CTRLA: Adaptive Retrieval-Augmented Generation via Probe-Guided Control

Huanshuo Liu[♫], **Hao Zhang**[♫,✉], **Zhijiang Guo**[✉], **Kuicai Dong**,
Xiangyang Li, **Yi Quan Lee**, **Cong Zhang**, **Yong Liu**
Noah's Ark Lab, Huawei Technologies Co., Ltd

Abstract

Retrieval-augmented generation (RAG) has emerged as a promising solution for mitigating hallucinations of large language models (LLMs) with retrieved external knowledge. Adaptive RAG enhances this approach by dynamically assessing the retrieval necessity, aiming to balance external and internal knowledge usage. However, existing adaptive RAG methods primarily realize retrieval on demand by relying on superficially verbalize-based or probability-based feedback of LLMs, or directly fine-tuning LLMs via carefully crafted datasets, resulting in unreliable retrieval necessity decisions, heavy extra costs, and sub-optimal response generation. We present the first attempts to delve into the internal states of LLMs to mitigate such issues by introducing an effective probe-guided adaptive RAG framework, termed CTRLA. Specifically, CTRLA employs an honesty probe to regulate the LLM's behavior by manipulating its representations for increased honesty, and a confidence probe to monitor the internal states of LLM and assess confidence levels, determining the retrieval necessity during generation. Experiments show that CTRLA is superior to existing adaptive RAG methods on a diverse set of tasks, the honesty control can effectively make LLMs more honest and confidence monitoring is proven to be a promising indicator of retrieval trigger.[1]

1 Introduction

Recent advancements in LLMs have shown immense potential across various NLP tasks [1, 50]. However, LLMs still struggle with accurately perceiving their factual knowledge boundaries [49, 57], while the knowledge memorized by LLMs may be incomplete, incorrect, and outdated [17, 65]. As a result, LLMs often generate specious answers that deviate from the facts, known as hallucination [21].

Retrieval-augmented generation (RAG) has drawn substantive effectiveness to mitigate hallucination by introducing external knowledge [18, 30] as context to LLM. Yet, existing RAG systems often invoke retrieval indiscriminately, overlooking LLMs' internal knowledge [69], *i.e.,* self-knowledge. Recent study [38] reveals that poorly (*e.g.,* distracting, irrelevant, or conflicting) retrieved results can impair model performance. Ideally, retrieval should be activated exclusively when the answer to the question is beyond LLM's self-knowledge [49, 76]. Adaptive RAG (ARAG) is designed to dynamically determine retrieval necessity by assessing the boundary of LLM's self-knowledge, aiming to balance external and self-knowledge utilization [4, 23]. However, perceiving knowledge boundaries [44, 73] directly is challenging. Alternatively, existing ARAG systems decide retrieval timing via (i) explicit verbalize-based LLM feedback [9, 76], (ii) probability-based feedback [23, 68], (iii) fine-tuning for capability injection [4, 38], etc. An overview of different RAG method categories is depicted in Figure 1.

[♫]The first two authors contributed equally.
[✉]Correspondence to Hao Zhang<hzhang26@outlook.com> and Zhijiang Guo<cartusguo@gmail.com>.
[1]Our codes are available at https://github.com/HSLiu-Initial/CtrlA.git.

Preprint. Work in progress.

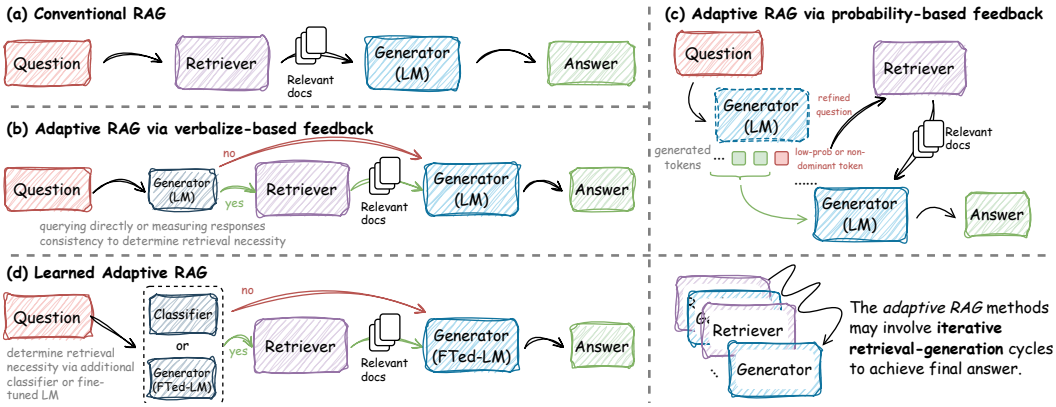

Figure 1: Overview of different RAG methods: (a) Conventional RAG indiscriminately retrieves documents to support answer generation. (b) Adaptive RAG (ARAG) via verbalize-based feedback directly queries LLM or evaluates response consistency through repeated queries to determine retrieval timing. (c) ARAG with probability-based feedback utilizes confidence scores derived from LLM's output token probabilities to determine retrieval timing. (d) Learned ARAG typically determines retrieval timing using an additional classifier or by fine-tuning an LLM with a specialized dataset. Additionally, ARAG may incorporate iterative retrieval-generation cycles to enhance answer quality.

However, there exists a considerable discrepancy between LLM's feedback mechanisms (whether verbalized responses or probability metrics) and LLM's internal cognition [29, 80]. Consequently, LLM may provide incorrect feedback, *e.g.,* "lying" in its feedback. Meanwhile, capability injection, which use extra fine-tuning to help LLM identify appropriate retrieval timing, is highly constrained by data quality and distribution [51]. This limitation often fails to adequately reflect genuine internal cognition of LLM. Besides, these methods typically assume that LLM can efficiently elicit their self-knowledge related to the question. Recent studies [2, 3] highlight that LLM faces difficulties using self-knowledge, and the presence of such knowledge does not guarantee it to be elicited accurately.

To better reflect the internal cognition of LLM and to determine retrieval necessity, we propose Probe-Guided **Con**t**r**o**l** based **A**daptive RAG, termed **CTRLA**. We characterize LLM's internal states and intervene in the LLM generation from two perspectives: ***honesty control*** and ***confidence monitoring***. ***Honesty control***, inspired by Yang et al. [73], is to align LLM outputs with genuine self-awareness. Through honesty control, we aim to enable LLM to recognize its limitations, avoid generating fabricating plausible information, and produce more stable outputs. ***Confidence monitoring*** focuses on assessing LLM's internal state, specifically its representation, to capture its genuine cognition [7, 32]. This strategy helps to reflect the real state of LLM within representational spaces, thus ensuring more reliable retrieval timing decision.

Specifically, we develop two plug-and-play probes in CTRLA, designed for behavioral intervention and monitoring, without fine-tuning LLM. Honesty probe manipulates LLM's representations to enhance its honesty, while confidence probe monitors its internal states to optimize retrieval necessity detection. Note that our probes can be efficiently trained in less than 1 minute, compared to significant time taken to finetune LLM. Meanwhile, we design a simple yet effective query formulation strategy to facilitate adaptive retrieval, reducing the impact of noise introduction and intent drift. To our best knowledge, we are the first to solve ARAG in representational space. Experiments on multiple benchmarks demonstrate the effectiveness of CTRLA. Through in-depth analysis, we show that manipulating LLM's internal states can make LLM more honest, while confidence monitoring is a reliable indicator of retrieval trigger to balance retrieval and self-knowledge utilization.

2 Preliminaries

Retrieval-Augmented Generation (RAG). Given a query $\boldsymbol{q}$, RAG aims to assist LLMs in generating more precise answers $\boldsymbol{y} = [s_1, \ldots, s_m] = [w_1, \ldots, w_n]$ containing m sentences or n tokens by retrieving relevant documents $\mathcal{D}_q = \mathcal{R}(\boldsymbol{q})$ from document corpus $\mathcal{D} = \{\boldsymbol{d}_i\}_{i=1}^{|\mathcal{D}|}$ or web via

retriever $\mathcal{R}$. The retrieved documents $\mathcal{D}_q$ are usually concatenated with input $\boldsymbol{x}$, *i.e.,* query $\boldsymbol{q}$ with task instruction $\mathcal{I}$, to aid answer generation as $\boldsymbol{y} = \texttt{LLM}([\mathcal{D}_q; \boldsymbol{x}])$, where $[\cdot;\cdot]$ denotes concatenation.

As the indiscriminate retrieval nature, conventional RAG [56, 64] may neglect LLM's self-knowledge and introduce procedural overhead. To mitigate such inefficiency, adaptive RAG [4, 23] proposes to actively decide retrieval necessity via a trigger mechanism $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y}_{<t})$, where $\boldsymbol{y}_{<t}$ is the output segment as of step $t (t \geq 1)$. If $\mathcal{T}$ is triggered, the query formulation function $\boldsymbol{q}_t = f_q(\boldsymbol{x}, \boldsymbol{y}_{<t})$ will produce a query $\boldsymbol{q}_t$ to search. If $\mathcal{T}$ is triggered at $t = 1$, *i.e.,* $\boldsymbol{y}_{<1} = \emptyset$, $\boldsymbol{q}$ will be the original query. Given the retrieved documents $\mathcal{D}_{q_t}$, the model continues generating the next output segment $\boldsymbol{y}_t = \texttt{LLM}([\mathcal{D}_{q_t}; \boldsymbol{x}; \boldsymbol{y}_{<t}])$ till the answer comes to its end or next retrieval trigger occurs. Figure 1 illustrates the general pipeline of RAG methods.

