text, as LLM have demonstrated notable capabilities in extracting information (Yang et al., 2023; Liu et al., 2023b).

It is intractable to exhaust all retriever combinations, hence we manually design eight typical retrievers (we try to keep their output documents in similar length) and the compressed version of them except for parametric knowledge,[5] for a total of 15 retrievers. We also regard retrieval-free (denoted by **ReFree**) as a special singular retriever. Full retrievers list and processing details could be found in Appendix C.1. We use the combination of operation abbreviations to represent the retriever, the operation priority order follows **@RR >@k>& >@CP**. For example, SE@RR@5&Wiki@5 stands for first reranking the search engine results and then concatenating the top-5 reranked search engine text chunks and the top-5 wiki chunks.

Datasets: We experiment on three English ODQA datasets: Natural Questions (NQ; Kwiatkowski et al., 2019), Web Questions (WebQ; Berant et al., 2013) and TriviaQA (Joshi et al., 2017), details refer to Appendix C.3. We evaluate Llama2-chat$_{7B,\ 13B}$ on the full validation split,[6] whereas for ChatGPT, we randomly sample 500 questions from each split for evaluation because of budget limitation.

Evaluation Metrics: We need two kinds of metrics, one to evaluate the correctness of answers and one to evaluate the example-level inconsistency. Following Kamalloo et al. (2023), we adopt BEM score (Bulian et al., 2022), a semantic similarity metric specifically developed for QA tasks, to evaluate QA accuracy with threshold 0.8.[7] It is reported to have good correlation with humans and cope with syntactical variation of answers.

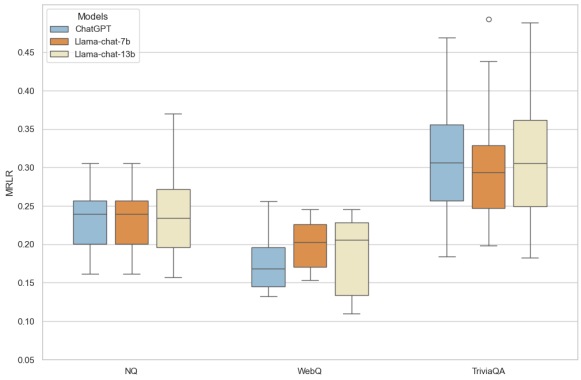

Figure 2: Boxplot displaying the distribution of MRLR of 15 different retrievers across different dataset and models.

As for measuring example-level inconsistency, we propose two naive metrics: Mean Relative Win Ratio (MRWR) and Mean Relative Lose Ratio (MRLR). Assuming we have M different retrievers $\mathcal{R} = \{r_1, r_2, ..., r_M\}$ and a dataset with N samples $\mathcal{D} = \{< q_n, a_n >\}_{n=1}^{N}$. For retriever r_m, we can evaluate the correctness of model response for each sample $s_n =< q_n, a_n >$, denoted by $\mathbf{I}^m(n) = 1$ if r_m answers correctly on sample s_n otherwise 0. Then we can calculate the Relative Win Ratio (RWR) of retriever r_i over another retriever r_j, which is defined as:

$$\mathrm{RWR}(i,j) = \frac{\sum_{n=1}^{N} \mathbf{I}^i(n) * (1 - \mathbf{I}^j(n))}{\sum_{n=1}^{N} 1 - \mathbf{I}^j(n)} \quad (1)$$

Clearly, $\mathrm{RWR}(i,j)$ represents the proportion of questions answered incorrectly by retriever r_j that were correctly answered by retriever r_i. The MRWR and MRLR are calculated by respectively averaging RWR across rows and columns:

$$\mathrm{MRWR}(i) = \frac{1}{M-1} \sum_{j \neq i} \mathrm{RWR}(i,j) \quad (2)$$

$$\mathrm{MRLR}(i) = \frac{1}{M-1} \sum_{j \neq i} \mathrm{RWR}(j,i) \quad (3)$$

MRLR and MRWR represent the degree of retriever inconsistency. Particularly, MRLR equals

[4]Some knowledge sources embrace innate ranking property such as google and DPR.

[5]Model generated documents are generally in short length, hence we do not compress them.

[6]The validation set of WebQ contains only 300 questions, hence we use the train split instead.

[7]We also tried Exact Match, the results are similar.

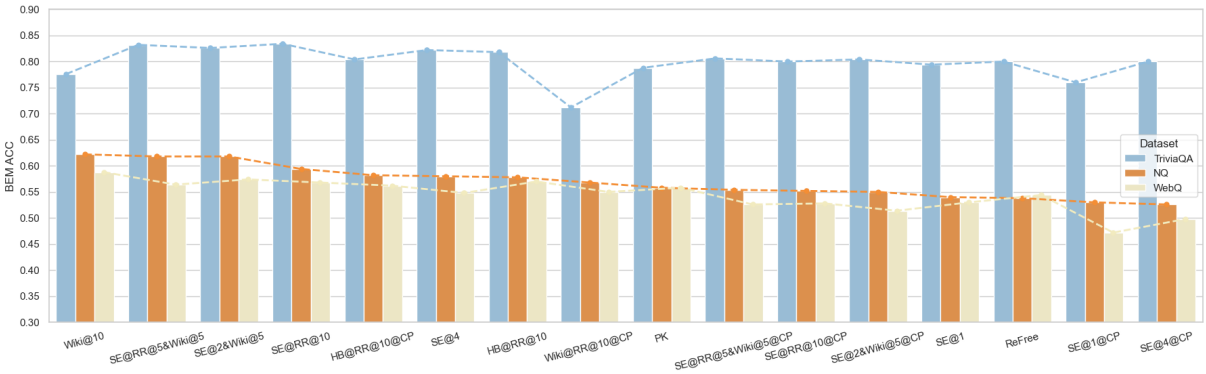

Figure 3: Corpus-level performance of different retrievers evaluated by BEM Accuracy on different datasets with ChatGPT as base LM. The order of retrievers is sorted by performance on NQ.

zero represents retriever r_i consistently outperforms all other retrievers.

2.2 Experimental Results

Figure 1 shows the RWR between different retrievers on NQ with ChatGPT as the base LM. We can observe significant inconsistency between any two different retrievers. Even for the top-performing retriever, Wiki@10 (62.2% BEM Acc), 27% of its failed questions can be correctly answered by the losest-ranked retriever, SE@1 (54.0% BEM Acc). In Figure 2, we compare the MRLR of all 15 retrievers across different datasets and models. The result indicates that, on average, more than 16% of questions incorrectly answered by one retriever can be addressed by an alternative retriever. This phenomenon is prevalent across different base models and datasets, and no evident pattern is observed that larger model can alleviate this phenomenon. The example-level inconsistency also results in the corpus-level performance inconsistency, see Figure 3, the performance curve of different retrievers on TriviaQA and WebQ do not consistently show a monotonic trend as the sorted NQ curve.

