ECAI 2024
U. Endriss et al. (Eds.)
© 2024 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA240778

ANSPRE: Improving Question-Answering in Large Language Models with Answer-Prefix Generation

Nguyen-Khang Le[a,*,1], Dieu-Hien Nguyen[a,1] and Le Minh Nguyen[a,]**

[a]Japan Advanced Institute of Science and Technology
ORCID (Nguyen-Khang Le): https://orcid.org/0000-0001-6585-5470, ORCID (Dieu-Hien Nguyen): https://orcid.org/0000-0001-5238-733X, ORCID (Le Minh Nguyen): https://orcid.org/0000-0002-2265-1010

Abstract. Large language models (LLMs) and Retrieval-Augmented-Generation (RAG) show remarkable capabilities in Open-domain question-answering (ODQA). Despite the advancements, LLMs tend to generate verbose responses, of which only a small part is the answer phrase. Although the ability to produce the confidence score for the answer is essential when deploying LLMs in high-risk domains, sequence probabilities obtained from LLMs do not correlate well with the probabilities of correctness and thus fail to represent confidence scores. This study introduces Answer-prefix Generation (ANSPRE) to improve generation quality, allowing the LLMs to output answer phrases and produce highly reliable confidence scores. We guide the model in predicting the answer phrase using an answer prefix and design a ranking score that integrates parametric and non-parametric knowledge. The answer phrases and their corresponding scores enable ANSPRE to aggregate results from different documents and samplings to boost performance and produce confidence scores highly correlated with correctness. We show that ANSPRE can be applied to any LLM and present an approach called SELF-ANSPRE to combine ANSPRE with Self-reflective RAG, a state-of-the-art framework based on reflection tokens. Empirical evaluation on popular ODQA benchmarks shows that ANSPRE and SELF-ANSPRE significantly improve state-of-the-art LLMs and RAG frameworks. An in-depth analysis shows that confidence scores produced by ANSPRE are highly correlated to the likelihood of correctness.

1 Introduction

State-of-the-art Large Language Models (LLMs) have demonstrated outstanding capabilities in Open-domain question-answering (ODQA). As the knowledge embedded in the LLMs parameters does not always suffice and is invalidated by the changing world, Retrieval-Augmented Generation (RAG) [15, 7] proposes augmenting the question with the documents retrieved from a knowledge base. Consider the question *"What gambling game, requiring two coins to play, was popular in World War I?"*. Although the desired answer is the word "Two-Up," RAG with a pre-trained LLM (Figure 1, left) tends to produce a verbose response, typically by providing contextual information to elaborate the answer. Although the response is informative, extracting the answer phrase ("Two-up" in the example) from the response is not trivial. While we can employ instruction-tuned [24] and reinforced [18] LLMs to output the answer phrase, experiments show that their performance is inadequate and highly depends on the models, system prompts, and instructions.

Another crucial aspect of LLMs is their ability to produce confidence scores for answers, particularly crucial when deploying them in high-risk domains such as law, finance, or healthcare[10]. Language model *calibration*, the property of the predicted confidence being correlated with the probabilities of correctness [6], is one common aspect to assess the reliability of the confidence score. Although LLMs can obtain the sequence probability of the response, we show that this probability is unreliable in terms of *calibration* and should not serve as the confidence score.

The inability to identify the answer phrase and produce the confidence score limits the application of LLMs. First, many practical systems require further processing of the answer phrase, such as mapping to an entity in the database [5, 4] to provide hyperlinks and references. Second, unreliable confidence scores restrict the application of LLMs in high-risk domains. Also, while aggregating the answer across generation samplings has shown remarkable performance gain in mathematics tasks [23], applying this method to RAG is problematic because of the lack of consensus in the LLMs' responses.

This paper introduces Answer-prefix Generation (ANSPRE) to improve the generation quality, allow the model to output the answer phrase, and produce a reliable confidence score. The main idea is to append the prompt with a sequence of text that leads to the answer phrase. We refer to this sequence of text as the *answer prefix*. Figure 1 (right) shows the overview of ANSPRE. In particular, given a question, ANSPRE first generates the *answer prefix* using curated few-shot examples (Step 1). Our preliminary study shows that only a handful of handcrafted examples are enough to generate a high-quality *answer prefix*. ANSPRE then uses an existing retriever to retrieve relevant documents from the knowledge base (Step 2), similar to ordinary RAG. Subsequently, ANSPRE combines the document, the question, and the answer prefix and prompts the LLM to generate the answer phrase (Step 3). Finally, ANSPRE aggregates the answer phrases and their confidence scores across documents and samplings to produce the final answer (Step 4). We show that ANSPRE outputs high-quality answer phrases and produces confidence scores that highly correlate with correctness. The answer phrases and confidence scores enable ANSPRE to aggregate answers from multiple documents and samplings to form a high-quality answer.

ANSPRE can be applied in any LLM and even sophisticated sys-

* Corresponding Author. Email: lnkhang@jaist.ac.jp
** Corresponding Author. Email: nguyenml@jaist.ac.jp
1 Equal contribution.