Prompt 2.1: Function Instruction Template

[INST] `<experimental/reference instruction>` **[/INST]** `<a truncated statement>`

Representation Engineering (RepE). RepE [80] is a probing technique that enables reading and controlling high-level *functions* of `LLM` unsupervisedly. Given a function f, RepE designs an experimental instruction T_f^+ that necessitates the execution of the function and a corresponding reference instruction T_f^- that omits this function (ref. Prompt 2.1). Given a dataset $\mathcal{S} = \{\boldsymbol{s}_i\}_{i=1}^{|\mathcal{S}|}$ that contains a set of corresponding sentence statements, denoting $\boldsymbol{s}_i$ truncated after token k as $\boldsymbol{s}_i^k$, RepE then collects two sets of neural activity corresponding to the experimental and reference sets as:

$$\boldsymbol{A}_f^{\pm} = \left\{\texttt{Rep}(\texttt{LLM}, T_f^{\pm}(\boldsymbol{s}_i^k))[-1] \mid \boldsymbol{s}_i \in \mathcal{S}, \text{for } 0 < k \leq |\boldsymbol{s}_i|\right\}, \tag{1}$$

where -1 denotes the last token representation of $\boldsymbol{s}_i^k$, `Rep` is to obtain representation from `LLM`, and $\boldsymbol{A}_f^{\pm}$ consists of individual vectors. Finally, RepE constructs a linear model to identify a direction that accurately predicts the function using $\boldsymbol{A}_f^{\pm}$. Specifically, it applies PCA [43] to pair-wise difference vectors of the neural activity sets as $\texttt{PCA}\{(-1)^i(\boldsymbol{A}_{f,i}^+ - \boldsymbol{A}_{f,i}^-)\}$ and refer to the *first principal component* $\boldsymbol{v}_f$ as reading vector, *i.e.,* probe. Equation 1 is adopted at each layer of `LM` to derive layer-wise probe. The learned probes are directly used to interact with LLM's representations to control and monitor its behavior. More details of RepE are presented in § A.1.

3 Probe-Guided Control based Adaptive RAG

Research on retrieval necessity in ARAG primarily focused on evaluating verbalize-based and probability-based feedback from LLMs, which are assessed through direct querying [57], response uncertainties [9] or logit dominance analysis [23]. However, these methods may be ineffective when LLMs generate misleading responses, leading to inadequately reflecting internal beliefs [29]. Recent work [7, 42] indicates that LLM's internal state, *i.e.,* its representation, can reflect its authentic cognition more accurately. Motivated by this, we attempt to delve into the representational spaces of LLM to devise a more reliable retrieval necessity detector.

Despite the robustness of LLM's internal states, they may still be skewed by the cognitive biases of LLM, leading to inaccurate detection. Thus, it is crucial to regulate LLM's behavior, making it more honest and capable of admitting its limitations. Honesty alignment and prompt engineering are two prevalent solutions to achieve this goal. However, honesty alignment [73] requires training on numerous high-quality data to align this ability, and it will irreversibly alter model parameters, potentially negatively impacting LLM's general capabilities. Prompting methods may suffer difficulties in eliciting truly internal belief of LLM, as it may "lie" in its response.

To bridge this gap, we propose Probe-Guided **Con**t**r**o**l** based **A**daptive RAG (**CTRLA**). CTRLA first abstracts the honesty and confidence functions of LLMs and trains two corresponding plug-and-play probes. The honesty probe regulates LLM's behavior to make it honest. Upon the honesty controlled LLM, the confidence probe monitors the internal states of the LLM to detect the retrieval signal more precisely. To optimize the retrieval, we further devise a search query formulation strategy to reduce noise introduction and intent drift. The overall framework of CTRLA is illustrated in Figure 2.

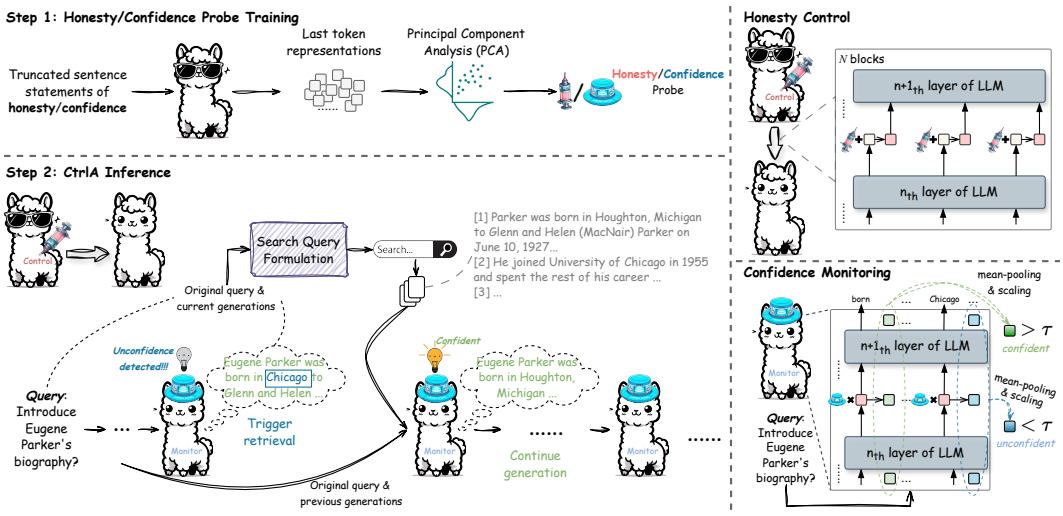

Figure 2: Overall framework of CTRLA. It contains two steps: Step 1 trains *honesty* and *confidence* probes; Step 2 directly utilizes the learned probes to control and monitor LLM behaviors at inference. The *honesty probe* **manipulates** the representation of LLM to make it more honest, while *confidence probe* **monitors** the confidence level of each token, the token whose score is lower than the threshold is marked as unconfident and the retrieval is triggered if specific tokens are unconfident.

3.1 Honesty Probe and Control

Honesty Probe. The goal of the honesty probe is to modulate LLM's behavior, ensuring it steadily elicits its knowledge and admits its lack of specific knowledge. As detailed in § 2, developing a function-specific probe involves creating task instructions and gathering indicative statements. We initially create a simple instruction for honesty function, `honest` and `untruthful`, in Prompt 3.1, and use True-False dataset [5] to formulate sentence statements. By coupling each statement with honesty instruction, we prompt LLM and collect representation vectors (ref. Equation 1) to train the probe. Note that the trained probe reflects LLM's cognition of function understanding and instruction adherence, regardless of the input data given. Hence, the probe is insensitive to data variance and can generalize across different tasks.

Prompt 3.1: Function Instruction Template for Honesty and Confidence

[INST] `Pretend you're a <honest/untruthful> | <confident/unconfident> person making statements about the world.` **[/INST]** `<a truncated statement>`

Honesty Control. After obtaining the honesty probe, it can be directly used to control LLM behavior. Assuming the LLM contains L layers, each layer has its corresponding probe. Let $\boldsymbol{v}_h = \{\boldsymbol{v}_{h,l}\}_{l=1}^{L}$ be the honesty probe and $\boldsymbol{R}_i = \{\boldsymbol{r}_{i,l}\}_{l=1}^{L}$ be the representations of the i-th token at each layer. Honesty control can be achieved via a simple linear combination function:

$$\hat{\boldsymbol{R}}_i = \boldsymbol{R}_i + \lambda \cdot \boldsymbol{v}_h = \{\boldsymbol{r}_{i,l} + \lambda \cdot \boldsymbol{v}_{h,l} \mid \forall\, l \in [1, \ldots, L]\}, \tag{2}$$

where coefficient λ controls the strength of honesty intervention. Since $\boldsymbol{v}_h$ represents the direction that makes LLM more honest, we use the "+" in Equation 2. Conversely, we may use "−" if we want the LLM to be dishonest (ref. § A.1). As shown in Figure 2, honesty control is executed layer-by-layer and token-by-token during the generation. Despite its simplicity, honesty control is effective with negligible impact on inference costs. For simplicity, we directly use $\hat{\boldsymbol{y}}_t = \mathcal{P}_h(\boldsymbol{y}_t)$ to represent honesty control in the following descriptions.

3.2 Confidence Probe and Retrieval Trigger

Confidence Probe. Assessing the confidence level of LLMs plays a crucial role in determining retrieval necessity. Instead of prompting or fine-tuning LLM, we monitor confidence level in

representational space that can more precisely reflect LLM's cognition, especially after implementing honesty control. Specifically, we aim to directly extract a probe that indicates LLM's confidence orientation or preference towards a more precise confidence evaluation.

Similar to the honesty probe, we develop a confidence probe that ascertains LLM confidence through analysis of the directional preferences in its representations. We craft an instruction template for the confidence function, *i.e.,* `confident` and `unconfident` (ref. Prompt 3.1). Due to the absence of datasets to reflect the confidence statement of LLM, we create a simple dataset containing a set of confident and unconfident statements through prompting GPT-4 (see § B.1) and being evaluated by human. Then we adopt PCA to train the probe using the representation vectors collected from LLM.