3 Why Dose Retriever Inconsistency Happen?

Before delving into the reasons behind retriever inconsistency, we firstly formulate the single-hop short-form ODQA problem and the RALM model. Let $q \in \mathcal{Q}$ denote the factoid question from the ODQA task and $a \in \mathcal{A}$ denote an answer to q (can be correct or wrong). We use $\mathcal{A}_q(a) \subset \mathcal{A}$ to denote the semantic equivalence class of the answer a to the question q and $\mathcal{A}_q^* \subset \mathcal{A}$ for the set of all correct answers to q. We assume the existence and uniqueness of $\mathcal{A}_q^*$ for simplicity.[8]

We consider a vanilla RALM $\mathcal{M}$ consisting of a probabilistic Retriever $\mathcal{R}$ and a probabilistic Reader $\mathcal{G}$. The Retriever $\mathcal{R}$ takes in a query q and return a text string $d \sim \mathcal{R}(q) \in \mathcal{P}(\mathcal{D})$. We call d document and use $\mathcal{D}$ to denote the set of all available documents, $\mathcal{P}(\mathcal{D})$ to denote the set of all probability measures over $\mathcal{D}$. The Reader $\mathcal{G}$ reads the document d and generate a response $y \sim \mathcal{G}(q, d) \in \mathcal{P}(\mathcal{A})$ based on query q.[9] We use random variable $\mathcal{M}(q)$ to represent the answer generated by the RALM and the event, RALM correctly answers the query q, can be formally written by $\mathcal{M}(q) \in \mathcal{A}_q^*$. We use $\mathcal{A}_q(a) \in d$ to denote that the document d contains a syntactical variation of the answer a for query q, then we define $\mathcal{D}_q^* = \{d \,|\mathcal{A}_q^* \in d\}$, i.e. the set of documents that contains the correct answer for q.

3.1 Error Decomposition

We now proceed to introduce three key errors contributing to the failures of the RALM.

Retriever Error E_r: Given a query q and a retriever $\mathcal{R}$, Retriever Error, denoted by E_r, represents the circumstance in which the document returned by Retriever $\mathcal{R}$ does not contain the ground-truth answer for a query q, formally defined by:

$$E_r(q, \mathcal{R}) := \{d \notin \mathcal{D}_q^* \,, \text{given } d \sim \mathcal{R}(q)\}$$

Hallucination Error E_h: Given a query q, a document d and a Reader $\mathcal{G}$, Hallucination Error, denoted by E_e, stands for the case where the

[8] Existence means $\mathcal{A}_q^*$ is not empty and Uniqueness means that for any $a \in \mathcal{A}_q^*, \mathcal{A}(a) = \mathcal{A}_q^*$. This assumption is reasonable for single-hop short-form ODQA because of its simple task format.

[9] For deterministic retriever and reader, $\mathcal{R}(q)$ and $\mathcal{G}(q, d)$ collapse to the one-point distribution.

Reader $\mathcal{G}$ generates an answer y that is not present in the document d, i.e.

$$E_h(q, d, \mathcal{G}) := \{\mathcal{A}(y) \notin d, \text{given } y \sim \mathcal{G}(q, d)\}$$

Extraction Error E_e: Given a query q, a document d and a Reader $\mathcal{G}$, Extraction Error, denoted by E_e, stands for the situation where the Reader $\mathcal{G}$ extracts the wrong portion from a correctly retrieved document, formally defined by:

$$E_e(q, d, \mathcal{G}) := \{y \notin \mathcal{A}_q^* \; and \; \mathcal{A}(y) \in d, \\ \text{given } d \in \mathcal{D}_q^*, y \sim \mathcal{G}(q, d)\}$$

The probability that these three errors occur for given q and RALM $\mathcal{M}$ can be written by:

$$\mathcal{E}_r^{q,\mathcal{R}} := \mathbb{P}_{d \sim R(q)}(d \notin \mathcal{D}_q^*)$$
$$\mathcal{E}_h^{q,\mathcal{G}}(d) := \mathbb{P}_{y \sim \mathcal{G}(q,d)}(\mathcal{A}(y) \notin d \,|\, d)$$
$$\mathcal{E}_e^{q,\mathcal{G}}(d) := \mathbb{P}_{y \sim \mathcal{G}(q,d)}(y \notin \mathcal{A}_q^*, \mathcal{A}(y) \in d \,|\, d \in \mathcal{D}_q^*)$$

Following above definition, we are able to show that the probability of RALM $\mathcal{M}$ failing on the query q, $\mathcal{E}_{\mathcal{M}}(q) := \mathbb{P}(\mathcal{M}(q) \notin \mathcal{A}_q^*)$, can be decomposed into:[10]

$$\mathcal{E}_{\mathcal{M}}(q) = \mathbb{E}_{d \sim \mathcal{R}(q)} \Big[\mathbf{I}_{\{d \in \mathcal{D}_q^*\}} \cdot \big(\mathcal{E}_h^{q,\mathcal{G}}(d) + \mathcal{E}_e^{q,\mathcal{G}}(d)\big) + \\ \mathbf{I}_{\{d \notin \mathcal{D}_q^*\}} \cdot \big(1 - \mathcal{E}_{luck}^{q,\mathcal{G}}(d) \cdot \mathcal{E}_h^{q,\mathcal{G}}(d)\big) \Big] \quad (4)$$

where $\mathcal{E}_{luck}^{q,\mathcal{G}} := \mathbb{P}(y \in \mathcal{A}_q^* \,|\, \mathcal{A}(y) \notin d, d \notin \mathcal{D}_q^*)$ represents the probability that $\mathcal{M}$ luckily 'hallucinate' the correct answer given an incorrect retrieved document, we call this event **Lucky Guess** and denote it by E_{luck}. Details of the derivation can be found in Appendix A.

3.2 Retriever Inconsistency Stems From Irregular Error Patterns

As shown in equation 4, RALM's failure on the single-hop short-form ODQA problem can be fully described by three errors, Retriever Error E_r, Hallucination Error E_h and Extraction Error E_e, and a special scenario, Lucky Guess E_{luck}. As a result, irregular example-level occurrence of any of these four types of errors[11] will contribute to the inconsistent behavior of the whole RALM. To quantitatively measure the irregular pattern of each error

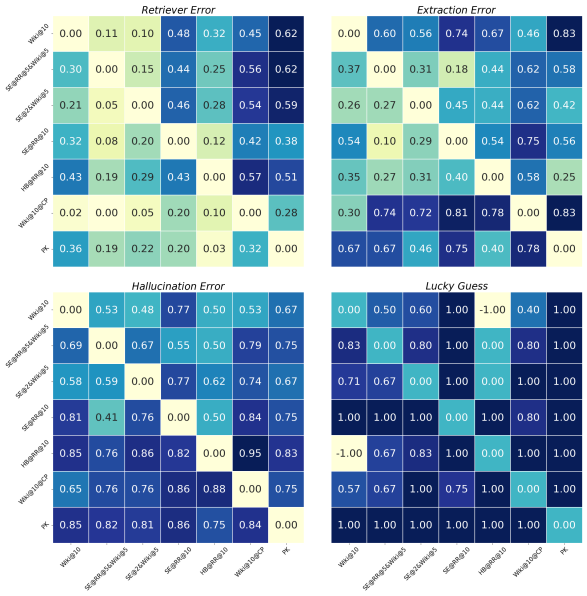

Figure 4: Error Relative Win Ratio between different Retrievers with ChatGPT as base LM, evaluated on NQ validation set. 0 represents the column retriever consistently outperforms the row retriever concerning error occurrence and -1 means at least one of the retrievers is free of this error. We only show part of the result because of space limitation, but the finding is same, more graphs please refer to Appendix D.

across different retrievers, we follow the similar definition of Relative Win Ratio in section 2.1 to define the RWR for error E, RWR_E, as:

$$\text{RWR}_E(i, j) = \frac{\sum_{n=1}^{N} (1 - \mathbf{I}_E^i(n)) * \mathbf{I}_E^j(n)}{\sum_{n=1}^{N} \mathbf{I}_E^j(n)} \quad (5)$$

where $\mathbf{I}_E^i(n) = 1$ if error E occurs for sample s_n and retriever r_i, more details refer to Appendix B. Therefore, $\text{RWR}_E(i, j)$ represents the proportion of Retriever Errors made by retriever r_j that are avoided by r_i, and $\text{RWR}_E(i, j) = 0$ implies that r_j consistently outperform r_i.