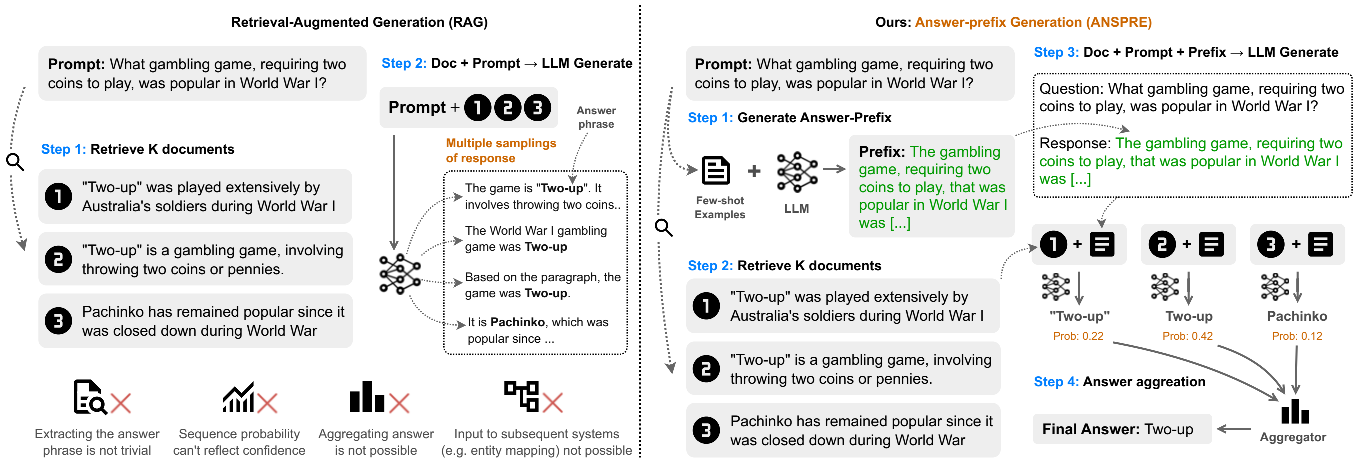

Retrieval-Augmented Generation (RAG)
Prompt: What gambling game, requiring two coins to play, was popular in World War I?
Step 1: Retrieve K documents
1 "Two-up" was played extensively by Australia's soldiers during World War I
2 "Two-up" is a gambling game, involving throwing two coins or pennies.
3 Pachinko has remained popular since it was closed down during World War
Step 2: Doc + Prompt → LLM Generate
Prompt + 1 2 3
Answer phrase
Multiple samplings of response
The game is "Two-up". It involves throwing two coins..
The World War I gambling game was Two-up
Based on the paragraph, the game was Two-up.
It is Pachinko, which was popular since ...
Extracting the answer phrase is not trivial
Sequence probability can't reflect confidence
Aggregating answer is not possible
Input to subsequent systems (e.g. entity mapping) not possible
Ours: Answer-prefix Generation (ANSPRE)
Prompt: What gambling game, requiring two coins to play, was popular in World War I?
Step 1: Generate Answer-Prefix
Few-shot Examples
LLM
Prefix: The gambling game, requiring two coins to play, that was popular in World War I was [...]
Step 2: Retrieve K documents
1 "Two-up" was played extensively by Australia's soldiers during World War I
2 "Two-up" is a gambling game, involving throwing two coins or pennies.
3 Pachinko has remained popular since it was closed down during World War
Step 3: Doc + Prompt + Prefix → LLM Generate
Question: What gambling game, requiring two coins to play, was popular in World War I?
Response: The gambling game, requiring two coins to play, that was popular in World War I was [...]
"Two-up"
Prob: 0.22
Two-up
Prob: 0.42
Pachinko
Prob: 0.12
Step 4: Answer aggregation
Final Answer: Two-up
Aggregator

Figure 1: Overview of ANSPRE.

tems. Recently, the inspiring work of Self-Reflecive RAG (SEFT-RAG)[1] improves the LLM generation by introducing reflection tokens to decide whether to retrieve documents and ranking the response by the utility of the document and the answer, achieving state-of-the-art results on various ODQA benchmarks. We show that ANSPRE can further improve SEFT-RAG and introduce the Self-Reflective Answer-Prefix Generation (SELF-ANSPRE). Figure 2 shows the overall of SELF-ANSPRE. In SELF-ANSPRE, we combine the confidence scores from ANSPRE with the scores from the reflection tokens to form the final score for ranking.

The contribution of this paper is as follows.

- We introduce ANSPRE to improve the generation quality, allow the model to output the answer phrase, and produce a reliable confidence score. The produced confidence scores highly correlate with the probabilities of correctness. The answer phrases and confidence scores broaden the applicability of LLM and enable the use of aggregating techniques to boost performance further.
- Inspired by the SEFT-RAG approach, we introduce SELF-ANSPRE. SELF-ANSPRE maintains the beneficial properties of SEFT-RAG while significantly improving the generation quality by combining the confidence scores and reflection token scores.
- Empirical results on three ODQA benchmarks and various LLM architectures show that ANSPRE significantly improves pre-trained and instruction-tuned LLMs. Also, SELF-ANSPRE significantly improves SEFT-RAG. Our analysis indicates the importance of each ANSPRE component and shows that the confidence scores from ANSPRE highly correlate with correctness and are more reliable than sequence probabilities of LLMs, in terms of *calibration*.

2 ANSPRE Generation

2.1 Problem Formulation and Overview

Problem Formulation. Consider the question *"What gambling game, requiring two coins to play, was popular in World War I?"*. Employing an RAG pipeline with LLM, we want the LLM to output the phrase "Two-up" with the corresponding confidence score.

Formally, given a question Q, an LLM $\mathcal{M}$, and k relevant documents $\mathcal{D} = \{D_1, D_2, ..., D_k\}$ retrieved by an existing retriever, we aim to obtain the answer phrase to the question Q, along with the confidence score. The LLM $\mathcal{M}$ is formulated by the next token prediction distribution $P_{\mathcal{M}}(y_t|X, y_{<t})$ where X is the input prompt and y_t is the t-th token in the response. Let Y be the response generated by $\mathcal{M}$. The sequence probability of the response is calculated as

$$P_{\mathcal{M}}(Y|X) = \prod_{t=1}^{|Y|} P_{\mathcal{M}}(y_t|X, y_{<t})$$

In the context of RAG, let Y be the LLM's response consisting of multiple tokens $Y = [y_1, y_2, ..., y_T]$ where y_t is the t-th token. The tokens in the response are generated by the distribution $P_{\mathcal{M}}(y_t|Q, D, y_{<t})$. The probability of the response is $P_{\mathcal{M}}(Y|Q, D)$. In practice, the cumulative log probability $log(P_{\mathcal{M}}(Y|Q, D))$ is used for computations. Standard RAG poses some limitations. First, only a small set of tokens in Y belongs to the answer phrase. Second, we show that $P_{\mathcal{M}}(Y|Q, D)$ does not correlate well with the probability of correctness and thus cannot represent the confidence of the answer.

Background on Calibration. Consider an answer with a confidence score in a range $[0, 1]$. If the confidence score is 0.6, we would expect this answer's probability to be correct is 0.6. Formally, given an answer prediction $\hat{Y}$, the true answer Y, and the prediction's confidence score $f_{con}(\hat{Y})$ estimated by the model f_{con}, a perfectly calibrated model satisfies the following.

$$P(\hat{Y} = Y|f_{con}(\hat{Y} = p)) = p, \forall p \in [0, 1]$$

Expected Calibration Error (ECE)[6] is a commonly used metric to measure calibration. In practice, we calculate the ECE by bucketing predictions into M disjoint equally sized interval bins based on confidence. The ECE is a weighted average of the discrepancy between each bucket's accuracy and confidence.

$$\sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{con}(B_m)|$$

where $|B_m|$ is the m-th bucket, acc(B_m) is the average accuracy, and con(B_m) is the average confidence of the bucket.

Few-shot and Instruction-tuning Approaches. One straightforward approach to obtaining the answer phrase is to provide few-shot examples [19] to guide the LLM to output only the answer phrase. However, in the RAG pipeline, the input is augmented with the multiple retrieved documents, whose sequence length can be hundreds of tokens. This property of RAG requires few-shot examples to be very long, rendering the few-shot approach infeasible regarding memory and scalability. Another approach is to use LLMs that are instruction-tuned [24] and reinforced [18]. Although we can instruct these LLMs to output the answer phrase, our experiments show that their performance is inadequate and highly sensitive to minor changes in system