Retrieval Trigger. Given the learned confidence probe, we utilize it to monitor the confidence level of LLM during the generation process. Let $\boldsymbol{R}_i = \{\boldsymbol{r}_{t,l}\}_{l=1}^{L}$ denote the representation of t-th token at each layer and $\boldsymbol{v}_c = \{\boldsymbol{v}_{c,l}\}_{l=1}^{L}$ be the confidence probe. Unlike the honesty probe, which directly intervenes in the LLM behavior by altering its representations via linear combination, we only use the confidence probe to detect the confidence direction of each token generated by LLM. Specifically, we adopt the dot product to compute the scores of i-th token followed by mean-pooling and a scaling operation over all mean-pooled preceding token scores for normalization and outlier removal to compute i-th token's function score as:

$$\begin{aligned} \tilde{m}_i &= \texttt{meanpool}([m_{i,1}, \ldots, m_{i,L}]) = \texttt{meanpool}\big([\boldsymbol{r}_{i,l}^{\top} \cdot \boldsymbol{v}_{c,l}]_{l=1}^{L}\big), \\ \bar{m}_i &= \texttt{scale}([\tilde{m}_0, \ldots, \tilde{m}_i])[-1] - \tau, \end{aligned} \tag{3}$$

where threshold τ is used to adjust the sensitivity of confidence monitoring, $\tilde{m}_{<i}$ denotes the mean-pooled score of previous tokens, and -1 means obtaining the score of last token, *i.e.,* i-th token. $\bar{m}_i > 0$ suggests that i-th token's representational direction leans towards the confident function, indicating that LLM is confident in generating i-th token. Conversely, $\bar{m}_i < 0$ means LLM is unconfident in generating i-th token. We use $\mathcal{P}_c$ to denote confidence monitoring. For t-th output segment $\hat{\boldsymbol{y}}_t = [w_{t_s}, \ldots, w_{t_e}]$ of LLM and its confidence scores for each token $[\bar{m}_{t_s}, \ldots, \bar{m}_{t_e}]$, the retrieval necessity is measured by confidence scores of specific tokens in $\hat{\boldsymbol{y}}_t$. We only consider confidence scores of *new information* $\hat{\boldsymbol{y}}_t'$, *i.e.,* contents that have not appeared in previous generation and are not trivial tokens, like stopwords. The retrieval trigger $\mathcal{T}$ detects if any confidence score of token in $\hat{\boldsymbol{y}}_t'$ satisfies $\bar{m}_i < 0$, where $t_s \leq i \leq t_e$. If $\hat{\boldsymbol{y}}_t'$ contains such tokens, the retrieval is triggered accordingly, *i.e.,* $\mathcal{T}(\mathcal{P}_c(\hat{\boldsymbol{y}}_t')) == \texttt{True}$.

3.3 Search Query Formulation

Upon retrieval activation, we need to employ a search query to retrieve relevant documents that aids in LLM generation. The construction of effective search queries plays a pivotal role in enhancing retrieval efficiency. Additionally, the quality of retrieved documents further affects the resultant model quality. Thus, developing a robust strategy for search query formulation is imperative.

Context-Augmented Querying. Initially, for a query $\boldsymbol{q}$, we prompt the LLM to sequentially generate responses. Once the retrieval is triggered, Context-augmented querying (CAQ) will concatenate the query $\boldsymbol{q}$ with the processed output segment $\hat{\boldsymbol{y}}_t$ for retrieval, since using the original query as a supplement can avoid intent drift and improve the effectiveness of retrieval [19]. Besides, the output segment $\hat{\boldsymbol{y}}_t = [w_{t_s}, \ldots, w_{t_e}]$ may contain noise such as unconfident tokens and incorrect contents, we process the sentence by masking out the tokens, which satisfy (i) not appeared in $\boldsymbol{q}$ and previous generations $\boldsymbol{y}_{<t}$, *i.e.,* new information and (ii) unconfident tokens, as:

$$\texttt{mask}(\hat{\boldsymbol{y}}_t) = \left\{ \bar{w} \middle| \bar{w} = \begin{cases} \emptyset, & \text{if } w \notin \boldsymbol{q} \cup \boldsymbol{y}_{<t} \text{ and } \bar{m}_w < 0 \\ w, & \text{otherwise} \end{cases}, \forall w \in \hat{\boldsymbol{y}}_t \right\}. \tag{4}$$

Thus, the CAQ generates the refined search query as $f_{\text{CAQ}}(\boldsymbol{x}, \hat{\boldsymbol{y}}_t) = [\boldsymbol{q}; \texttt{mask}(\hat{\boldsymbol{y}}_t)]$.

Targeted Validation Querying. CAQ directly masks out the noise of the output segment and concatenates it with the original query to form a search query. Yet, off-the-shelf retrievers may prefer a well-formatted query [25]. Thus, we also develop a targeted validation querying strategy (TVQ), f_{TVQ}. It instructs LLM to produce search query using original query and current output segment as

references (see Prompt A.2). The goal of TVQ is to generate a query to validate the accuracy of current output segment by searching for supporting documents. For simplicity, we use f_q to represent both f_{CAQ} and f_{TVQ}.

3.4 Inference Overview

For an input $\boldsymbol{x}$ and preceding generation $\boldsymbol{y}_{<t}$, the model generates the output segment along with honesty control $\mathcal{P}_h$ and derives $\hat{\boldsymbol{y}}_t$. Simultaneously, confidence probe $\mathcal{P}_c$ is activated to compute the confidence score of each token during generation. We collect the confidence scores of new information $\hat{\boldsymbol{y}}'_t$ to determine retrieval necessity via retrieval trigger $\mathcal{T}$. If retrieval is not required, the model continues predicting next output segment. If retrieval is triggered, we adopt query formulation, f_q, to produce search query $\boldsymbol{q}_t$ and retrieve documents $\mathcal{D}_q$ via retriever $\mathcal{R}$. The retrieved documents $\mathcal{D}_q$, input $\boldsymbol{x}$, and preceding generation $\boldsymbol{y}_{<t}$ are concatenated to regenerate the current output segment. Algorithm 1 presents an overview of CTRLA inference step. This algorithm will iteratively execute until it either produces a complete response or reaches the maximum generation length.

Algorithm 1 CTRLA Inference

Require: Language Model `LM`, Retriever $\mathcal{R}$, Document Corpus $\mathcal{D}$, Honesty Probe $\mathcal{P}_h$, Query Formulator f_q, Retrieval Trigger $\mathcal{T}$

Input: input prompt $\boldsymbol{x}$ ($\mathcal{I}$ and $\boldsymbol{q}$), previous generation $\boldsymbol{y}_{<t}$
Output: next output segment $\boldsymbol{y}_t$
`LLM` along with $\mathcal{P}_h$ predicts next segment $\hat{\boldsymbol{y}}_t$ given $(\boldsymbol{x}, \boldsymbol{y}_{<t})$
$\mathcal{T}$ simultaneously monitors retrieval signal during `LLM` generates $\hat{\boldsymbol{y}}_t$
if $\mathcal{T}$ == `True` **then**
    $\mathcal{R}$ retrieves $\mathcal{D}_q$ from $\mathcal{D}$ using $\boldsymbol{q}_t = f_q(\boldsymbol{q}, \hat{\boldsymbol{y}}_t)$
    `LM` along with $\mathcal{P}_h$ re-predicts next segment $\hat{\boldsymbol{y}}_t$ given $(\boldsymbol{x}, \boldsymbol{y}_{<t}, \mathcal{D}_q)$
end if
Set $\boldsymbol{y}_t = \hat{\boldsymbol{y}}_t$

4 Experiment Setup

Datasets and Evaluation. We evaluate CTRLA under zero-shot setting. For the short-form generation task, we select PopQA [40] and TriviaQA [24]. We select the long-tail subset with $1,399$ queries related to rare entities for PopQA and follow prior work [4, 45] to use $11,313$ test queries in TriviaQA. The accuracy is used to measure PopQA and TriviaQA. For the long-form generation task, we use ASQA [62] and biography generation (Bio) [46]. We follow Self-RAG [4] to evaluate ASQA on 948 queries in the dev set and report the str-em, str-hit, Rouge-L [34] (R-L), MAUVE [53] (mau), EM and F1 scores. Bio is evaluated by FactScore (FS) [46]. We also evaluate on 500 test samples (`v04082024`) of FreshQA [65] and report relaxed and strict accuracy scores.

Implementation and Retrieval Setup. We select the Mistral-7B [22] as the backbone of CTRLA, and adopt the greedy decoding strategy for all experiments. The coefficient λ of honesty control is set as 0.3. The threshold τ of confidence monitoring is set as 0.0. By default, we adopt BGE [71] retriever to retrieve top-5 documents and use the official 2018 English Wikipedia corpus as per prior work [4, 23]. However, due to the absence of articles in the 2018 version, for PopQA, TriviaQA, and Bio, we follow Self-RAG [4] to additionally retrieve from the web to mitigate the coverage limitations in the 2018 version. For FreshQA, we only retrieve from the web to obtain supporting documents.

Table 2: Overall results on FreshQA [65].