In Figure 4, we show the results of different errors, where the retrievers are sorted in descending order by their corpus-level performance (Figure 3). For Retriever Error, we observe significant bidirectional RWR_{E_r} between retrievers with different sources (such as 0.62/0.36 for Wiki@10 versus PK) which indicates the innate differences of knowledge sources serve as a main reason for the inconsistency of retrieval errors. As a result, retrievers with hybrid sources (containing concatenation operation "**&**") witnessed more consistent behaviors over other retrieves, see the low column RWR_{E_r} values of them, although still suffer from a small

[10] For deterministic RALM, the decomposition can be written more concisely: $\mathbb{P}(\mathcal{M}(q) \notin \mathcal{A}_q^*) = (1 - \mathcal{E}_r)(\mathcal{E}_h + \mathcal{E}_e) + \mathcal{E}_r(1 - \mathcal{E}_{luck}\mathcal{E}_h)$

[11] We still use the term 'error' to represent the Lucky Guess event for simplicity.

portion of unpredictable errors caused by different processing methods.

As for Extraction Error, we observe a widespread inconsistency across all retrievers. In particular, even SE@RR@5&Wiki@5 and SE@2&wiki@5, which share a large portion of contents and both contain the correct answer,[12] obtain non-neglectable bidirectional RWR_{E_e} (0.31/0.27). We believe these ubiquitous inconsistent behaviors stem from Reader $\mathcal{G}$'s weak robustness to long and irrelevant contexts (Liu et al., 2023a; Shi et al., 2023). A similar phenomenon is observed in Hallucination Error, but with more severe randomness. Kalai and Vempala (2023) demonstrate that hallucination is inevitable for a statistical reason.

Therefore, the inconsistent occurrence patterns of three errors collectively contribute to unpredictable RALM's degeneration. Furthermore, Lucky Guess, which compensate the mistakes made by retriever error, also demonstrate an inconsistent behavior and the irregular occurrence will further exacerbating retriever inconsistency.

4 Ensemble of Retrievers

Our analysis provides a strong rationale for adopting an ensemble of retrievers, which can retrieve from different sources, and leverage the irregular error behavior to reduce the degeneration of the reader model. In fact, we can calculate the theoretical upper bound of the ensemble of retrievers by assuming a perfect voting mechanism that will always select the correct answer as long as one retriever outputs the correct one, see figure 5. It demonstrates an obvious monotonic increasing trend implying the great potential of the ensemble of retrievers. However, the variability in performance within models with the same retriever pool size underscores the importance of understanding both what to include in the ensemble and how to effectively combine them.

4.1 Our Method

We propose EoR, a trainable generate-then-rerank framework that can dynamically determine what to retrieve and how to retrieve. Formally, suppose we have M different retrievers $\mathcal{R} = \{r_i\}_{i=1}^{M}$ and a reader model $\mathcal{G}$. EoR accepts an input query q and first generates M responses based on different retrievers, written as $y_m = \mathcal{G}(q, r_m(q)),\ y_m \in \mathcal{A},\ m \in \{1, 2, ..., M\}$, then the voter module $S_{voter} : \mathcal{A}^M \rightarrow \mathbb{R}^M$ takes in the responses and calculates a score s_m for each response y_m, i.e. $\mathcal{S}_{voter}(y_1, y_2, ..., y_M) = [s_1, s_2, ..., s_m]$.

[12]This comes from our constructing and Extraction Error estimation methods, see Appendix B and C.1

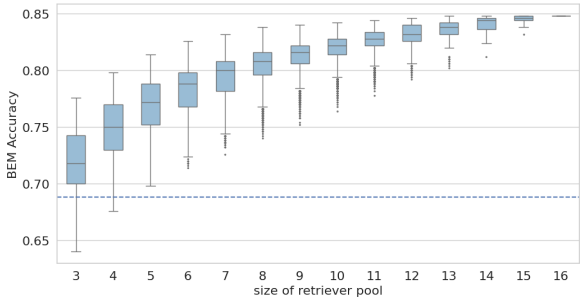

Figure 5: The upper bound of BEM Accuracy by ensembling different retrievers on NQ with ChatGPT as base LM. Each boxplot represents the distribution of the upper bound for different retriever combinations with the same pool size. The dashed line shows the best single retriever performance.

The voter module comprises two functions, similarity function $\mathcal{S}_{sim} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}$ and pooling function $\mathcal{S}_{pool} : \mathbb{R}^{M-1} \rightarrow \mathbb{R}$. The similarity function measures the semantic similarity between two responses. Specifically, we consider a weighted sum of multiple similarity measurement metrics, written as:

$$\mathcal{S}_{sim}(y_m, y_n | \omega^s) = \sum_{i}^{K} \omega_i^s \cdot sim_i(y_m, y_n) \quad (6)$$

where each $sim(\cdot, \cdot)$ represents a distinct semantic similarity metric, such as EM, BERTScore (Zhang et al., 2020), or the entailment score of a Natural Language Inference (NLI; Bowman et al., 2015) model, and K represents the total number of different metrics. The metric weight ω_s can be predefined or learned as parameters. As a result, each response y_m corresponds to $M - 1$ similarity scores, and the pooling function is responsible for compressing these scores into a single one. Typical pooling functions can be mean, maximum, plurality voting (Zhou, 2012) or majority voting (Wang et al., 2023a), detailed formula refer to Appendix C.4. Then the voter score s_m can be formally written as:

$$s_m = \omega_m^r \cdot \mathcal{S}_{pool}(\{\mathcal{S}_{sim}(y_m, y_n | \omega^s)\}_{n \neq m}) \quad (7)$$

where ω^r is preset or trainable parameters representing the confidence in different retrievers. The final response is selected with the highest score:

$$\mathcal{M}_{\text{EoR}}(q) = y_{m^*}$$

where $m^* = \arg\max_m \mathcal{S}_{voter}(\{y_m\} \,|\, \omega^s, \omega^r)$

4.2 Ensemble by Learning

Like the Stacking methods (Wolpert, 1992), we can use our EoR model to generate data to train the controlling parameters ω^s and ω^r. Specifically, assuming we have a training dataset $\mathcal{D}_{train} = \{< q_i, a_i >\}_{i=1}^N$. We pass each query q_i through our model $\mathcal{M}_{\text{EoR}}$ and get the corresponding response y_m^i for each retriever and the similarity score $sim_k(y_m^i, y_n^i)$ for each answer pair y_m^i, y_n^i under the k-th similarity metric. We can further calculate the correctness of each response y_m^i to the ground truth answer a_i with some evaluation metric g, written as $g(y_m^i, a_i)$. The evaluation metric g can be EM, BEM, or even human annotation.

Finding the optimal ω^s and ω^r is equivalent to solve the following optimization problem:

$$max \quad \frac{1}{N}\sum_{i=1}^{N} g(y_{m_i^*}, a_i)$$
$$s.t. \quad m_i^* = \arg\max_m \mathcal{S}_{voter}(\{y_m^i\} \,|\, \omega^s, \omega^r)$$

For given ω^s and ω^r, $\mathcal{S}_{voter}(\{y_m^i\} \,|\, \omega^s, \omega^r)$ can be quickly evaluated by equation 6 and 7. Therefore, we can solve this problem with a heuristic search algorithm, which searches the feasible region by continuously evaluating the objective function. To conduct automatic retriever selection, we can simply replace ω_m^r with $\omega_m^r \cdot \mathbf{I}_{\omega_m^r > t}$, where t is a hyperparameter. During searching, retrievers with small weights will be directly ignored.

4.3 Experimental Setting

Same as section 2.1, we use NQ, WebQ and TriviaQA as our experiment datasets and Llama2-chat$_{7B,\,13B}$ and ChatGPT as our base LM. We search parameters on the validation split (train split for WebQ) and report performance on the test split. For ChatGPT, we randomly sample 500 questions from each split same as Section 2.1. We use BEM accuracy, MRWR and MRLR as evaluation metrics.