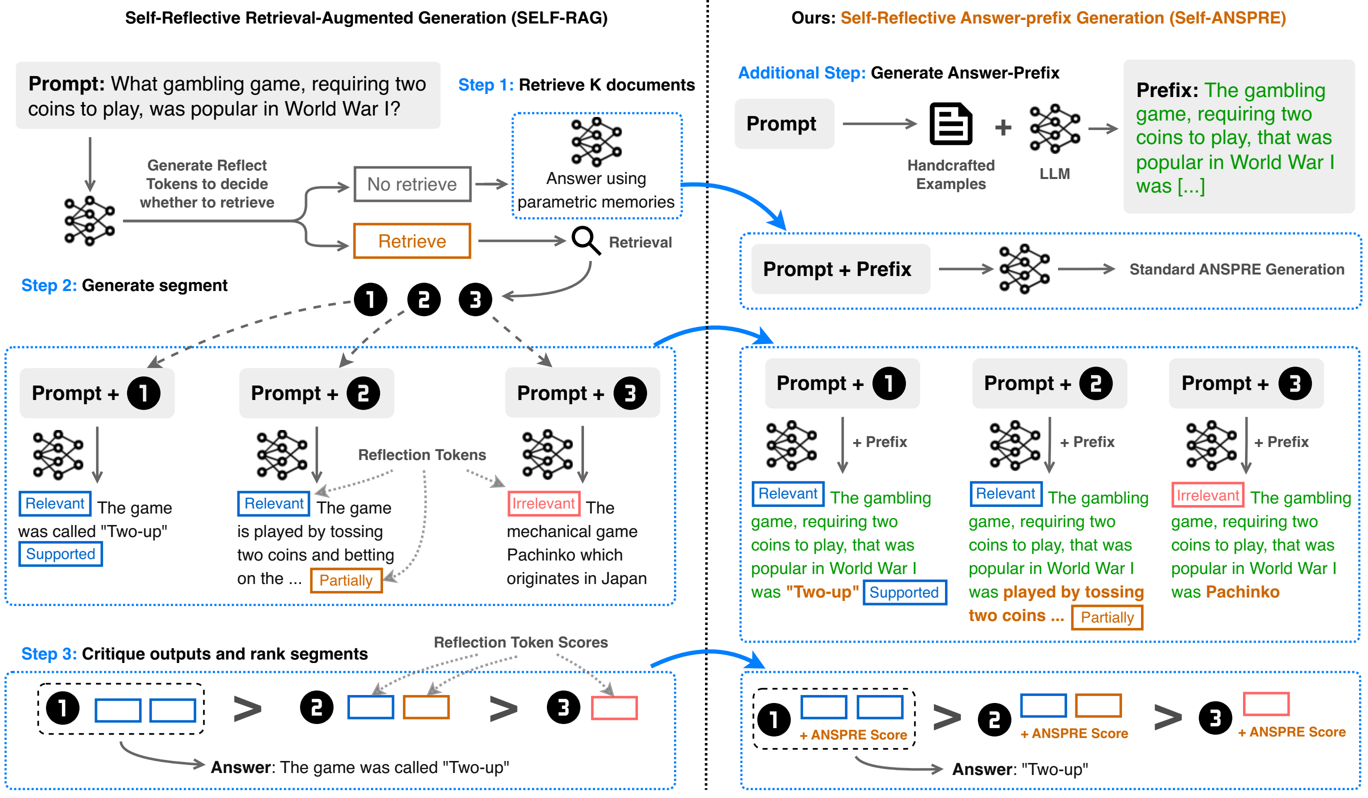

Self-Reflective Retrieval-Augmented Generation (SELF-RAG)
Ours: Self-Reflective Answer-prefix Generation (Self-ANSPRE)
Prompt: What gambling game, requiring two coins to play, was popular in World War I?
Step 1: Retrieve K documents
Generate Reflect Tokens to decide whether to retrieve
No retrieve
Retrieve
Answer using parametric memories
Retrieval
Step 2: Generate segment
Prompt + 1
Prompt + 2
Prompt + 3
Reflection Tokens
Relevant The game was called "Two-up" Supported
Relevant The game is played by tossing two coins and betting on the ... Partially
Irrelevant The mechanical game Pachinko which originates in Japan
Step 3: Critique outputs and rank segments
Reflection Token Scores
Answer: The game was called "Two-up"
Additional Step: Generate Answer-Prefix
Prompt
Handcrafted Examples
LLM
Prefix: The gambling game, requiring two coins to play, that was popular in World War I was [...]
Prompt + Prefix
Standard ANSPRE Generation
+ Prefix
Relevant The gambling game, requiring two coins to play, that was popular in World War I was "Two-up" Supported
Relevant The gambling game, requiring two coins to play, that was popular in World War I was played by tossing two coins ... Partially
Irrelevant The gambling game, requiring two coins to play, that was popular in World War I was Pachinko
+ ANSPRE Score
Answer: "Two-up"

Figure 2: Overview of SEFT-RAG and the modifications in SELF-ANSPRE

prompts and instruction prompts. Second, unlike pre-trained LLMs, fine-tuned LLMs require an intensive amount of labeled data for fine-tuning and reinforcement learning, which is not always available.

Overview. Figure 1 (right) shows the overview of ANSPRE. We aim to improve LLM generation quality by applying an *answer prefix* to the prompt before generating the answer. For the question *"What gambling game, requiring two coins to play, was popular in World War I"*, one possible *answer prefix* is *"The gambling game, requiring two coins to play, that was popular in World War I was ___"*. Most LLMs are trained following causal language modeling, which trains the model to predict the next token in the sequence. Intuitively, by appending the *answer prefix* to the question, we can expect that the next generated tokens will form the answer phrase. The cumulative probabilities of these tokens can also be used to produce the confidence score of the answer phrase. The next subsection describes the main steps in ANSPRE and the adaption of ANSPRE to the complex system Self-Reflective RAG.

2.2 *Generating Answer Prefix*

Consider the example question *"What gambling game, requiring two coins to play, was popular in World War I?"*. Suppose the answer to this question is *[ANSWER]*. The declarative form of the question will be *"The gambling game, requiring two coins to play, that was popular in World War I was [ANSWER]"*. The part of the declarative form without the [ANSWER] is the *answer prefix*. We leverage this intuition and use the LLM to generate the *answer prefix*.

Let E be the *answer prefix*. We use the LLM $\mathcal{M}$ with few-shot prompting and handcrafted examples to generate E. Our preliminary study shows that only a handful of handcrafted examples are enough to generate a high-quality *answer prefix*. We manually curate the questions and their declarative forms to use as few-shot examples.