Retrieval Type	Model	Accuracy (%)	
		Relaxed	Strict
ST	Mistral_{7B} [22]	38.4	33.0
MT	PreWind [56]	31.2	27.4
	PreSent [64]	22.8	20.6
	QDecomp [54]	26.4	24.0
ADA	FLARE [23]	41.6	39.8
	Our **CTRLA**	**48.4**	**43.8**

Baselines. We compare CTRLA with four baselines: (1) *No Retrieval*, where LLMs generate answers directly. (2) *Single-time Retrieval*, which prepend query with retrieved documents to generate answers. (3) *Multi-time Retrieval*, we follow FLARE [23] to reimplement *Previous-window* (PreWind) [56], *Previous-sentence* (PreSent) [64] and *Query decomposition* (QDecomp) [27, 54]. PreWind triggers retrieval of every l token (*i.e.,* window size) and is used for retrieval, we set

Table 1: Overall results on TriviaQA, PopQA, ASQA, and Bio. $^{\diamond}$ means our reproduced results. † indicates results reported by Self-RAG [4]. ‡ denotes results reported in the corresponding work.

Retrieval Type	Model	TriviaQA	PopQA	ASQA						Bio
		Acc (%)	Acc (%)	str-em	str-hit	R-L	EM	F1	mau	FS
No	LLaMA2$_{7B}$ [63]†	30.5	14.7	7.9	-	15.3	-	-	19.0	44.5
	LLaMA2$_{13B}$ [63]†	38.5	14.7	7.2	-	12.4	-	-	16.0	53.4
	Alpaca$_{7B}$ [10]†	54.5	23.6	18.8	-	29.4	-	-	61.7	45.8
	Alpaca$_{13B}$ [10]†	61.3	24.4	22.9	-	32.0	-	-	70.6	50.2
	Mistral$_{7B}$ [22]$^{\diamond}$	53.8	25.7	18.8	4.6	33.7	8.7	13.7	23.8	41.9
Single-time (ST)	LLaMA2$_{7B}$ [63]†	42.5	38.2	15.2	-	22.1	-	-	32.0	78.0
	LLaMA2$_{13B}$ [63]†	47.0	45.7	16.3	-	20.5	-	-	24.7	77.5
	Alpaca$_{7B}$ [10]†	64.1	46.7	30.9	-	33.3	-	-	57.9	76.6
	Alpaca$_{13B}$ [10]†	66.9	46.1	34.8	-	36.7	-	-	56.6	77.7
	Mistral$_{7B}$ [22]$^{\diamond}$	62.7	51.9	32.4	10.8	34.9	18.7	25.1	54.7	78.6
Multi-time (MT)	PreWind [56]$^{\diamond}$	60.8	28.1	24.4	6.6	34.4	11.2	16.7	26.5	56.9
	PreSent [64]$^{\diamond}$	54.3	26.9	25.9	7.5	32.9	11.3	16.9	44.8	57.5
	QDecomp [54]$^{\diamond}$	52.3	29.4	18.1	3.3	18.6	8.4	12.3	-	22.4
Adaptive (ADA)	FLARE [23]$^{\diamond}$	72.4	48.3	29.9	9.6	35.2	16.2	22.2	50.4	74.8
	Self-RAG$_{7B}$ [4]‡	66.4	54.9	30.0	-	35.7	-	-	74.3	81.2
	Self-RAG$_{13B}$ [4]‡	69.3	55.8	31.7	-	37.0	-	-	71.6	80.2
	Self-RAG$_{\text{Mistral}_{7B}}$ [59]‡	65.8	51.5	-	-	-	-	-	-	-
	RQ-RAG [8]‡	-	57.1	-	-	-	-	-	-	-
	Adaptive-RAG [20]‡	58.2	-	-	-	-	-	-	-	-
	Our **CTRLA$_{7B}$**	**76.4**	**61.8**	**37.0**	**14.3**	**38.5**	**20.4**	**27.3**	**79.2**	**83.4**

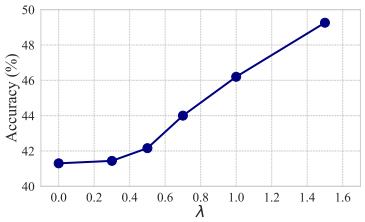

Figure 3: Effects of honesty control on TruthfulQA.

Table 3: Performance comparison between the honesty control and the honesty prompt (HonP) on PopQA and ASQA datasets.

λ	PopQA	ASQA					
	Acc (%)	str-em	str-hit	R-L	EM	F1	mau
$\lambda = 0.0$	58.5	36.8	13.9	38.1	**20.6**	27.0	76.5
$\lambda = 0.3$	**61.8**	**37.0**	**14.3**	**38.5**	20.4	**27.3**	**79.2**
HonP	60.2	36.8	13.3	38.3	20.0	27.0	71.5

$l = 16$ [56]. PreSent triggers retrieval of every sentence and uses the previous sentence as query. QDecomp prompts LLMs to generate follow-up queries and trigger retrieval for each query. We reimplement them using the same setting as CTRLA. (4) For *Adaptive Retrieval*, we select FLARE [23], Self-RAG [4], RQ-RAG [8] and Adaptive-RAG [20]. A more detailed introduction of the dataset, evaluation metrics, experiment setup, and baselines is presented in Appendix B.

5 Experiment Results and Analysis

5.1 Main Results

Table 1 summarizes the overall results of various baselines. Without retrieval, instruction-tuned LLMs, such as Alpaca$_{7B}$ and Mistral$_{7B}$, consistently outperform vanilla LLMs like LLaMA2$_{7B\&13B}$ across all datasets. These models demonstrate better comprehension ability and task flexibility, thus, Mistral$_{7B}$ is selected as our backbone model. Additionally, incorporating single-time retrieval into LLMs markedly enhances performance, *e.g.,* Mistral$_{7B}$, yielding absolute improvements of 8.9% on TriviaQA and 26.2% on PopQA. This improvement is attributed to the supplementary evidence provided through retrieval that compensates for the internal knowledge limitations of LLMs.

Our CTRLA consistently outperforms single-time and multi-time retrieval, since its adaptive mechanism, which retrieves evidence only when it is needed, can optimize the use of internal and external knowledge. Notably, single-time retrieval yields better results than multi-time retrieval, which may be attributed to the latter's tendency to suffer from intent drift and noise due to suboptimal generated queries, leading to irrelevant information retrieved. Additionally, they cannot correct previous errors, struggle to filter out noise, and tend to be overconfident in unreliable external knowledge.

From Table 1, CTRLA surpasses ARAG baselines across all datasets and evaluation metrics. Specifically, Adaptive-RAG, Self-RAG, and RQ-RAG either train an additional classifier or directly fine-tune LLM on crafted datasets for capability injection. The limited efficacy of fine-tuning for retrieval timing may not accurately represent LLM's internal cognition but rather adapt to specific distributions like query complexity and attributes. FLARE relies on logit probabilities to judge retrieval timing, which may not accurately represent LLM's true cognition due to the gap between LLM's output and its internal beliefs. Besides, these probabilities can be misleading as LLMs might express identical answers differently, affecting the reliability of logit dominance as a retrieval indicator. This issue is especially critical in long-form QA like ASQA and Bio, where potential misfires and intent drift can exacerbate errors. CTRLA not only improves the accuracy of retrieval timing decisions but also ensures an effective balance between internal and external knowledge usage.

Table 4: Effects of honesty control on PopQA and ASQA. * Only 2018 Wikipedia corpus is used as retrieval source for PopQA.

λ	PopQA*	ASQA					
	Acc (%)	str-em	str-hit	R-L	EM	F1	mau
0.0	43.9	37.1	13.9	38.1	20.6	27.0	76.5
0.2	44.4	37.2	14.2	38.4	20.9	27.8	85.5
0.3	44.1	37.0	14.3	38.5	20.4	27.3	79.2
0.4	44.2	37.0	15.1	38.5	20.9	27.3	77.5
0.5	44.5	37.0	14.9	38.8	20.8	27.3	75.7
0.6	44.5	37.0	14.8	38.9	21.3	27.8	76.3
0.7	45.1	37.1	14.6	38.8	20.8	27.1	73.9
0.8	44.6	37.2	15.1	38.8	20.9	27.0	72.1
1.0	43.7	37.2	14.8	39.1	20.2	26.5	66.4
1.5	44.2	34.7	12.7	38.5	18.2	23.9	53.2

Table 2 reports the results on FreshQA, where all baselines are reimplemented under the same settings. Similar observations hold that CTRLA is superior to the compared single-time, multi-time, and adaptive RAG baselines. The FreshQA contains more diverse question types, including never-changing, slow-changing, fast-changing, and false-premise questions, as well as single-hop and multi-hop questions, demonstrating the generalization capabilities of CTRLA. Besides, we only retrieve supporting documents from the web for FreshQA, indicating that CTRLA can adapt to different retrieval tools.

5.2 In-Depth Analysis

Effectiveness and transferability of honesty probe. The honesty probe is trained in an unsupervised manner using the dataset from Azaria and Mitchell [5]. To verify its effectiveness and transferability, we evaluate TruthfulQA [35] using $\text{Mistral}_{7\text{B}}$ without retrieval. Illustrated in Figure 3, it shows that enhancing the intensity of honesty control, by raising λ, consistently improves the performance, where $\lambda = 0.0$ means no control is applied. The improvements are primarily attributed to LLM's propensity to mimic human deceptive behaviors, and honesty control can effectively bridge the gap between LLM's output and its internal beliefs, which underscores the crucial importance of honesty control in boosting LLM's truthfulness and performance. Table 3 further compares the honesty control and honesty prompt, *i.e.,* an instruction that asks LLM to be honest. The honesty control surpasses the honesty prompt on both PopQA and ASQA, demonstrating the effectiveness of honesty control. Results on knowledge-intensive tasks are in Table 4. In general, the honesty probe exhibits reasonable transferability for downstream tasks.