For EoR, we use the 15 retrievers introduced in section 2.1 and ReFree, in a total of 16 retrievers, as our initial retriever pool. We choose EM, BertScore, and NLI[13] as our base similarity metrics in equation 6, and mean pooling for the pooling function.[14] We choose the Nelder-Mead method as the heuristic search method and implement it with SciPy. An upper bound of 0.6 is set for ω^s and ω^r to prevent overreliance on a single retriever. We set ω^r filtering threshold t to 0.1.

[13] https://huggingface.co/microsoft/deberta-xlarge-mnli

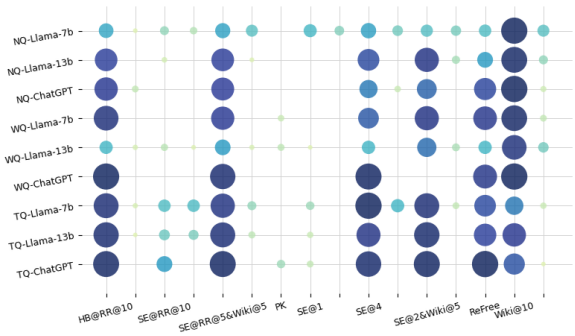

Figure 6: Visualization of Retriever weight ω^r learned with different base LM and datasets. Each row represents the weights from the same EoR model. A larger circle with darker color implies a higher weight on the corresponding retriever.

4.4 Results and Analysis

EoR exhibits more consistent behavior than single retriever models. Table 1 presents the results. EoR has a large reduction in MRLR compared to the best-performed single retriever model across all datasets and models except for one exception, which also results in a general corpus-level performance increase. The only exception is Llama2-chat$_{13B}$'s performance on WebQ, which results from the discrepancy in retriever performance between the train and test set, see Table 2. EoR trained on the train set is prone to rely on Wiki@10 which suffers from a significant performance drop in the test set.

EoR adaptively learns what to retrieve. Figure 6 visualizes the retriever weights learned by our training methods. Almost every row demonstrates a sparse weight distribution and the remaining retrievers with large weights all performed well on the corresponding training dataset, which implies that EoR can effectively filter out redundant or unreliable retrievers. We can also observe that EoR with Llama-13b trained with WebQ indeed puts most of its weight on Wiki@10, while EoR with Llama-7b and ChatGPT overcome the performance deduction brought by distribution shift by spreading the weight to more retrievers.

[14] Actually we can search for the optimal pooling methods by transforming to a categorical variable, but our preliminary experiments found that mean pooling is effective enough.

Base Models	$\mathcal{R}$	NQ			WebQ			TriviaQA		
		BEM	MRWR	MRLR	BEM	MRWR	MRLR	BEM	MRWR	MRLR
Llama2-chat$_{7B}$	ReFree	34.35	17.51	23.64	51.82	25.03	24.47	55.57	22.31	48.84
	Top $\mathcal{R}$	55.57	29.61	16.33	56.25	24.33	15.66	77.34	38.63	20.11
	EoR	**58.92**	**31.01**	**12.36**	**59.45**	**25.15**	**10.66**	**80.62**	**40.08**	**10.62**
Llama2-chat$_{13B}$	ReFree	46.43	23.83	32.84	58.76	**26.56**	19.44	66.36	27.94	47.92
	Top $\mathcal{R}$	62.30	**33.43**	15.40	**62.20**	26.33	**11.48**	82.37	**41.51**	18.69
	EoR	**64.24**	33.15	**12.05**	60.63	23.56	12.86	**83.80**	40.10	**11.68**
ChatGPT	ReFree	52.40	21.85	25.46	59.20	23.22	20.69	81.60	38.35	37.90
	Top $\mathcal{R}$	60.20	27.33	16.15	61.00	**24.91**	18.63	84.40	35.49	23.08
	EoR	**63.00**	**29.17**	**12.97**	**61.60**	20.87	**13.38**	**87.40**	**39.68**	**12.00**

Table 1: Main results on the test split of NQ, WebQ and TriviaQA. Top $\mathcal{R}$ represents the best-performed single retrieval model on the corresponding test set. **Bold** number indicates the best performance across retrievers with the same base model and test set.

Retrievers	Train	Test
Wiki@10	**62.10**	58.17
SE@2&Wiki@5	61.90	60.97
SE@RR@5&Wiki@5	61.24	**62.20**
EoR	63.02	60.63

Table 2: comparison of Llama2-chat$_{13B}$'s performance with different retrievers on WebQ. **Bold** number represents the best-performed single retriever on the corresponding split.

5 Related Work

Degeneration of RALMs: Many works have empirically demonstrated that retrieval-augmentation sometimes hurts LM's performance (Ren et al., 2023; Mallen et al., 2023; Yoran et al., 2023). Some works attribute these failures to incorrect retrieval (Mallen et al., 2023; Chen et al., 2022), while some blame the degeneration on the weak robustness of LM on irrelevant or long context (Ren et al., 2023; Li et al., 2023; Gao et al., 2023a). Particularly, Ren et al. (2023) found that a large portion of failure cases of ChatGPT come from extracting wrong answers from the supporting documents. Li et al. (2023) claimed that models are prone to ignore noisy context, although sometimes they can generate correct answer with their parametric knowledge. Liu et al. (2023a) showed that LM's performance is vulnerable to the position of relevant information in a long context.

Relieving RALMs's Degeneration: Some works focus on improving retrieval recall (Izacard et al., 2022; Trivedi et al., 2023; Ma et al., 2023), while some works try to increase RALM's robustness on retrieved context by increasing reader model's ability to leverage context (Yoran et al., 2023; Liu et al., 2023b) or increasing retriever precision (Xu et al., 2023; Liu et al., 2023b). There are also some works that try to solve this problem from a system perspective, such as using rejection sampling on the reader side to reduce the hallucination (Menick et al., 2022; Asai et al., 2023), or dynamically deciding whether to retrieve (Mallen et al., 2023; Yoran et al., 2023; Asai et al., 2023; Wang et al., 2023b). However, all these methods focus on improving the performance of a single retriever-augmented LM, which makes most of them compatible with our EoR framework as long as they can be fitted into a retriever or a reader.

6 Conclusion

This work focus on RALM's performance inconsistency across different retrievers. Our results show that retriever inconsistency is ubiquitous across different retrievers and base models. To investigate the reasons behind it, we theoretically decompose RALM's failure into four categories, which serve as a basis for analyzing the degeneration behavior of RALM. Our experiments based on our decomposition reveal that innate differences in knowledge sources and the unpredictable degeneration of the reader model are the main causes for the inconsistency behavior. We further propose Ensemble of Retrievers (EoR), a trainable framework compatible with any LLMs and retrievers. Our experiments demonstrate that EoR can effectively boost RALM's performance by adaptively retrieving from multiple knowledge sources and reducing irregular errors made by the reader model.

7 Limitation

Our theoretical derivation and empirical analysis are all based on the naive singe-hop short-from ODQA setting, whereas multi-hop reasoning and long answer questions are quite common in real-world. We do not choose the task under more complex scenarios because of the absence of reliable automatic evaluation metrics and costly human annotation fees, and we also believe that our conclusion and methods can be generalized to more complex cases. This is because the retriever inconsistency comes from the irregular error occurrence in RALM and intuitively more complex scenario implies weaker robustness. Hence we hypothesize that complex questions will not only exacerbate the errors introduced in section 3.1 but introduce new errors such as reasoning errors, which will result in more severe inconsistent behaviors. But this need further human evaluation to verify. Our EoR framework can also work with more complex questions by using reliable similarity metrics in the voting mechanism and learning by human-annotated data. Example-wise scoring functions (Asai et al., 2023) can also be combined with the voter module by some trainable parameters.