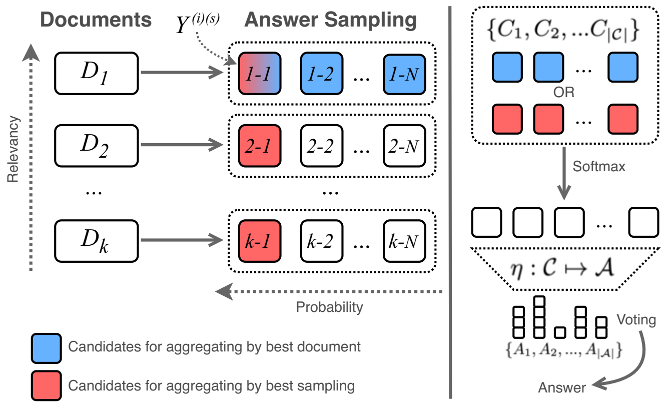

Documents
$Y^{(i)(s)}$
Answer Sampling
Relevancy
D_1
D_2
D_k
1-1
1-2
1-N
2-1
2-2
2-N
k-1
k-2
k-N
Probability
$\{C_1, C_2, ...C_{|\mathcal{C}|}\}$
OR
Softmax
$\eta : \mathcal{C} \mapsto \mathcal{A}$
Voting
$\{A_1, A_2, ..., A_{|\mathcal{A}|}\}$
Answer
Candidates for aggregating by best document
Candidates for aggregating by best sampling

Figure 3: Answer aggregation overview

We use the tag *[ANSWER]* to represent the answer in the declarative form. The questions are across the wh-questions ("who", "what", "where", "when", "why", "which", "whose", "how"). We create three examples for each type of wh-question, resulting in 24 few-shot examples. The number of examples is fixed and does not affect the scalability of the method. We use the following prompt template for the few-shot examples: *"Given an interrogative sentence: "{question}". Assuming the answer to this question is [ANSWER], transform the sentence to the declarative form."*. We prompt the LLM $\mathcal{M}$ using the few-shot examples to generate E. When the tag *[ANSWER]* is generated, we stop the generation to obtain the *answer prefix* E.

2.3 *Answer Phrase Generation*

With the answer prefix E obtained from the previous step, we condition the answer generation as the next token generation. For each document D_i, we generate the answer phrase using the next-token distribution $P_{\mathcal{M}}(y_t|Q \oplus E, D_i, y_{<t})$, where $\oplus$ indicates concatenation. We continue to generate the next tokens until the end of the phrase, indicated by punctuation or special tokens. We use beam search with beam width B to obtain the best N answer samplings ($B \geq N$). The samplings are used in the answer aggregation step.

In language models, *parametric knowledge* refers to the knowledge stored in the model parameters, and *non-parametric knowledge* refers to knowledge from external sources such as retrieved documents. We design the following scores to integrate these two sources of knowledge.

$$\mathcal{S}_{\text{Phrase}} = P_{\mathcal{M}}(Y|Q \oplus E, D_i)$$
$$\mathcal{S}_{\text{Sentence}} = P_{\mathcal{M}}(E \oplus Y)$$

$\mathcal{S}_{\text{Phrase}}$ is the probability of the answer phrase conditioned on the question, the *answer prefix*, and the retrieved document. $\mathcal{S}_{\text{Phrase}}$ reflects the *non-parametric knowledge* as it is conditioned on the retrieved document. On the other hand, $\mathcal{S}_{\text{Sentence}}$ is the sequence probability of the answer sentence, which is the concatenation of the *answer prefix* and the answer phrase. Consider the example in Figure 1 (right); the answer sentence will be *"The gambling game, requiring two coins to play, that was popular in World War I was Two-up"*. The model uses only the knowledge stored in the parameters to assess the probability of this sentence. Therefore, $\mathcal{S}_{\text{Sentence}}$ reflects the *parametric knowledge*. The final ANSPRE score is the weighted sum of the two scores and is calculated as follows.

$$\mathcal{S}_{\text{ANSPRE}} = w_{\text{Phrase}} * \mathcal{S}_{\text{Phrase}} + w_{\text{Sentence}} * \mathcal{S}_{\text{Sentence}}$$

where w_{Phrase} and w_{Sentence} are the pre-defined weight terms and $w_{\text{Phrase}} + w_{\text{Sentence}} = 1$. We use $\mathcal{S}_{\text{Phrase}}$ as the confidence score and use $\mathcal{S}_{\text{ANSPRE}}$ to aggregate and rank the answers.

2.4 Answer Aggregation

After obtaining the candidate answer phrases and their scores, we propose a method to aggregate them. Figure 3 shows the overview of the answer aggregation process. We calculate each candidate's final score and normalize them using the softmax function. We group the candidate answer phrases and use the sum of the scores in each group to rank the answers.

Formally, assuming $\mathcal{C} = \{C_1, C_2, ...C_{|\mathcal{C}|}\}$ is the candidate answer phrases, $\mathcal{A} = \{A_1, A_2, ..., A_{|\mathcal{A}|}\}$ is the set of normalized answer phrases, surjective function $\eta : \mathcal{C} \mapsto \mathcal{A}$ is the normalizing function (removing punctuation and converting to lower case), $\mathcal{S}(c)$ is the final score of the answer phrase c, we calculate the score used for ranking the answer as follows.

$$Score(A_i) = \sum_{\eta(c_j)=A_i} \frac{e^{\mathcal{S}(C_j)}}{\sum_{C \in \mathcal{C}} e^{\mathcal{S}(C)}}$$

The answer with the highest score is selected as the final answer. We propose two approaches to aggregate the answers. Assuming that the retrieved documents $\mathcal{D} = \{D_1, D_2, ..., D_k\}$ are ordered by the decrease of relevancy, the most relevant document is D_1. The beam search generates N samplings for each document in the answer phrase generation step. We denote $Y^{(i)(s)}$ as the s-th output in the samplings ($1 \leq s \leq N$) generated using document D_i. We assume that the samplings are ordered by the decrease of cumulative probability ($Y^{(i)(1)}$ has the highest probability). The first approach aggregates all samplings in the most relevant documents. For this approach, the candidates are collected as $\mathcal{C} = \{Y^{(1)(s)}|1 \leq s \leq N\}$. The second approach aggregates the best answer samplings across all documents. For this approach, the candidates are collected as $\mathcal{C} = \{Y^{(i)(1)}|1 \leq i \leq k\}$.

2.5 Adaptation to Self-Reflective RAG

Self-Reflective Retrieval-Augmented Generation (SEFT-RAG) [1] achieves state-of-the-art results on various ODQA benchmarks. SEFT-RAG (Figure 2, left) trains an LLM that dynamically retrieves documents as needed during text generation and critiques the documents and the response using unique tokens known as reflection tokens. We introduce the modifications to key components of SEFT-RAG to adapt the ANSPRE technique and propose SELF-ANSPRE (Figure 2, right).

The overall process of SELF-ANSPRE is similar to SEFT-RAG except for some modifications in the answer generation and critique steps. We use an LLM trained by SEFT-RAG approach [1]. We introduce an additional step to generate the *answer prefix*, similar to Section 2.2. Similar to SELF-RAG, the LLM first generates the reflection tokens (*[Retrieve]/[No retrieve]*) to decide whether to retrieve documents (Step 1). If no documents are needed, we use the prompt and the *answer prefix* to generate the answer, similar to standard ANSPRE but without the document. If documents are needed, we retrieve k documents using an existing retriever and use them to generate the answers (Step 2). In this step, SEFT-RAG uses the LLM to generate the response with reflection tokens. Each reflection token critiques different aspects of the triplet (document, question, response). In SELF-ANSPRE, we modify this component by adding the *answer prefix* before the main generation. After generation, we use the tokens between the *answer prefix* and the following reflection tokens as the answer phrase and calculate the ANSPRE score, as in Section 2.3.