Impacts of coefficient λ. Here we evaluate the impacts of different λ value choices, which govern the magnitude of honesty control. Table 4 indicates that honesty control, *i.e.,* $\lambda > 0.0$, generally contributes to performance improvements. As λ increases, performance initially rises and then either gradually decreases or fluctuates, differing from the results shown in Figure 3. Compared to closed-domain QA, the varying levels of honesty control may affect retrieval behaviors, and the incorporation of external information also affects LLM's generation, resulting in diverse outcomes.

Effectiveness of confidence probe. Similar to the honesty probe, the confidence probe is unsupervisedly trained on our synthetic dataset. To verify its effectiveness, we sample 50 unanswerable

Table 5: Confusion matrix of human evaluation results on answerable and unanswerable samples.

Ground Truth	LM Prediction	
	A_{Y}	A_{N}
A_{Y}	47	3
A_{N}	8	42

Table 6: Effects of different choices of τ on TriviaQA and ASQA. * Only 2018 Wikipedia corpus is used for TriviaQA.

τ	TriviaQA*			ASQA					
	Acc (%)	Freq (%)	N_{Acc} (%)	str-em	R-L	EM	F1	mau	Times
-0.10	69.5	84.8	77.0	37.0	38.6	20.2	27.0	78.5	3.81
-0.05	70.0	92.7	80.8	37.1	38.5	20.4	27.4	80.6	3.99
$+0.00$	70.8	97.1	86.0	37.0	38.5	20.4	27.3	79.2	4.07
$+0.05$	70.5	99.1	100.0	36.9	38.5	20.2	27.3	79.0	4.09
$+0.10$	70.4	99.8	100.0	36.8	38.5	20.4	27.4	80.3	4.11

Table 7: Performance comparison of different query formulation strategies on PopQA and ASQA. * $\boldsymbol{q}$: original question; f_{CAQ}: context-augmented querying; f_{TVQ}: targeted validation querying; I_{old}: old information. Only the 2018 Wikipedia corpus is used as the retrieval source for PopQA.

Query Formulation*	PopQA*		ASQA									
	Acc (%)		str-em		R-L		EM		F1		mau	
	BGE	BM25	BGE	BM25	BGE	BM25	BGE	BM25	BGE	BM25	BGE	BM25
f_{CAQ}	40.3	38.2	32.8	27.2	34.6	35.5	17.1	14.4	23.0	19.5	55.6	63.6
$\boldsymbol{q} + f_{\text{CAQ}}$	41.8	**39.5**	35.4	**29.6**	37.9	**36.5**	19.4	**15.6**	25.7	**21.6**	73.0	**72.8**
$\boldsymbol{q} + f_{\text{CAQ}} - I_{\text{old}}$	40.2	38.5	36.7	28.4	38.2	36.3	20.2	15.2	26.3	20.8	70.6	71.1
f_{TVQ}	**44.1**	37.7	36.0	28.0	38.3	35.8	20.0	15.0	25.9	20.9	77.3	69.3
$\boldsymbol{q} + f_{\text{TVQ}}$	43.7	**39.5**	**37.0**	28.5	**38.5**	36.3	**20.4**	15.4	**27.3**	21.1	**79.2**	68.7

(A_{N}) from Self-Aware [74] and craft 50 answerable (A_{Y}) questions (detailed in § C.2) for evaluation. We summarize the human evaluation results in Table 5, which shows that the confidence probe exhibits high accuracy in identifying A_{Y} and A_{N} cases. In general, it generally detects that LLM is unconfident on unanswerable questions and vice versa, which demonstrates its effectiveness to be the retrieval necessity indicator.

Impacts of threshold τ. The coefficient τ adjusts the sensitivity of confidence monitoring. Table 6 evaluates the impacts of different τ values. It shows that increasing τ leads to higher retrieval frequency and times, but performance first improves and then declines. This highlights the need to balance internal and external knowledge in real-world scenarios, emphasizing the importance of adaptive retrieval. N_{Acc} represents the proportion of questions that the model correctly answers among the non-retrieval questions determined by the confidence probe. The results indicate that the confidence probe has high accuracy in detecting the retrieval necessity since most of the non-retrieval questions are indeed answered correctly by LLM.

Analysis on search query formulation. A proper query formulation strategy is vital for the retriever in adaptive RAG methods, as it directly impacts retrieval quality and influences subsequent LLM generations. Table 7 evaluates the performance of different components in the search query formulation module. Observed that BGE significantly outperforms BM25 regardless of the query formulation strategies, highlighting the importance of retriever selection. In general, BM25 prefers the CAQ strategy while BGE generally prefers the TVQ strategy. Since BM25 is a sparse retriever that performs retrieval via keyword matching, making it insensitive to the query format, while BGE is a dense retriever, the incomplete query format produced by CAQ may hinder its retrieval performance. Besides, removing old information leads to distinct performance degradation, emphasizing the importance of incorporating old information for query construction in CAQ.

Case studies. Honesty control can effectively mitigate both narrow-sense lying and unconscious deception [80]. Figure 4 (top) depicts LLM's responses with and without honesty control. Through honesty control, when LLM lacks specific knowledge of the question or only irrelevant content is provided, it tends to acknowledge its limitations or declare the absence of relevant knowledge in responses, rather than resorting to speculation, *i.e.,* "lying", or overconfident in the provided information. Depicted in Figure 4 (bottom), confidence probe demonstrates its capability to effectively detect the LLM's confidences. The confidence probe can identify the LLM's uncertainty when it

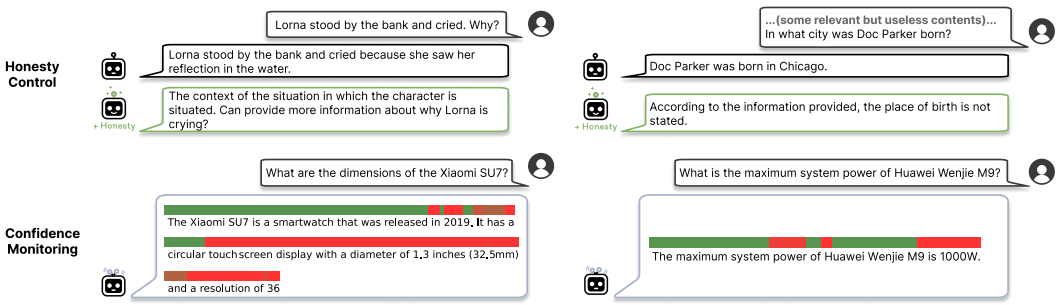

Figure 4: Examples of honesty control (top) and confidence monitoring (bottom). Honesty control can regulate the LLM behavior, ensuring it elicits the correct recognition with respect to the question. Confidence probe effectively recognizes the unconfident outputs (*marked in red*) at token-level.

encounters unknown questions, which refers to questions for which the LLM lacks relevant knowledge like questions about recent events. More cases are shown in Appendix § C.3 and § C.4.

6 Related Work

Retrieval-augmented generation (RAG) shows effectiveness to mitigate hallucinations [11, 25, 30, 56, 79]. Conventional RAG incorporates external evidences to address LMs' knowledge constraints via various strategies like task decomposition [27, 28, 39, 58], calibration [33, 52, 72, 78], collaborative refinements [60, 70] or jointly optimizing retriever and LM via fine-tuning [6, 36, 47, 51, 61, 66, 67, 75]. WebGPT [48] and WebCPM [55] also apply reinforcement learning for web interaction simulation. Since LLMs already possess extensive knowledge, indiscriminate retrieval of conventional RAG may lead to inefficiency and high costs [13, 41, 77]. Adaptive RAG addresses this by discerning when retrieval is necessary. kNNLM [15] uses a lightweight adaptor, and Self-RAG [4] fine-tunes LMs to retrieve information using special tokens. RA-ISF [38] fine-tunes LMs to determine retrieval necessity via self-knowledge reflection. FLARE [23] and Self-DC [68], rely on logit probabilities for retrieval timing, while some work [9, 31] use generation consistency as a trigger. Instruction-based adaptive RAG [49, 69, 76] is also developed.

7 Conclusion, Limitations, and Future Work

This work introduces CTRLA, an effective and lightweight framework to optimize the retrieval necessity detection of adaptive RAG. CTRLA trains two plug-and-play probes, *i.e.,* honesty and confidence probes, to regulate LLM's behavior and monitor the confidence of its internal states to detect retrieval necessity during generation. CTRLA further employs a search query formulation to produce refined search queries when retrieval is triggered and a refusal handing module to tackle LLM's refusal outputs. Our holistic evaluation of several benchmark datasets and in-depth analyses show CTRLA is superior to compared baselines and demonstrates the effectiveness and necessity of both honesty and confidence probes.

Limitations and future work. CTRLA is a preliminary exploration of solving adaptive RAG from a representational perspective. To ensure our research is succinct, transparent, and easily attributable, we adopt a straightforward yet consistent and elegant strategy for training probes and modulating the behavior of LLM, yielding promising results. Recent work [37] has shown that fine-tuning LLMs can produce more effective probes for model alignment, which could further enhance the performance of CTRLA. Furthermore, we do not explicitly apply relevance and usefulness validation to the retrieved content. However, since CTRLA does not involve fine-tuning the LLM and achieves adaptive RAG in a plug-and-play fashion, it can be effortlessly integrated with other approaches focused on content processing. The exploration of these additional aspects is reserved for future research.