Another concern is the computational cost of ensembling multiple retrievers. In fact, EoR can leverage the batch inference of LLMs because of the sharing of the reader model. Hence compared with sampling on the reader side such as self-consistency(Wang et al., 2023a), the additional computational cost mainly comes from multiple retrievers and the answer similarity calculation with smaller models. If we do not use parametric knowledge from LLMs, most of them are computationally efficient compared to LLMs and can be easily parallelized.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *CoRR*, abs/2310.11511.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *CoRR*, abs/2302.04023.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 291–305. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Hung-Ting Chen, Michael J. Q. Zhang, and Eunsol Choi. 2022. Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2292–2307. Association for Computational Linguistics.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Trans. Assoc. Comput. Linguistics*, 10:257–273.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023a. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 6465–6488. Association for Computational Linguistics.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Adam Tauman Kalai and Santosh S. Vempala. 2023. Calibrated language models must hallucinate. *CoRR*, abs/2311.14648.

Ehsan Kamalloo, Nouha Dziri, Charles L. A. Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5591–5606. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix X. Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1774–1793. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *CoRR*, abs/2307.03172.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023b. Webglm: Towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 4549–4560. ACM.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *CoRR*, abs/2305.14283.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, H. Francis Song, Martin J. Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes. *CoRR*, abs/2203.11147.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *CoRR*, abs/2302.00083.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *CoRR*, abs/2307.11019.

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, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu,

Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation. *CoRR*, abs/2310.03214.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023b. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10303–10315. Association for Computational Linguistics.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: improving retrieval-augmented lms with compression and selective augmentation. *CoRR*, abs/2310.04408.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. *CoRR*, abs/2302.08081.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *CoRR*, abs/2310.01558.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.

A Error Decomposition Derivation

Given a query q and a RALM model $\mathcal{M}(q) = \mathcal{G}(q, \mathcal{R}(q))$. We consider the probabilistic retriever model $\mathcal{R}(\cdot) : \mathcal{Q} \rightarrow \mathcal{P}(\mathcal{D})$, $\mathcal{P}(\mathcal{D})$ denote the set of all probability measures over the set of all documents $\mathcal{D}$, and the probabilistic reader model $\mathcal{G}(\cdot,\cdot) : \mathcal{Q} \times \mathcal{D} \rightarrow \mathcal{P}(\mathcal{A})$. $\mathcal{A}$ denote the set of all possible answers. We use $\mathcal{A}_q^*$ to denote the set of all correct answers to q and $\mathcal{A}(a)$ to denote the semantic equivalent class of a. We assume the uniqueness of the correct answers in the case of equivalent class, i.e. for any $y_1, y_2 \in \mathcal{A}_q^*$, we have $\mathcal{A}(y_1) = \mathcal{A}(y_2)$, hence $\mathcal{A}_q^*$ is an equivalent class itself.

Now, let's try to decompose the probability that RALM $\mathcal{M}$ answers incorrectly given q.

$$
\begin{aligned}
&\mathbb{P}(\mathcal{M}(q) \notin \mathcal{A}_q^*) \\
=&\mathbb{E}_{d\sim\mathcal{R}(q)}\left[\mathbb{E}_{y\sim\mathcal{G}(q,d)}\left[\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}} \,\middle|\, d\right]\right] \\
=&\mathbb{E}_d\left[\mathbb{E}_y\left[\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}}\cdot\mathbf{I}_{\{d\notin\mathcal{D}_q^*\}} + \mathbf{I}_{\{y\notin\mathcal{A}_q^*\}}\cdot\mathbf{I}_{\{d\in\mathcal{D}_q^*\}} \,\middle|\, d\right]\right] \\
=&\mathbb{E}_d\left[\mathbf{I}_{\{d\in\mathcal{D}_q^*\}}\cdot\mathbb{E}_y\left[\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}} \,\middle|\, d\notin\mathcal{D}_q^*\right]\right]+ \\
&\mathbb{E}_d\left[\mathbf{I}_{\{d\notin\mathcal{D}_q^*\}}\cdot\mathbb{E}_y\left[\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}} \,\middle|\, d\in\mathcal{D}_q^*\right]\right] \qquad (8)
\end{aligned}
$$

Now, considering $\mathbb{E}_{y\sim\mathcal{G}}\left[\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}}\middle|d\notin\mathcal{D}_q^*\right]$:

$$
\begin{aligned}
&\mathbb{E}_y\left[\,\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}}\,\middle|\,d\notin\mathcal{D}_q^*\,\right]\\
=&\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,d\notin\mathcal{D}_q^*)\\
=&\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\notin d, d\notin\mathcal{D}_q^*)\cdot\mathbb{P}\left(E_h(q,d,\mathcal{G})\right)+\\
&\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\in d, d\notin\mathcal{D}_q^*)\cdot\mathbb{P}\left(E_h(q,d,\mathcal{G})\right)\\
=&\Big[1-\mathbb{P}(y\in\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\notin d, d\notin\mathcal{D}_q^*)\Big]\cdot\mathcal{E}_h(d)+\\
&1\cdot\Big(1-\mathcal{E}_h(d)\Big)\\
=&1-\mathbb{P}(y\in\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\notin d, d\notin\mathcal{D}_q^*)\cdot\mathcal{E}_h(d)\\
=&1-\mathcal{E}_{luck}(d)\cdot\mathcal{E}_h(d)
\end{aligned}
\tag{9}
$$

$\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\in d, d\notin\mathcal{D}_q^*)=1$ because if $y\in\mathcal{A}_q^*$, then $\mathcal{A}_q^*\in d$ by $\mathcal{A}(y)\in d$. This is contradicted to the definition of $d\notin\mathcal{D}_q^*$. Actually, if the document d does not contain any semantic variant of the ground-truth answer but contain a semantic variant of y, then y can not be a semantic variant of the ground-truth answer.

Similarly,

$$
\begin{aligned}
&\mathbb{E}_y\left[\,\mathbf{I}_{\{y\notin\mathcal{A}_q^*\}}\,\middle|\,d\in\mathcal{D}_q^*\,\right]\\
=&\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\notin d, d\in\mathcal{D}_q^*)\cdot\mathbb{P}\left(E_h(q,d,\mathcal{G})\right)+\\
&\mathbb{P}(y\notin\mathcal{A}_q^*,\,\mathcal{A}(y)\in d\,\big|\,d\in\mathcal{D}_q^*)\\
=&1\cdot\mathbb{P}\left(E_h(q,d,\mathcal{G})\right)+\mathbb{P}\left(E_e(q,d,\mathcal{G})\right)\\
=&\mathcal{E}_h(d)+\mathcal{E}_e(d)
\end{aligned}
\tag{10}
$$

$\mathbb{P}(y\notin\mathcal{A}_q^*\,\big|\,\mathcal{A}(y)\notin d, d\in\mathcal{D}_q^*)=1$ because if $y\in\mathcal{A}_q^*$, then $\mathcal{A}(y)=\mathcal{A}_q^*$ by the uniqueness assumption. Then $\mathcal{A}(y)\notin d\Rightarrow\mathcal{A}_q^*\notin d$ which is contradicted to the definition of $d\in\mathcal{D}_q^*$.