The final step is to critique and rank the outputs (Step 3). In SELF-RAG, there are three types of critic tokens (ISREL, ISSUP, ISUSE). During the generation, SELF-RAG uses beam search to acquire the top sequence continuations at each step t and return the best sequence at the end. The score of the sequence is updated with the critic score, which is the linear weighted sum of the normalized probability of each critic token type. The following is the formula for the critique score from the work of Asai et al. [1].

$$\mathcal{S}_{\text{critique}} = \sum_{G \in \mathcal{G}} w^G s_t^G$$

where $\mathcal{G} = \{\text{ISREL}, \text{ISSUP}, \text{ISUSE}\}$ is the set of critic token types, s_t^G is the probability of the desirable token for type G, and w^G is the weights. In SELF-ANSPRE, we extend the formula to include the ANSPRE score of the answer.

$$\mathcal{S}_{\text{critique}} = \sum_{G \in \mathcal{G}} w^G s_t^G + w^A \mathcal{S}_{\text{ANSPRE}}$$

where w^A is the weight terms and $\mathcal{S}_{\text{ANSPRE}}$ is the ANSPRE score. The sequence with the highest score is returned at the end of the generation.

3 Experiments

3.1 Tasks and Datasets

We conduct evaluations on three ODQA benchmarks.

	TriviaQA			PopQA			NaturalQuestion		
Model	Match	EM	F1	Match	EM	F1	Match	EM	F1
Pre-trained Foundation LLM with retrieval									
Llama-2$_{7B}$	48.28	9.83	24.83	38.74	6.72	18.65	23.85	0.94	7.42
Llama-2$_{7B}$-ANSPRE	52.85$_{\uparrow 4.57}$	47.61$_{\uparrow 37.78}$	55.55$_{\uparrow 30.72}$	42.17$_{\uparrow 3.43}$	43.1$_{\uparrow 36.38}$	47.01$_{\uparrow 28.36}$	24.9$_{\uparrow 1.05}$	23.71$_{\uparrow 22.77}$	30.73$_{\uparrow 23.31}$
Llama2$_{13B}$	52.48	17.38	32.35	42.17	5.15	18.79	26.15	4.96	14.32
Llama2$_{13B}$-ANSPRE	58.51$_{\uparrow 6.03}$	53.64$_{\uparrow 36.26}$	61.78$_{\uparrow 29.43}$	46.03$_{\uparrow 3.86}$	44.39$_{\uparrow 39.24}$	47.98$_{\uparrow 29.19}$	30.36$_{\uparrow 4.21}$	29.34$_{\uparrow 24.38}$	37.52$_{\uparrow 23.2}$
Mistral$_{7B}$	54.29	12.59	26.25	44.53	4.50	17.05	33.52	3.32	10.31
Mistral$_{7B}$-ANSPRE	61.11$_{\uparrow 6.82}$	55.13$_{\uparrow 42.54}$	64.88$_{\uparrow 38.63}$	47.53$_{\uparrow 3.0}$	45.82$_{\uparrow 41.32}$	49.97$_{\uparrow 32.92}$	32.52$_{\downarrow 1.0}$	29.89$_{\uparrow 26.57}$	40.23$_{\uparrow 29.92}$
Gemma$_{7B}$	58.91	31.40	44.23	42.74	21.30	31.45	38.67	9.72	20.06
Gemma$_{7B}$-ANSPRE	61.89$_{\uparrow 2.98}$	55.18$_{\uparrow 23.78}$	65.35$_{\uparrow 21.12}$	46.53$_{\uparrow 3.79}$	44.89$_{\uparrow 23.59}$	47.88$_{\uparrow 16.43}$	31.58$_{\downarrow 7.09}$	28.48$_{\uparrow 18.76}$	39.46$_{\uparrow 19.4}$
OPT$_{6.7B}$	46.37	3.08	11.22	40.89	1.36	9.12	22.13	0.14	3.28
OPT$_{6.7B}$-ANSPRE	41.52$_{\downarrow 4.85}$	35.16$_{\uparrow 32.08}$	43.19$_{\uparrow 31.97}$	40.1$_{\downarrow 0.79}$	37.6$_{\uparrow 36.24}$	41.21$_{\uparrow 32.09}$	19.83$_{\downarrow 2.3}$	18.64$_{\uparrow 18.5}$	26.23$_{\uparrow 22.95}$
Pre-trained Foundation LLM without retrieval (only parametric memories)									
Llama2$_{13B}$	46.47	33.38	42.62	15.23	9.86	13.88	16.29	8.17	15.48
Llama2$_{13B}$ANSPRE	61.23$_{\uparrow 14.76}$	55.08$_{\uparrow 21.7}$	64.66$_{\uparrow 22.04}$	24.16$_{\uparrow 8.93}$	23.3$_{\uparrow 13.44}$	26.67$_{\uparrow 12.79}$	30.19$_{\uparrow 13.9}$	28.03$_{\uparrow 19.86}$	38.12$_{\uparrow 22.64}$
Llama2$_{7B}$	28.39	10.64	20.19	11.58	4.65	9.04	12.13	2.96	8.68
Llama2$_{7B}$-ANSPRE	55.59$_{\uparrow 27.2}$	49.31$_{\uparrow 38.67}$	58.34$_{\uparrow 38.15}$	24.37$_{\uparrow 12.79}$	25.66$_{\uparrow 21.01}$	27.13$_{\uparrow 18.09}$	25.24$_{\uparrow 13.11}$	23.68$_{\uparrow 20.72}$	32.41$_{\uparrow 23.73}$
Pre-trained Foundation LLM on multilingual data (focus on Chinese and English) with retrieval									
Baichuan2$_{7B}$	44.20	13.58	26.46	40.74	4.65	15.34	20.78	2.11	7.78
Baichuan2$_{7B}$-ANSPRE	54.98$_{\uparrow 10.78}$	48.48$_{\uparrow 34.9}$	57.88$_{\uparrow 31.42}$	41.24$_{\uparrow 0.5}$	40.81$_{\uparrow 36.16}$	43.92$_{\uparrow 28.58}$	24.43$_{\uparrow 3.65}$	21.33$_{\uparrow 19.22}$	31.05$_{\uparrow 23.27}$
Baichuan2$_{13B}$	48.99	8.10	21.54	39.10	4.29	14.62	27.12	0.83	6.49
Baichuan2$_{13B}$-ANSPRE	53.89$_{\uparrow 4.9}$	48.04$_{\uparrow 39.94}$	57.07$_{\uparrow 35.53}$	40.53$_{\uparrow 1.43}$	39.53$_{\uparrow 35.24}$	42.85$_{\uparrow 28.23}$	27.34$_{\uparrow 0.22}$	25.4$_{\uparrow 24.57}$	34.9$_{\uparrow 28.41}$
Qwen$_{7B}$	54.49	0.44	5.27	48.96	0.14	2.10	33.93	0.19	3.68
Qwen$_{7B}$-ANSPRE	57.09$_{\uparrow 2.6}$	48.0$_{\uparrow 47.56}$	59.64$_{\uparrow 54.37}$	48.61$_{\downarrow 0.35}$	46.18$_{\uparrow 46.04}$	50.94$_{\uparrow 48.84}$	30.72$_{\downarrow 3.21}$	24.43$_{\uparrow 24.24}$	36.17$_{\uparrow 32.49}$
Qwen$_{14B}$	57.04	1.38	5.76	47.18	0.36	2.86	36.62	0.50	3.55
Qwen$_{14B}$-ANSPRE	61.0$_{\uparrow 3.96}$	52.0$_{\uparrow 50.62}$	63.48$_{\uparrow 57.72}$	47.75$_{\uparrow 0.57}$	45.89$_{\uparrow 45.53}$	49.4$_{\uparrow 46.54}$	29.53$_{\downarrow 7.09}$	22.91$_{\uparrow 22.41}$	35.07$_{\uparrow 31.52}$
Instruction-tuned & Reinforced LLM									
Llama2-C$_{7B}$	60.41	33.20	48.32	45.18	23.73	36.33	38.14	9.81	21.19
Llama2-C$_{7B}$-ANSPRE	57.73$_{\downarrow 2.68}$	40.97$_{\uparrow 7.77}$	55.54$_{\uparrow 7.22}$	50.25$_{\uparrow 5.07}$	40.53$_{\uparrow 16.8}$	48.05$_{\uparrow 11.72}$	31.08$_{\downarrow 7.06}$	19.58$_{\uparrow 9.77}$	31.38$_{\uparrow 10.19}$
Llama2-C$_{13B}$	66.84	5.67	33.03	54.32	0.36	17.40	41.69	1.30	15.92
Llama2-C$_{13B}$-ANSPRE	62.03$_{\downarrow 4.81}$	47.0$_{\uparrow 41.33}$	61.08$_{\uparrow 28.05}$	50.46$_{\downarrow 3.86}$	42.82$_{\uparrow 42.46}$	49.11$_{\uparrow 31.71}$	33.13$_{\downarrow 8.56}$	22.99$_{\uparrow 21.69}$	35.43$_{\uparrow 19.51}$
Vicuna$_{13B}$	66.53	30.86	48.98	56.25	3.65	23.60	43.21	5.29	20.33
Vicuna$_{13B}$-ANSPRE	62.35$_{\downarrow 4.18}$	44.33$_{\uparrow 13.47}$	59.98$_{\uparrow 11.0}$	47.46$_{\downarrow 8.79}$	39.67$_{\uparrow 36.02}$	46.04$_{\uparrow 22.44}$	34.10$_{\downarrow 9.11}$	18.01$_{\uparrow 12.72}$	32.46$_{\uparrow 12.13}$
Self-Reflective Retrieval-Augmented-Generation									
SELF-RAG $_{7B}$	66.18	20.91	38.70	54.97	1.14	19.67	36.15	36.73	44.89
SELF-ANSPRE $_{7B}$	66.21$_{\uparrow 0.03}$	40.15$_{\uparrow 19.24}$	58.42$_{\uparrow 19.72}$	54.11$_{\downarrow 0.86}$	42.17$_{\uparrow 41.03}$	51.91$_{\uparrow 32.24}$	38.14$_{\uparrow 1.99}$	31.25$_{\downarrow 5.47}$	39.73$_{\downarrow 5.16}$
SELF-RAG $_{13B}$	67.59	19.14	38.37	55.83	0.79	20.10	38.73	40.75	48.72
SELF-ANSPRE $_{13B}$	66.99$_{\downarrow 0.6}$	46.78$_{\uparrow 27.64}$	62.98$_{\uparrow 24.61}$	52.61$_{\downarrow 3.22}$	38.74$_{\uparrow 37.95}$	49.41$_{\uparrow 29.31}$	40.3$_{\uparrow 1.57}$	35.68$_{\downarrow 5.07}$	44.79$_{\downarrow 3.93}$