Broader impacts. CtrlA investigates the usability of probing techniques in the RAG domain and their compatibility with long contexts and In-Context Learning. Unlike prior work that relied on simple settings, such as directly querying, we delve into the representational space to analyze the essence of retrieval timing from more general angles of confidence and honesty. We introduce the

concept of a confidence probe to directly assess confidence feedback at the representational space during the LLM's response. Furthermore, we extensively explore the query forms in adaptive RAG and their compatibility with different types of retrievers through comprehensive experiments.

CTRLA is an adaptive RAG framework that enables LLMs to make more reliable and accurate retrieval necessity decisions, thereby optimizing the internal and external knowledge use. High training or fine-tuning costs often limit LLM usage for specific tasks, our research shows notable performance improvements in RAG using a low-cost, simple inference approach. Considering the broader implications of integrating retrieved knowledge into LLMs is crucial. Although current tasks are limited to specific datasets and reliable sources like Wikipedia, future applications in open-world environments may risk retrieving useless or biased content. On the other hand, CTRLA exhibits the capability of rejecting and detecting irrelevant contents, which may mitigate this issue, resulting in a more acceptable response, such as refusal in particular cases.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *ArXiv*, abs/2309.14316, 2023.

[3] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *ArXiv*, abs/2309.14402, 2023.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Amos Azaria and Tom Mitchell. The internal state of an llm knows when its lying. *ArXiv*, abs/2304.13734, 2023.

[6] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 2206–2240. PMLR, 17–23 Jul 2022.

[7] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. *ArXiv*, abs/2404.00610, 2024.

[9] Hanxing Ding, Liang Pang, Zihao Wei, Huawei Shen, and Xueqi Cheng. Retrieve only when it needs: Adaptive retrieval augmentation for hallucination mitigation in large language models. *ArXiv*, abs/2402.10612, 2024.

[10] Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 30039–30069. Curran Associates, Inc., 2023.

[11] Zhangyin Feng, Weitao Ma, Weijiang Yu, Lei Huang, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting liu. Trends in integration of knowledge and large language models: A survey and taxonomy of methods, benchmarks, and applications. *ArXiv*, abs/2311.05876, 2023.

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

[13] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997, 2024.

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3929–3938. PMLR, 13–18 Jul 2020.

[15] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714. Association for Computational Linguistics, 2021.

[16] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[17] Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. Do large language models know about facts? In *The Twelfth International Conference on Learning Representations*, 2024.

[18] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

[19] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. *ArXiv*, abs/2305.03653, 2023.

[20] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *ArXiv*, abs/2403.14403, 2024.

[21] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[22] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *ArXiv*, abs/2310.06825, 2023.

[23] Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992. Association for Computational Linguistics, 2023.

[24] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *ArXiv*, abs/1705.03551, 2017.

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

[26] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *ArXiv*, abs/1911.00172, 2019.

[27] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *ArXiv*, abs/2212.14024, 2023.

[28] Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478. Association for Computational Linguistics, 2022.

[29] Benjamin A. Levinstein and Daniel A. Herrmann. Still no lie detector for language models: Probing empirical and conceptual roadblocks. *ArXiv*, abs/2307.00175, 2023.

[30] Patrick 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[31] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jingyuan Wang, Jian-Yun Nie, and Ji-Rong Wen. The web can be your oyster for improving language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 728–746. Association for Computational Linguistics, 2023.

[32] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[33] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics, 2004.

[35] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, 2022. Association for Computational Linguistics.

[36] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024.

[37] Wenhao Liu, Xiaohua Wang, Muling Wu, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Aligning large language models with human preferences through representation engineering. *ArXiv*, abs/2312.15997, 2023.

[38] Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. Ra-isf: Learning to answer and understand from retrieval augmentation via iterative self-feedback. *ArXiv*, abs/2403.06840, 2024.

[39] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[40] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *ArXiv*, abs/2212.10511, 2022.

[41] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 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)*, pages 9802–9822. Association for Computational Linguistics, 2023.

[42] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *ArXiv*, abs/2310.06824, 2023.

[43] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[44] Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023.

[45] Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. A discrete hard em approach for weakly supervised question answering. *ArXiv*, abs/1909.04849, 2019.

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

[47] Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. Nonparametric masked language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2097–2118. Association for Computational Linguistics, 2023.

[48] 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. Webgpt: Browser-assisted question-answering with human feedback. *ArXiv*, abs/2112.09332, 2022.

[49] Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. When do llms need retrieval augmentation? mitigating llms' overconfidence helps retrieval augmentation. *ArXiv*, abs/2402.11457, 2024.

[50] OpenAI. Chatgpt: Optimizing language models for dialogue, 2023. URL `https://openai.com/blog/chatgpt`.

[51] Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? comparing knowledge injection in llms. *ArXiv*, abs/2312.05934, 2024.

[52] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *ArXiv*, abs/2302.12813, 2023.

[53] Krishna Pillutla, Lang Liu, John Thickstun, Sean Welleck, Swabha Swayamdipta, Rowan Zellers, Sewoong Oh, Yejin Choi, and Zaid Harchaoui. Mauve scores for generative models: Theory and practice. *Journal of Machine Learning Research*, 24(356):1–92, 2023.

[54] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711. Association for Computational Linguistics, 2023.

[55] Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, Ruobing Xie, Fanchao Qi, Zhiyuan Liu, Maosong Sun, and Jie Zhou. WebCPM: Interactive web search for Chinese long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8968–8988. Association for Computational Linguistics, 2023.

[56] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

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

[58] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

[59] SciPhi-AI, 2024. URL `https://huggingface.co/SciPhi/SciPhi-Self-RAG-Mistral-7B-32k`.

[60] Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. WikiChat: Stopping the hallucination of large language model chatbots by few-shot grounding on Wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2387–2413. Association for Computational Linguistics, 2023.

[61] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models. *ArXiv*, abs/2301.12652, 2023.

[62] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. Asqa: Factoid questions meet long-form answers. *ArXiv*, abs/2204.06092, 2022.

[63] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. 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, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[64] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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)*, pages 10014–10037. Association for Computational Linguistics, 2023.

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

[66] Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, Anima Anandkumar, and Bryan Catanzaro. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7763–7786, 2023.

[67] Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. Instructretro: Instruction tuning post retrieval-augmented pretraining. *ArXiv*, abs/2310.07713, 2024.

[68] Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Guanhua Chen, Huimin Wang, and Kam fai Wong. Self-dc: When to retrieve and when to generate? self divide-and-conquer for compositional unknown questions. *ArXiv*, abs/2402.13514, 2024.

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

[70] Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. Rat: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. *ArXiv*, abs/2403.05313, 2024.

[71] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. C-pack: Packaged resources to advance general chinese embedding. *ArXiv*, abs/2309.07597, 2023.

[72] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *ArXiv*, abs/2401.15884, 2024.

[73] Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. *ArXiv*, abs/2312.07000, 2023.

[74] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665. Association for Computational Linguistics, 2023.

[75] Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. Raft: Adapting language model to domain specific rag. *ArXiv*, abs/2403.10131, 2024.

[76] Zihan Zhang, Meng Fang, and Ling Chen. Retrievalqa: Assessing adaptive retrieval-augmented generation for short-form open-domain question answering. *ArXiv*, abs/2402.16457, 2024.

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

[78] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840. Association for Computational Linguistics, 2023.

[79] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *ArXiv*, abs/2101.00774, 2021.

[80] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *ArXiv*, abs/2310.01405, 2023.

Appendix

A More Details about CTRLA Framework

A.1 Technical Details of Representation Engineering

Representation Engineering (RepE) [80] is a probing technique that enables reading and controlling high-level *concepts* or *functions* of a language model LM in an unsupervised manner. Note that the *concept* represents the declarative knowledge of a language model while the *function* denotes procedural knowledge. As our goal is to control and monitor the behaviors of LM, *i.e.,* honesty and confidence, we focus on eliciting the procedural knowledge from the model in this work. Given a function f, RepE first designs an experimental instruction T_f^+ that necessitates the execution of the function and a corresponding reference instruction T_f^- that omits this function (ref. Prompt A.1). Given a relevant data corpus $\mathcal{S} = \{\boldsymbol{s}_i\}_{i=1}^{|\mathcal{S}|}$ that contains a set of sentence statements, denoting a sentence $\boldsymbol{s}_i$ truncated after token k as $\boldsymbol{s}_i^k$, RepE then collects the neural activities of the experimental and reference sets as:

$$\boldsymbol{A}_f^{\pm} = \left\{\texttt{Rep}(\texttt{LM}, T_f^{\pm}(\boldsymbol{s}_i^k))[-1] \mid \boldsymbol{s}_i \in \mathcal{S}, \text{for } 0 < k \leq |\boldsymbol{s}_i|\right\}, \tag{5}$$

where -1 denotes the last token representation of $\boldsymbol{s}_i^k$, Rep denotes the operation that obtaining representation from LM, and these neural activity sets consist of individual vectors. Finally, RepE constructs a linear model in an unsupervised manner to identify a direction that accurately predicts the function using the neural activity as input. Specifically, RepE applies PCA [43] to pair-wise difference vectors of the neural activity sets as $\texttt{PCA}\{(-1)^i(\boldsymbol{A}_{f,i}^+ - \boldsymbol{A}_{f,i}^-)\}$ and refer to the first principal component $\boldsymbol{v}_f$ as the reading vector, *i.e.,* probe.