Combing equation 4,5 and 6, we get:

$$
\begin{aligned}
&\mathbb{P}(\mathcal{M}(q)\notin\mathcal{A}_q^*)\\
=&\mathbb{E}_d\Big[\mathbf{I}_{\{d\in\mathcal{D}_q^*\}}\cdot\big(\mathcal{E}_h(d)+\mathcal{E}_e(d)\big)\\
&+\ \mathbf{I}_{\{d\notin\mathcal{D}_q^*\}}\cdot\big(1-\mathcal{E}_{luck}(d)\cdot\mathcal{E}_h(d)\big)\Big]
\end{aligned}
\tag{11}
$$

B Experiment Setting for Error Analysis

Assuming we have a dataset with N samples $\mathcal{D}=\{<q_n,a_n>\}_{n=1}^N$ and a RALM $\mathcal{M}$ with retriever $\mathcal{R}$. For certain example $<q_n,a_n>$, we can calculate the Answer Correctness Indicator,

$$\mathbf{I}_{\mathcal{A}}^{\mathcal{M}}(n)=1\ \ if\ \ \mathcal{M}(q_n)=a_n,$$

the Retriever Error Occurrence Indicator,

$$\mathbf{I}_{E_r}^{\mathcal{M}}(n)=1\ \ if\ \ a_n\notin\mathcal{R}(q_n)$$

and the Hallucination Error Occurrence Indicator,

$$\mathbf{I}_{E_h}^{\mathcal{M}}(n)=1\ \ if\ \ \mathcal{M}(q_n)\notin\mathcal{R}(q_n)$$

for each query. We use BEM score with threshold 0.8 to evaluate the answer correctness and Exact Match to evaluate whether a retrieved document contains a certain piece of text, i.e. $a\in d$ if a normalized form of a is matched in d. We can then evaluate the occurrence indicator for Extraction Error and Lucky Guess:

$$\mathbf{I}_{E_e}^{\mathcal{M}}(n)=(1-\mathbf{I}_{E_r}^{\mathcal{M}}(n))\cdot(1-\mathbf{I}_{E_h}^{\mathcal{M}}(n))\cdot(1-\mathbf{I}_{\mathcal{A}}^{\mathcal{M}}(n))$$

$$\mathbf{I}_{E_{luck}}^{\mathcal{M}}(n)=\mathbf{I}_{E_r}^{\mathcal{M}}(n)\cdot\mathbf{I}_{E_h}^{\mathcal{M}}(n)\cdot\mathbf{I}_{\mathcal{A}}^{\mathcal{M}}(n)$$

Because of the training objective of LLM, it tends to generate long answers, which affects the EM accuracy on estimating whether a document contain an answer. Although we have tried several methods to shorten average answer length, such as special instruction, there are still long answers with unimportant content such as "Surely, the answer to the question is". Therefore, before conducting error analysis, we filter out the question with an answer longer than 5 words generated by some retriever and all answers to the same question generated by other retrievers. We denote the filtered dataset as $\mathcal{D}^*$ with size N^*.

Following equation 5, we calculate the Relative Win Ratio (RWR) for Retriever Error E_r between RALM $\mathcal{M}^i$ and $\mathcal{M}^j$ by:

$$\text{RWR}_{E_r}(i,j)=\frac{\sum_{n=1}^{N^*}(1-\mathbf{I}_{E_r}^{\mathcal{M}_i}(n))*\mathbf{I}_{E_r}^{\mathcal{M}_j}(n)}{\sum_{n=1}^{N^*}\mathbf{I}_{E_r}^{\mathcal{M}_j}(n)}$$

and the RWR for Hallucination Error E_h by:

$$\text{RWR}_{E_h}(i,j)=\frac{\sum_{n=1}^{N^*}(1-\mathbf{I}_{E_h}^{\mathcal{M}_i}(n))*\mathbf{I}_{E_h}^{\mathcal{M}_j}(n)}{\sum_{n=1}^{N^*}\mathbf{I}_{E_h}^{\mathcal{M}_j}(n)}$$

As for the Extraction Error E_e, notably it is defined under the condition that retriever returns the correct documents. Therefore, when calculating $\text{RWR}_{E_h}(i,j)$, we only consider the circumstance that both retriever in $\mathcal{M}^i$ and $\mathcal{M}^j$ return the correct documents, i.e.

$$\text{RWR}_{E_e}=\frac{\sum_{n=1}^{N^*}(1-\mathbf{I}_{E_e}^{\mathcal{M}_i}(n))*\mathbf{I}_{E_e}^{\mathcal{M}_j}(n)*\mathbf{I}_{E_e}^{*}(n)}{\sum_{n=1}^{N^*}\mathbf{I}_{E_e}^{\mathcal{M}_j}(n)*\mathbf{I}_{E_e}^{*}(n)}$$

where $\mathbf{I}_{E_e}^{*}(n)=(1-\mathbf{I}_{E_r}^{\mathcal{M}_i}(n))*(1-\mathbf{I}_{E_r}^{\mathcal{M}_j}(n))$. Similarly, $\text{RWR}_{E_{luck}}$ is defined by:

$$\frac{\sum_{n=1}^{N^*}(1-\mathbf{I}_{E_{luck}}^{\mathcal{M}_i}(n))*\mathbf{I}_{E_{luck}}^{\mathcal{M}_j}(n)*\mathbf{I}_{E_{luck}}^{*}(n)}{\sum_{n=1}^{N^*}\mathbf{I}_{E_{luck}}^{\mathcal{M}_j}(n)*\mathbf{I}_{E_{luck}}^{*}(n)}$$

where $\mathbf{I}^*_{E_{luck}}(n) = \mathbf{I}^{\mathcal{M}^i}_{E_r}(n) * \mathbf{I}^{\mathcal{M}^i}_{E_h}(n) * \mathbf{I}^{\mathcal{M}^j}_{E_r}(n) * \mathbf{I}^{\mathcal{M}^j}_{E_h}(n)$.

C Implementation Details

C.1 Implementation Details of Retrievers

Following is the full list of 15 retrievers and corresponding processing methods:

Wiki@10: We directly select the top-10 passages returned from DPR (implement with pyserini) and concatenate them. Each passage has an average length of 100 words.

SE@1: We fetch and extract contents from the top-1 URL returned by Google API and them truncate it to 1000 words.

SE@4: We fetch and extract contents from the top-4 URLs returned by Google API. For the content in each URL, we select 250 words from the beginning including the title. Then concatenate them.

PK: We prompt the base LM to generate background documents to answer the query.

SE@RR@10: We fetch and extract contents from all URLs returned by Google API. Then we split all contents into chunks with an average of 100 words. We then use the reranking module introduced, a retrained contriever model from WebGLM, to encode the query and each chunk, and select the top-10 chunks with highest cosines similarity.

SE@2&Wiki@5: We concatenate the top-2 chunks from SE@4 and top-5 chunks from Wiki@10.

SE@RR@5&Wiki@5: We concatenate the top-5 chunks from SE@RR@10 and top-5 chunks from Wiki@10.

HB@RR@10: Similar to SE@RR@10, we fetch and extract contents from all URLs returned by Google API and then split all contents as long as the document returned by PK into chunks with an average of 100 words. We then put them with the top-20 chunks returned by DPR and use reranking module to rerank all chunks. We select the top-10 as the final result.

Wiki@10@CP: We summarize Wiki@10 with our compression model.

SE@1@CP: We summarize SE@1 with our compression model.

SE@4@CP: We summarize SE@4 with our compression model.

SE@RR@10@CP: We summarize SE@RR@10 with our compression model.

SE@2&Wiki@5@CP: We summarize SE@2&Wiki@5 with our compression model.

SE@RR@5&Wiki@5@CP: We summarize SE@RR@5&Wiki@5 with our compression model.

HB@RR@10@CP: We summarize HB@RR@10 with our compression model.

C.2 Prompt Templates

Template for generating Parametric Knoweldge (PK):
we use {query} to represent the placeholder for inserting the corresponding query. This template is following (Yu et al., 2023).

> Generate a background document to answer the given question.
> {query}.

Template for Compression (@CP):
we use {query} to represent the placeholder for inserting the corresponding query and {document} for the document to be compressed.