Table 1: Result on three ODQA tasks of baseline LLMs with and without ANSPRE. Every two lines compare the performance of LLM before and after applying ANSPRE. Blue numbers indicate a performance gain and red numbers indicate a performance decrease of ANSPRE compared to the baseline version.

PopQA[17] includes questions about factual knowledge. We evaluate the long-tail subset, containing 1,399 rare entity queries whose monthly Wikipedia views are below 100.

TriviaQA[12] includes questions about factual knowledge. We use the TriviaQA-unfiltered (open) subset. Due to the unavailability of the TriviaQA-unfiltered test set, we adopted the validation and test split used in previous studies, comprising 7,313 test queries for assessment.

NaturalQuestion[13] includes factual questions from real users. We use the Open-NQ subset [14], containing 3,610 questions in the validation set.

We use zero-shot evaluations, where we use instructions to describe the tasks without providing few-shot examples. We evaluate the performance using **Exact-Match (EM)**, which requires the prediction to match the gold answer exactly; **F1-score**, which is measured based on the overlap between the prediction and the gold answer; and **Match-accuracy**, which assesses whether responses include the gold answers.

3.2 Baselines

We evaluate competitive, publicly available pre-trained LLMs into the following groups.

Pre-trained Foundation LLM. We evaluate Llama2$_{7B,13B}$[22], Mistral$_{7B}$ [9], Gemma$_{7B}$[21],OPT$_{6.7B}$[27].

LLM without retrieval. We evaluate Llama2$_{7B,13B}$ in the setting without retrieval. In this setting, the LLMs only use parametric memories to generate the answer.

LLM trained on multilingual data. We evaluate LLM trained on multilingual data, focusing on Chinese and English, that show competitive performance on a wide range of tasks. This group includes Qwen$_{7B,14B}$[2], Baichuan2$_{7B,13B}$[26].

Instruction-tuned/Reinforced LLM. We evaluate LLMs that are instruction-tuned and reinforced using proprietary data. This group includes Llama2-Chat$_{7B,13B}$, Vicuna$_{13B}$[3].

Self-Reflective RAG. We evaluate the SELF-RAG$_{7B,13B}$[1] and compare with our SELF-ANSPRE.

We compare the performance of each LLM with and without applying ANSPRE to evaluate the effectiveness of the method.

3.3 Experimental Settings

We use greedy decoding to generate the answer prefix in ANSPRE. Following previous work [1], we use the official 2018 English Wikipedia as the knowledge base by default, except for the PopQA dataset, where we use the December 2020 English Wikipedia. We use Contriever-MS MARCO[8] to retrieve top $k = 5$ documents for the retrieval step. For each document, we use beam search with a beam width of 10 to sample $N = 10$ answer samplings. By default, we assign the weight terms of ANSPRE as $w_{\text{Phrase}} = w_{\text{Sentence}} = 0.5$. We set the maximum number of new tokens to 100 during the ANSPRE's answer generation. For answer normalization, function η removes articles and punctuation, normalizes white spaces, and converts the answer to lowercase.

For instruction-tuned and reinforced baselines, we employ various prompts that instruct the LLM to output only the answer phrase and report the highest results. For SELF-RAG and SELF-ANSPRE, we use the same weight terms for the reflection tokens as in the work of Asai et al. [1], which is 1.0, 1.0, 0.5 for token types ISREL, ISSUP, and ISUSE, respectively. In SELF-ANSPRE, we use $w^A = 0.5$ by default.