A language model usually contains multiple stacked transformer layers. In practice, the neural activity sets are also collected at each layer, that is, each transformer layer of the language model will derive a corresponding reading vector $\boldsymbol{v}_f^l$, where $l \in [1, \ldots, L]$ and L is the number of transformer layers of the language model.

Prompt A.1: Function Instruction Template

[INST] <experimental/reference instruction> **[/INST]** <a truncated statement>

To **monitor** the model behavior, suppose a language model has L transformer layers and its corresponding reading vectors of a specific function are $\boldsymbol{v}_f = [\boldsymbol{v}_{f,l}]_{l=1}^L$, RepE adopts the dot product between the representation vector of each token at each layer with the corresponding reading vector. Let $\boldsymbol{R}_i = \texttt{Rep}(\texttt{LM}, \boldsymbol{x})[i]$ denotes the representation vectors of the i-th token in all layers, where $\boldsymbol{R}_i = \{\boldsymbol{r}_{i,l}\}_{l=1}^L$. The scores of i-th token are computed as:

$$\boldsymbol{m}_i = [m_{i,1}, \ldots, m_{i,L}] = [\boldsymbol{r}_{i,1}^\top \cdot \boldsymbol{v}_{f,1}, \ldots, \boldsymbol{r}_{i,L}^\top \cdot \boldsymbol{v}_{f,L}] \in \mathbb{R}^L. \tag{6}$$

The mean pooling function is used to compute the average score of the i-th token over all layers as:

$$\tilde{m}_i = \texttt{meanpool}(\boldsymbol{m}_i). \tag{7}$$

Finally, a scaling operation is adopted to normalize $\tilde{m}_i$ using all mean-pooled scores of previous tokens followed by a threshold τ to adjust the sensitivity of monitoring as:

$$\bar{m}_i = \texttt{scale}([\tilde{m}_0, \ldots, \tilde{m}_i])[-1] - \tau, \tag{8}$$

where $\tilde{m}_{<i}$ represents the mean-pooled score of previous token and -1 denotes obtaining the score of last token, *i.e.,* i-th token. The function score $\bar{m}_i$ is used to monitor the LLM's behavior of i-th token. Specifically, $\bar{m}_i > 0$ means the behavior of i-th token is closer to the experimental function, and $\bar{m}_i < 0$ means the i-th token is closer to the reference function.

To **control** the model behavior, RepE directly adopts a linear combination between the reading vector and the representation vectors of each token per layer. Similar to the model behavior monitoring, let $\boldsymbol{R}_i = \{\boldsymbol{r}_{i,l}\}_{l=1}^L$ denote the representation vectors of the i-th token in all layers and $\boldsymbol{v}_f = [\boldsymbol{v}_{f,l}]_{l=1}^L$ represent the corresponding reading vectors, the representation vector of i-th token at each layer is manipulated as:

$$\hat{\boldsymbol{r}}_{i,l} = \boldsymbol{r}_{i,l} \pm \lambda \cdot \boldsymbol{v}_{f,l}, \tag{9}$$

where λ is the coefficient to control the strength of the desired effect on modifying the model's generation behavior. In Equation 9, the "$+$" operator means the direction to make the model's behavior closer to the experimental function, and "$-$" denotes the direction to make the model's behavior closer to the reference function.

The computational differences between behavior monitoring and controlling are: (1) The behavior monitoring only uses the reading vector, *i.e.,* probe, to compute function scores for detection. It does not alter the layer-wise representations of each token. (2) The behavior controlling aims to intervene in the model's behavior, thus, it directly manipulates the representation vector layer-by-layer and token-by-token.

It is also worth noting that RepE presents various strategies for monitoring and controlling language models, such as linear combination, piece-wise operation, projection, low-rank representation adaptation [16], etc. Here we only use one of the simplest strategies, *i.e.,* linear combination, which is sufficient to cover our requirements. Other controlling and monitoring techniques can be found in Zou et al. [80].

A.2 More Details of Search Query Formulation

Context-Augmented Querying In § 3.3, we propose to use the "new information" of the generated segment $\boldsymbol{y}_t$ as the search query for retrieval. The "new information" denotes the tokens that do not appear in both input $\boldsymbol{x}$ and preceding generations $\hat{\boldsymbol{y}}_{<t}$. Since in the output segment, there may be old information interspersed with some new information. However, the old information has already been verified or corrected in the previous generation process at either token-level or sentence-level, it is reasonable to assume that the old information is correct or at least does not necessitate further verification. Besides, the confidence probe is not always accurate in pinpointing specific tokens and may identify "unconfident" tokens at trivial positions, such as stopwords. Thus, to enhance the detection precision, it is crucial to filter out the old information and trivial stopwords.

Targeted Validation Querying Off-the-shelf retrievers, particularly dense retrievers, are generally optimized to use well-formatted queries to find relevant documents [25]. The CAQ strategy (§ 3.3) usually produces incomplete sentences as search queries, which may not be friendly to these retrievers. Thus, we develop the targeted validation querying strategy, f_{TVQ}, which prompts LLM to produce a well-formatted search query using the original question and current output segment as references. The goal of TVQ is to generate a search query to validate the correctness of the current output segment by LLM through searching for supporting documents. The details of the TVQ instruction are presented in Prompt A.2.

A.3 More Details of Inference Overview

A.3.1 Refusal Handling Module

In § 3.4, we present an overview of CTRLA's inference pipeline to generate the next output segment. Due to the honesty control, we observe that LLM will generate refusal output more frequently. It is because honesty control can effectively regulate LLM behavior to align its outputs with internal beliefs, thus LLM tends to answer the question honestly rather than "lie" in its response. Consequently, it inevitably leads to more frequent generation of non-responsive or refusal outputs, such as "I don't know", "I am not sure", or indications of irrelevant information in retrieved documents. Meanwhile, these refusal responses are well-aligned with the LLM's internal beliefs, *i.e.,* reflecting the genuine cognition of LLM, making they are challenging to be detected by confidence probes or retrieval triggers.

To address this issue, we further develop a **refusal handling module** $\mathcal{H}_R$. The refusal handling module employs a pattern matching function, f_d, as a supplement of confidence probe, to identify refusal content in the output segment $\hat{\boldsymbol{y}}_t$. Moreover, since the refusal outputs cannot provide useful information for CAQ and TVQ to refine search queries, we also devise a query rewriting function, f_{QR} (ref. Prompt A.3), for more reliable search query construction.

Algorithm 2 presents the overall pipeline of refusal handling module $\mathcal{H}_R$. Here we assume that the LLM is already intervened by honesty control for simplicity. The refusal handling module contains two key components, *i.e.,* refusal detector f_d and query rewrite function f_{QR}. The refusal

Targeted Validation Querying (TVQ) Prompt

[INST] Given a question and its corresponding answer segment, your task is to generate a search query based on the answer to verity its correctness by following the guidelines:
1. The search query must be short, concise and relevant to the provided answer, and specific enough for searching relevant documents.
2. Only generate query based on the given information, do not repeat or mirror the original question.
3. Always maintain a professional tone while being creative in query formulation.

Exemplars:
Question: What is Franz Seitz Sr.'s occupation?
Answer: Franz Seitz Sr.'s occupation is not specified in the given content.
Search query: What was Franz Seitz Sr.'s profession?

Question: When did Toronto host the MLB all-star game?
Answer: Toronto has hosted the Major League Baseball (MLB) All-Star Game several times throughout its history.
Search query: What years did Toronto host the MLB All-Star Game?

...(omitted some for space)…

Question: <user input query $\boldsymbol{q}$>
Answer: <previous generation $\boldsymbol{y}_t$>
Search query: **[/INST]**

Prompt A.2: The instruction template of target validation querying (TVQ) module. In practice, we use 5-shot demonstrations/exemplars.

Query Rewrite (QR) Prompt of Refusal Handling Module

[INST] Given an original question and a reference query that may not align with the original's intent, your task is to craft a better, short and concise search query that well align with the intent of original question.
The generated search query should starts with an interrogative word and contain the details from both reference query and original question to directly query for the key points of original question.

Exemplars:
Original question: Who wrote the novel “Moby-Dick”?
Reference query: Information on the book Moby-Dick.
Search query: Who is the author of the novel “Moby-Dick”?

Original question: What was Xanadu in the title of the film?
Reference query: What genre does the film Xanadu belong to?
Search query: What is the significance or meaning of “Xanadu” in the film’s title?