> Please truthfully summarize the document below, the summary should contain the most important information relevant to answer the query and be within 200 words:
> query: {query}
>
> document: {document}
>
> summary:

Template for Retrieval-free Generation:
we use {query} to represent the placeholder for inserting the corresponding query. We manually examine many different templates and select the one with highest average validation set performance with our automatic evaluation metrics.
Template for ChatGPT:

> Please directly answer the following question within 15 words:
> {query}

Template for Llama2-chat, inspired by (Ren et al., 2023):

> Please directly answer the following question with one or few words:
> {query}

Template for Retrieval-Augmented Generation:
we use {query} to represent the placeholder for inserting the corresponding query and {document} for the document returned by retriever.
Template for ChatGPT:

> Assuming the following paragraphs are true:
>
> {document}
>
> Please directly answer the following question within 15 words:
> {query}

Template for Llama2-chat:

> Assuming the following paragraphs are true:
>
> {document}
>
> Please directly answer the following question with one or few words:
> {query}

C.3 Datasets

NQ (Kwiatkowski et al., 2019) consists of questions collected from real Google search queries and the answers are extracted from Wikipedia by humans.
WebQ (Berant et al., 2013) contains questions collected from the Google Suggest API and answers collected by AMT workers based on Freebase.
TriviaQA (Joshi et al., 2017) contains question-answer pairs from several trivia and quiz-league websites.
We use the same dataset splits as GenRead (Yu et al., 2023). They unify the formats of all three datasets and the datasets can be download from this URL. Dataset statistics can be found in Table 3.

C.4 Pooling Functions

Assuming we have N similarity scores $s_1, s_2, ..., s_N$, then the pooling functions $\mathcal{S}_{pool} : \mathbb{R}^N \rightarrow \mathbb{R}$ are defined as follows:
Mean Pooling:

$$\mathcal{S}_{pool}(s_1, s_2, ..., s_N) = \frac{1}{N}\sum_{i=1}^{N} s_i$$

Max Pooling:

$$\mathcal{S}_{pool}(s_1, s_2, ..., s_N) = max\{s_1, s_2, ..., s_N\}$$

Datasets	Train	Valid	Test
NQ	79,168	8,757	3,610
WebQ	3,478	300	2,032
TriviaQA	78,785	8,837	11,313

Table 3: Dataset statistics

Majority Voting:

$$\mathcal{S}_{pool}(s_1, s_2, ..., s_N) = \mathbf{I}_{\left\{\sum_i^N \mathbf{I}_{\{s_i > s\}} \geq \frac{N}{2}\right\}}$$

where s is the threshold to identify semantic equivalent answers. The majority voting pooling filters out all answers with the number of semantic equivalent answers less than $\frac{N}{2}$
Plurality Voting: Assume we have M answers, for the i-th answer, we have calculated M-1 similarity scores $s_{i,1}, s_{i,2}, ..., s_{i,M-1}$. We denote $c_i^* = \sum_j^{M-1} \mathbf{I}_{\{s_{i,j} > s\}}$ which represent the estimated number of semantic equivalent answers given above similarity scores. Then the plurality voting pooling score for the i-th answer is given by:

$$\mathcal{S}_{pool}(s_{i,1}, s_{i,2}, ..., s_{i,M-1}) = \mathbf{I}_{\left\{c_i^* = max_k\ c_k^*\right\}}$$

D Complete Figures

Figure 7, 8 and 9 show the Error RWR between all retrievers on different combination of datasets and models. We can see the findings are same as what we discussed in section 3.2.

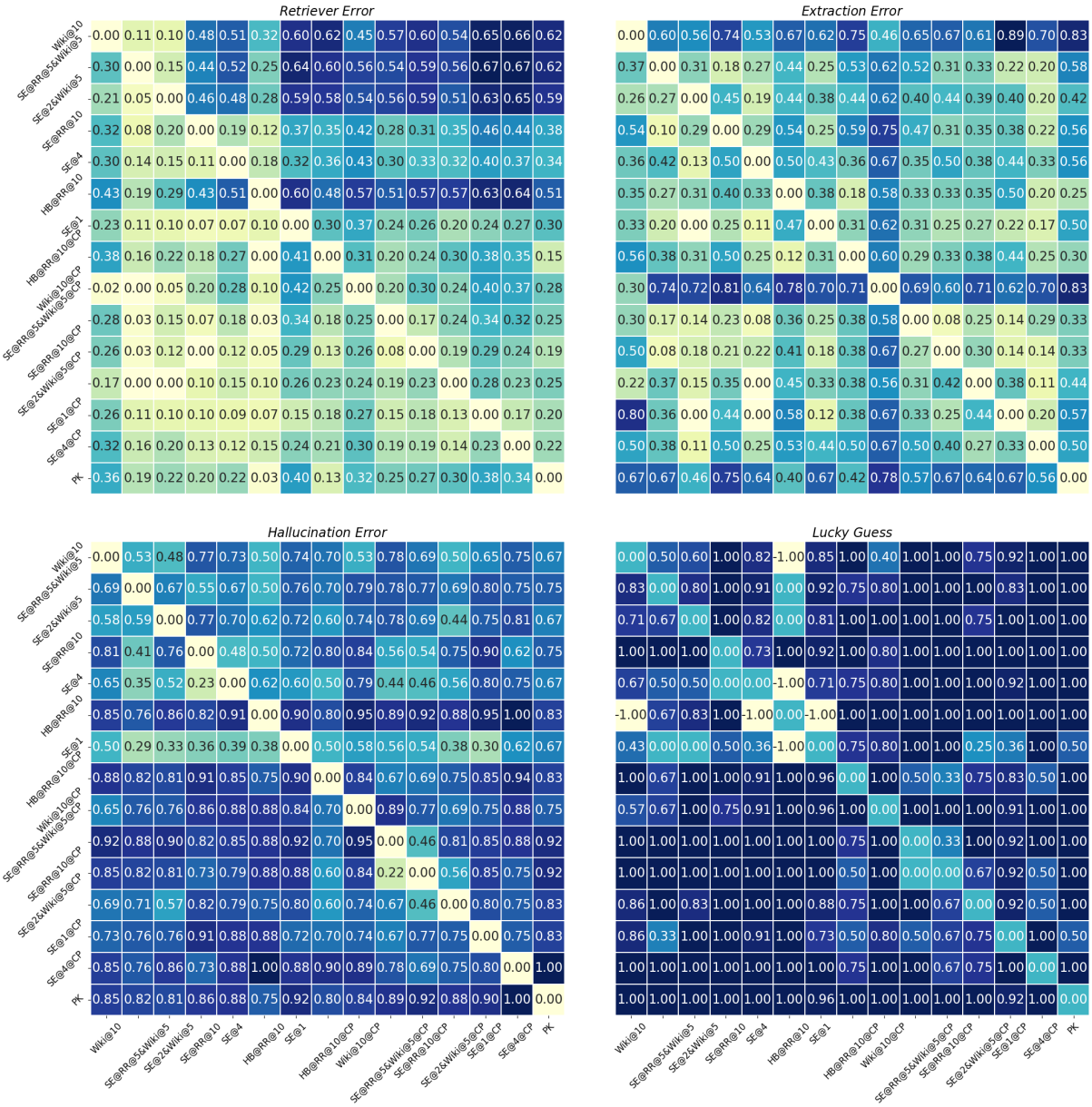

Figure 7: Full Error Relative Win Ratio between different Retrievers with ChatGPT as base LM, evaluated on NQ validation set.