4 Results and Analysis

4.1 Main Results

Performance comparison with baselines. Table 1 shows the performance difference between the baseline LLMs with and without ANSPRE. We use the same aggregator (aggregating samplings in the most relevant document) for all ANSPRE results for a fair comparison. In most cases, ANSPRE significantly improves the performance of baseline LLM in all metrics. The results suggest a constant improvement in EM and F1 scores across all models and benchmarks. In some cases, we witness a minor decrease in Match-accuracy from ANSPRE. This is reasonable, considering the baseline LLMs generate a much longer response than ANSPRE and increase the chance of containing the gold answers.

For the instruction-tuned and reinforced group, we instruct the baseline LLMs to output only the answer phrase. When not constrained to output only the answer phrase, baseline LLMs yield notably lower results. Even with this constraint, the performance of baseline LLMs remains significantly lower compared to ANSPRE. The result of this group suggests that ANSPRE significantly improves the baselines in EM and F1, despite some decrease in Match-accuracy. This decrease is expected as the baseline tends to provide contextual information to elaborate the answer, which helps the Match-accuracy. We observe a significant gap in the performance of baseline Llama2-C$_{7B}$ and Llama2-C$_{13B}$. Our error analysis shows that although being instructed to only output the answer phrase, Llama2-C$_{13B}$ tends to respond with conversation fillers. This result highlights the limitation of instruction-tuned models and shows that applying ANSPRE can mitigate the issue to some degree.

The result in the group of Self-Reflective RAG shows that SELF-ANSPRE outperforms SELF-RAG in EM and F1 in TriviaQA and PopQA while remaining a comparable Match-accuracy. In NaturalQuestion, SELF-ANSPRE outperforms SELF-RAG in Match-accuracy despite a decrease in EM and F1.

Confidence calibration comparison with baselines. Figure 4 shows the reliability diagram of the confidence score obtained by the sequence probability of the LLM and by ANSPRE. The result depicts that the confidence scores obtained by ANSPRE are better aligned with the probability of correctness. Also, normalizing across the sampling helps calibrate the confidence score better. Table 2 compares the confidence score produced by ANSPRE and baselines, measured in Expected Calibration Error (ECE). The result suggests that the confidence scores produced by ANSPRE significantly outperform the baselines. We observe that using ANSPRE score and normalizing the score across the samplings tends to produce the best calibration.

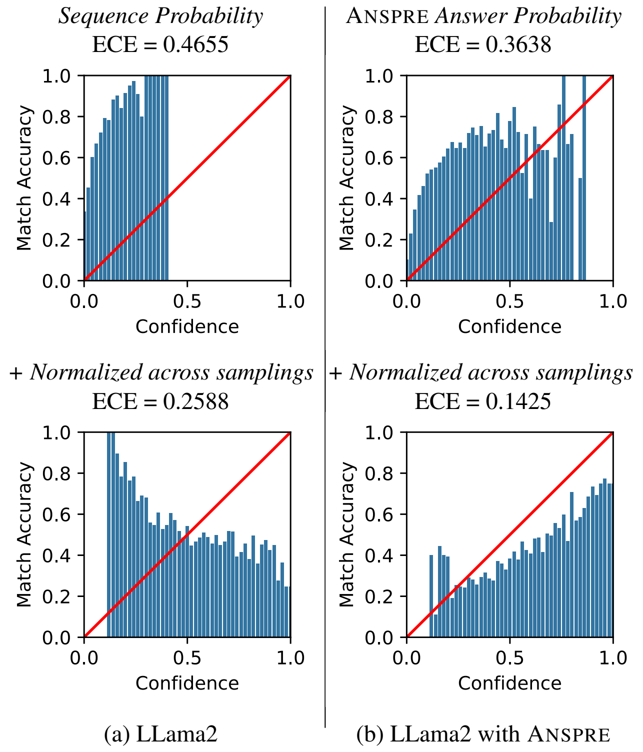


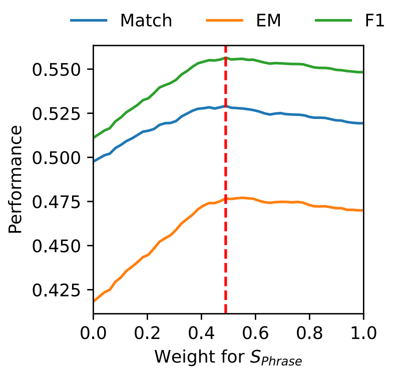
Figure 4: Llama2 reliability diagram. The left side shows the confidence score obtained by the sequence probability of LLM (top-left) and after normalized across samplings (bottom-left). The right side shows the confidence score obtained by ANSPRE (top-right) and after normalized across samplings (bottom-right). Each bar corresponds to one bucket, and the height is the average accuracy. The perfectly calibrated model should have the bars aligned with the red diagonal.

Model	ANSPRE	ECE (default)	ECE (normalized)
Llama2$_{7B}$		0.4655	0.2588
	✓	0.3638	**0.1426**
Mistral$_{7B}$		0.4572	0.2669
	✓	0.3519	**0.1045**
Gemma$_{7B}$		0.4518	0.3907
	✓	0.2489	**0.0840**
Baichuan2$_{7B}$		0.4150	0.2460
	✓	**0.0884**	0.1055
Qwen$_{7B}$		0.5489	0.3399
	✓	0.2037	**0.1463**

Table 2: Confidence score reliability, measured in Expected Calibration Error (ECE), lower is better. By *default*, the confidence score is obtained by cumulative probability (for baseline) and S_{Phrase} (for ANSPRE). *normalized* indicate normalizing across samplings.

		TriviaQA			PopQA			NaturalQuestion		
Model	Aggregator	Match	EM	F1	Match	EM	F1	Match	EM	F1
Llama2$_{7B}$	None	37.59	31.52	37.71	**43.46**	39.03	44.06	14.71	12.88	17.73
	Best samp across docs	46.66	39.68	47.49	40.39	36.31	41.04	20.86	19.14	25.08
	Samps in best doc	**52.85**	**47.61**	**55.55**	42.17	**43.10**	**47.01**	**24.90**	**23.71**	**30.73**
Gemma$_{7B}$	None	60.29	50.38	61.92	49.82	44.17	**49.10**	33.38	27.51	39.52
	Best samp across docs	**64.30**	55.15	**66.40**	45.96	42.82	45.99	**36.32**	**31.52**	**43.80**
	Samps in best doc	61.89	**55.18**	65.35	**46.53**	**44.89**	47.88	31.58	28.48	39.46
Mistral$_{7B}$	None	58.62	50.29	60.53	51.11	**46.82**	**51.48**	33.19	28.17	38.98
	Best samp across docs	**63.27**	**55.30**	**65.67**	44.75	42.17	45.46	**35.76**	**31.39**	**42.44**
	Samps in best doc	61.11	55.13	64.88	**47.53**	45.82	49.97	32.52	29.89	40.23
Qwen$_{7B}$	None	55.34	43.63	56.16	48.61	40.46	48.01	31.14	21.80	34.62
	Best samp across docs	**60.40**	**49.76**	**61.79**	43.82	39.81	45.10	**34.63**	**25.90**	**39.21**
	Samps in best doc	57.09	48.00	59.64	**48.61**	**46.18**	**50.94**	30.72	24.43	36.17

(a) Performance of ANSPRE with different aggregators



(b) Effect of ANSPRE weight

Figure 5: Analysis of ANSPRE: (a) Performance of LLMs with different aggregator settings: without using aggregator (None), aggregate the best samplings across all documents (Best samp across docs), and aggregate the samplings in the best document (Samps in best doc). **Bold** numbers indicate the best performance in each model group. (b) Effect of ANSPRE weight term w_{Phrase} on TriviaQA Match accuracy. The dotted red line indicates the weight with the highest Match accuracy.