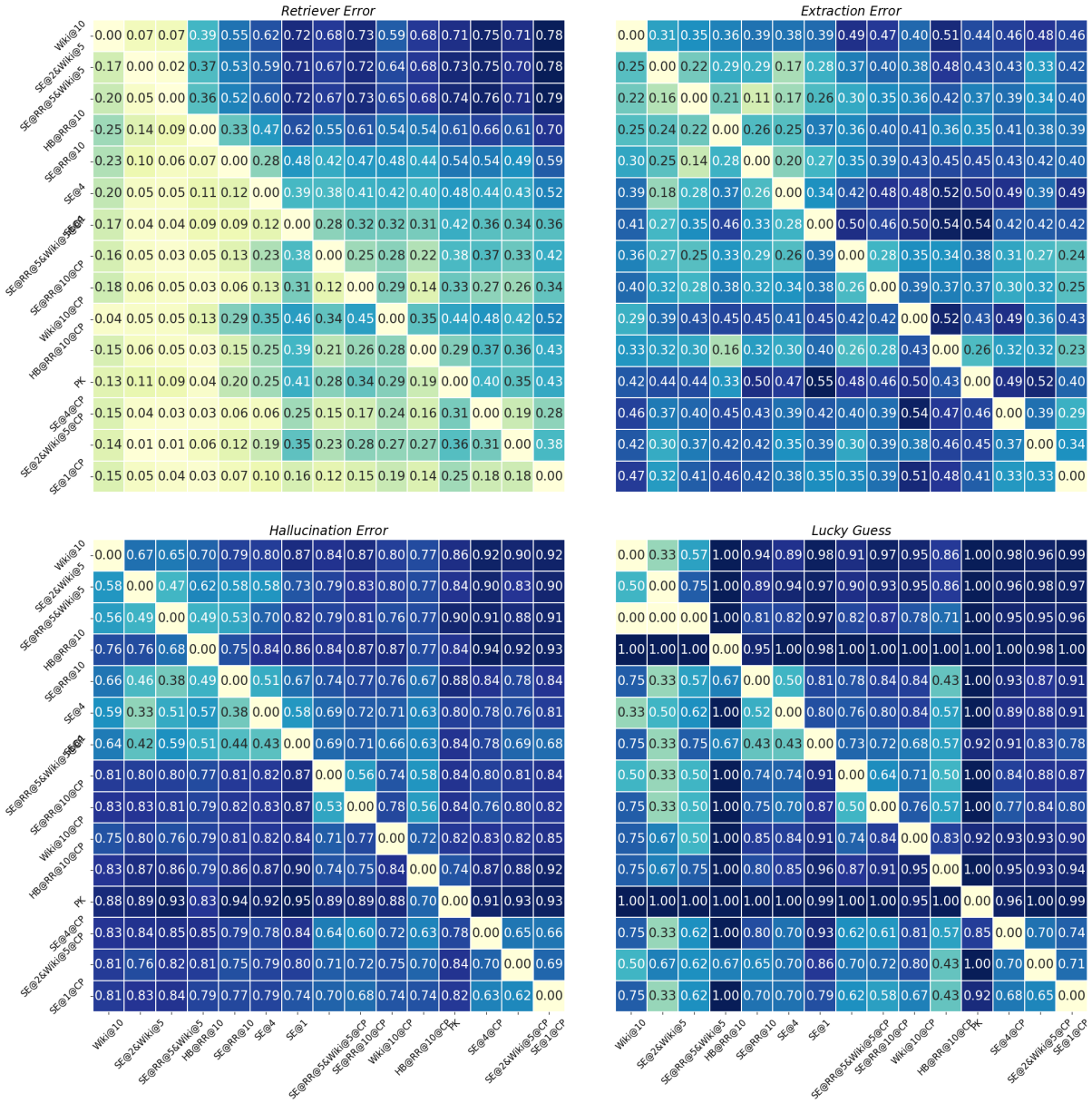

Figure 8: Full Error Relative Win Ratio between different Retrievers with Llama-chat 7b as base LM, evaluated on WebQ train set.

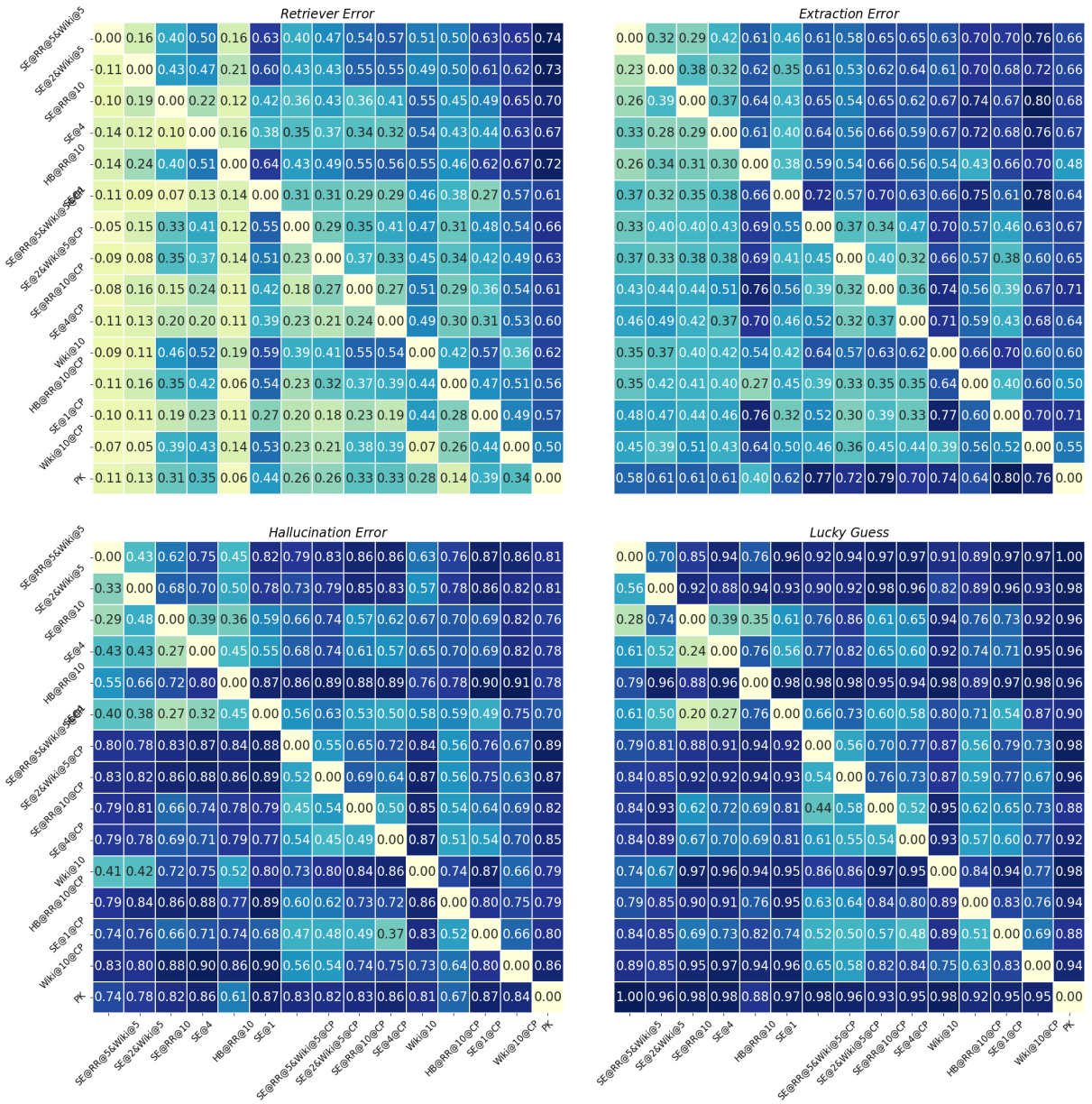

Figure 9: Full Error Relative Win Ratio between different Retrievers with Llama-chat 13b as base LM, evaluated on TriviaQA validation set.