4.2 Analysis

Effect of ANSPRE weights. Figure 5b shows the effect of changing the weight terms w_{Phrase} and w_{Sentence} on the performance of Llama2 ($w_{\text{Phrase}} + w_{\text{Sentence}} = 1$). The result suggests that ANSPRE achieves the best performance when $w_{\text{Phrase}} \approx w_{\text{Sentence}}$, indicating the importance of both *parametric knowledge* ($\mathcal{S}_{\text{Phrase}}$) and *non-parametric knowledge* ($\mathcal{S}_{\text{Sentence}}$). We also observe this trend in other LLMs.

Effect of aggregators. Table 5a compares the performance of ANSPRE with different aggregation techniques. For each LLM, the result compares the performance without aggregating and when applying the two proposed aggregation methods. The result suggests that applying the proposed aggregation techniques almost always boosts the performance in all metrics across all benchmarks. The result has no clear pattern for which method works best. Instead, which method works best depends on the type and size of the LLM. However, in most cases, the difference between the two methods is insignificant.

5 Related Work

Retrieval-Augmented Generation (RAG). RAG augments the input with retrieved documents from the knowledge base to improve knowledge-intensive tasks [15]. Other variants of RAG instruction-tune an LLM with input augmented by a fixed number of retrieved documents[16], or few-shot fine-tuning after jointly pre-training the retriever and LLM[8]. Instead of statically retrieving once, another line of work actively retrieves documents across the course of the LLM generation[11] or trains an LLM to generate API calls for information [20]. The inspiring work of SELF-RAG trains an arbitrary LLM that can decide whether retrieved documents are necessary and reflects on retrieved documents and the responses using reflection tokens. SELF-RAG achieves state-of-the-art results on various ODQA benchmarks. However, these approaches do not consider the importance of locating the answer phrase in the LLM's verbose responses. The confidence of the answers has yet to be fully studied in these approaches.

Re-ranking and Aggregating. Re-ranking, referring to re-ranking samplings or results from language models, is a common approach to enhance generation quality. With the advancements in LLM, many re-ranking and aggregating techniques have been proposed. Self-Consistency[23] samples a set of Chain-of-thought (CoT)[25] reasoning paths and answers, represented by sequences, and aggregates the answers. This technique requires marginalizing the answer in the sequence and is only studied in mathematics tasks, where the reasoning paths and answers follow a template. This requirement limits the application of Self-Consistency in other question-answering tasks. Our two aggregating techniques are inspired by Self-Consistency with some improvements. Thanks to ANSPRE ability to output the answer phrase, our aggregating techniques can be applied to ODQA tasks or any tasks where ANSPRE is applicable. Instead of aggregating over the samplings like Self-Consistency, our aggregating technique also specializes in the context of RAG, where we aggregate the answers in two directions: samplings in best documents and best sampling across documents.

6 Conclusions

This study introduces ANSPRE, a new generation technique to improve generation quality, guiding the LLMs to output the answer phrase and produce reliable confidence scores. ANSPRE leverages LLMs' causal language modeling nature and appends the prompt with a prefix that encourages the next-token predictions to be the answer phrase. We design a score that integrates parametric and non-parametric knowledge to improve the ranking in aggregation and produce highly reliable confidence scores. The answer phrases and their confidence scores enable ANSPRE to gather results from different documents and samplings to boost performance. We show that ANSPRE can be applied to any LLM and present SELF-ANSPRE, which combines ANSPRE with Self-reflective RAG. Empirical evaluation on three ODQA benchmarks shows that ANSPRE and SELF-ANSPRE significantly improve state-of-the-art LLMs. An in-depth analysis indicates the importance of individual ANSPRE components and the reliability of the confidence scores produced by ANSPRE.

References

[1] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

[2] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu. Qwen Technical Report, 2023.

[3] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, 3 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[4] P. Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Publishing Company, Incorporated, 2012. ISBN 3642311636.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. on Knowl. and Data Eng.*, 19(1):1–16, 1 2007. ISSN 1041-4347.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1321–1330. JMLR.org, 2017.

[7] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval Augmented Language Model Pre-Training. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR, 10 2020.

[8] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=jKN1pXi7b0.

[9] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7B, 2023.

[10] Z. Jiang, J. Araki, H. Ding, and G. Neubig. How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl{_}a{_}00407. URL https://aclanthology.org/2021.tacl-1.57.

[11] Z. Jiang, F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active Retrieval Augmented Generation. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore, 12 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495.

[12] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, 7 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147.

[13] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl{_}a{_}00276.

[14] K. Lee, M.-W. Chang, and K. Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. pages 6086–6096, 10 2019. doi: 10.18653/v1/P19-1612.

[15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[16] H. Luo, T. Zhang, Y.-S. Chuang, Y. Gong, Y. Kim, X. Wu, H. Meng, and J. Glass. Search Augmented Instruction Learning. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3717–3729, Singapore, 12 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.242. URL https://aclanthology.org/2023.findings-emnlp.242.

[17] 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 A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, 7 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546.

[18] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[20] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

[21] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y.-h. Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open Models Based on Gemini Research and Technology, 2024.

[22] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

[23] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[24] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023.

[26] A. Yang, B. Xiao, B. Wang, B. Zhang, C. Bian, C. Yin, C. Lv, D. Pan, D. Wang, D. Yan, F. Yang, F. Deng, F. Wang, F. Liu, G. Ai, G. Dong, H. Zhao, H. Xu, H. Sun, H. Zhang, H. Liu, J. Ji, J. Xie, J. Dai, K. Fang, L. Su, L. Song, L. Liu, L. Ru, L. Ma, M. Wang, M. Liu, M. Lin, N. Nie, P. Guo, R. Sun, T. Zhang, T. Li, T. Li, W. Cheng, W. Chen, X. Zeng, X. Wang, X. Chen, X. Men, X. Yu, X. Pan, Y. Shen, Y. Wang, Y. Li, Y. Jiang, Y. Gao, Y. Zhang, Z. Zhou, and Z. Wu. Baichuan 2: Open Large-scale Language Models, 2023.

[27] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, 2022.