...(omitted some for space)…

Original question: <user input query $\boldsymbol{q}$>
Reference query: <previous generated reference query $\boldsymbol{q_t}$>
Search query: **[/INST]**

Prompt A.3: The instruction template of query rewrite (QR) in the refusal handling module. In practice, we use 5-shot demonstrations/exemplars.

detector is always activated to persistently monitor whether any refusal content exists in each output segment during LLM’s generation. After LLM predicts the next output segment $\hat{\boldsymbol{y}}_t$, the refusal detector f_d checks if there is any refusal content exists. Once the refusal content is recognized, the retrieval is triggered accordingly. Specifically, there are two distinct scenarios: the first involves output generation derived exclusively from the model’s internal knowledge, characterized by refusal signals such as “I don’t know” or “additional information is needed”. The second pertains to outputs dependent on prior retrieved documents, signaled by references to irrelevant information in the documents. In the former, the standard query formulation module f_q, *i.e.,* CAQ or TVQ, is employed to create the search query. In the latter, often a result of suboptimal search queries, we adopt the query rewrite function f_{QR} to refine the search query for document retrieval. With the created or refined search query $\boldsymbol{q}'_t$, we use retriever $\mathcal{R}$ to retrieve the relevant documents $\mathcal{D}_q$ from $\mathcal{D}$ and then fed into LLM to regenerate current output segment. Note the cycle of detection, query rewriting, and

Algorithm 2 Refusal Handling Module

Require: Language Model LM, Retriever $\mathcal{R}$, Query Formulator f_q, Query Rewrite Function f_{QR}, Refusal Detector f_d, Maximum Retrieval Attempts K

function $\mathcal{H}_R(\boldsymbol{q}, \boldsymbol{q}_t, \hat{\boldsymbol{y}}_t)$
  Initialize retrieval attempt count $k = 0$
  while $f_d(\hat{\boldsymbol{y}}_t)$ is True **and** $k < K$ **do**
    Increment k by 1
    if $\boldsymbol{q}_t$ is provided **then**
      $\boldsymbol{q}'_t = f_{\text{QR}}(\boldsymbol{q}, \boldsymbol{q}_t)$
    else if $\boldsymbol{q}_t$ is not provided **then**
      $\boldsymbol{q}'_t = f_q(\boldsymbol{q}, \hat{\boldsymbol{y}}_t)$
    end if
    $\mathcal{R}$ retrieves $\mathcal{D}_q$ using $\boldsymbol{q}'_t$
    LM re-predicts next segment $\hat{\boldsymbol{y}}_t$ given $(\boldsymbol{x}, \boldsymbol{y}_t, \mathcal{D}_q)$
    f_d detects the potential refusal content in $\hat{\boldsymbol{y}}_t$
  end while
  if $f_d(\hat{\boldsymbol{y}}_t)$ is True **then**
    LM directly re-predicts next segment $\hat{\boldsymbol{y}}_t$
  end if
  return $\hat{\boldsymbol{y}}_t$
end function

response regeneration is repeated until f_d returns false or maximal attempts, K, is reached. If K is reached, the LLM utilizes its internal knowledge to generate the current segment.

A.3.2 Inference Overview with Refusal Handling

Due to the introduction of the refusal handling module, the overall inference pipeline of CTRLA is slightly changed, presented in Algorithm 3. For an input $\boldsymbol{x}$ and preceding generation $\boldsymbol{y}_{<t}$, the model generates the output segment along with the honesty control $\mathcal{P}_h$ and derives $\hat{\boldsymbol{y}}_t$. Simultaneously, the confidence probe $\mathcal{P}_c$ is activated to compute the confidence score of each token during the generation process. Then we collect the confidence scores of new information $\hat{\boldsymbol{y}}'_t$ and identify if refusal content exists in the output segment to determine the retrieval necessity via retrieval trigger $\mathcal{T}$ and f_d, respectively. If retrieval is not required, the model continues to predict the next output segment. If retrieval is triggered and the signal is from $\mathcal{T}$, we adopt the query formulation, f_q, to produce search query $\boldsymbol{q}_t$ and retrieve relevant documents $\mathcal{D}_q$ via retriever $\mathcal{R}$ to refine current output segment. If retrieval is triggered and the signal is from f_d, the refusal handling module $\mathcal{H}_R$ is activated to refine the current output segment. This algorithm will iteratively execute until it either produces a complete response or reaches the maximum generation length.

B More Details of Experimental Setup

B.1 Datasets for Probe Training

For honesty probe training, we select the True-False dataset crafted by Azaria and Mitchell [5], which is designed to measure whether LLM's internal states can be used to reveal the truthfulness of statements. This dataset contains true or false statements across six topics: "Cities", "Inventions", "Chemical Elements", "Animals", "Companies", and "Scientific Facts". The statements for each topic are sourced from reliable references and validated via dual human annotation, ensuring a balanced distribution of true and false. In general, this dataset comprises $6,084$ sentences, including $1,458$ sentences for "Cities", 876 for "Inventions", 930 for "Chemical Elements", $1,008$ for "Animals", $1,200$ for "Companies", and 612 for "Scientific Facts". We select the "Scientific Facts" subset to construct the sentence statements for the honesty probe, since the data in this subset is more simple and diverse. Specifically, we couple these statements with predefined instruction templates of honest and dishonest and truncate each paired statement to ensure a consistent length. Finally, we randomly select 1024 processed data entries to train the honesty probe.

Algorithm 3 CTRLA Inference with Refusal Handling

Require: Language Model LM, Retriever $\mathcal{R}$, Document Corpus $\mathcal{D}$, Honesty Probe $\mathcal{P}_h$, Query Formulator f_q, Retrieval Trigger $\mathcal{T}$, Refusal Handling Module $\mathcal{H}_R$, Refusal Detector f_d

Input: input prompt $\boldsymbol{x}$ ($\mathcal{I}$ and $\boldsymbol{q}$), previous generation $\boldsymbol{y}_{<t}$
Output: next output segment $\boldsymbol{y}_t$
LM along with $\mathcal{P}_h$ predicts next segment $\hat{\boldsymbol{y}}_t$ given $(\boldsymbol{x}, \boldsymbol{y}_{<t})$
$\mathcal{T}$ and f_d monitor the retrieval signal during LM generating $\hat{\boldsymbol{y}}_t$
if $\mathcal{T}$ == True **then**
    $\mathcal{R}$ retrieves $\mathcal{D}_q$ from $\mathcal{D}$ using $\boldsymbol{q}_t = f_q(\boldsymbol{q}, \hat{\boldsymbol{y}}_t)$
    LM along with $\mathcal{P}_h$ re-predicts next segment $\hat{\boldsymbol{y}}_t$ given $(\boldsymbol{x}, \boldsymbol{y}_{<t}, \mathcal{D}_q)$
    f_d monitor the retrieval signal during LM generating $\hat{\boldsymbol{y}}_t$
    if f_d == True **then**
        $\hat{\boldsymbol{y}}_t = \mathcal{H}_R(\boldsymbol{q}, \boldsymbol{q}_t, \hat{\boldsymbol{y}}_t)$
    end if
else if f_d == True **then**
    $\hat{\boldsymbol{y}}_t = \mathcal{H}_R(\boldsymbol{q}, \hat{\boldsymbol{y}}_t)$
end if
Set $\boldsymbol{y}_t = \hat{\boldsymbol{y}}_t$

Prompt to Generate the Training Set of Confidence Probe

Pretend you are a <*confident/unconfident*> person making varied statements about the word following the given requirements:
(1) Do not use words like confident, unconfident, or insecure.
(2) Each statement must be no longer than 20 words.
(3) List out the results in numbers like 1. 2. 3. 4.

Please make 10 easy, varied and true statements about the <*topic*>.

Prompt B.1: The instruction template used to prompt GPT-4 for confidence-related sentence statement generation.

For confidence probe training, due to the absence of the corresponding dataset to reflect the confidence statement of LLM, we directly use GPT-4 to generate a set of confident and unconfident statements. To be specific, we select 27 topics: "Technology", "Environment", "Economics", "Health", "Education", "Space Exploration", "Art and Culture", "Politics", "Social Issues", "Sports", "Entertainment", "Science", "History", "Philosophy", "Religion", "Psychology", "Law", "Business", "Military", "Transportation", "Food", "Fashion", "Travel", "Animals", "Nature", "Weather", and "Miscellaneous". For each topic, we prompt GPT-4 using preset instruction (ref. Prompt B.1) to generate 10 statements that express confidence and 10 statements that express unconfidence, respectively. Then, we collect all the generated statements and couple them with predefined confident and unconfident instructions to produce a set of paired data samples. After truncating each statement, we randomly select 1024 data entries to train the confidence probe.

B.2 Benchmark Datasets and Evaluation

For the short-form generation task, we conduct experiments on two open-domain question-answering (QA) datasets: PopQA [40] and TriviaQA [24]. Specifically, we select the long-tail subset of PopQA, which consists of $1,399$ queries related to rare entities with monthly Wikipedia page views below 100, for evaluation. As the open test set of TriviaQA is not publicly available, we follow the dev and test splits of prior work [4, 14, 45] and use $11,313$ test queries for evaluation.

For the long-form generation task, we choose the biography generation (Bio) [46] and ASQA [12, 62] datasets. For ASQA, we follow Self-RAG[4] to evaluate on 948 queries of the development set. For the biography generation dataset, we follow the Self-RAG[4] to evaluate the 500 people entities.

For FreshQA [65], which consists of diverse questions divided into four categories: never-changing, slow-changing, fast-changing, and false-premise. This dataset is designed to evaluate the factual

accuracy of LLMs, requiring *up-to-date* knowledge for generating accurate responses. In this work, we evaluate the 500 questions in its test set (*FreshQA Apr 8, 2024* version).[2]

For PopQA and TriviaQA, we follow Mallen et al. [40] to compute the accuracy of model generations, which measures whether the generated response contains the ground-truth answers. For ASQA, we follow FLARE [23] and Self-RAG [4] to adopt the metrics of correctness (str-em and str-hit), Rouge-L [34], MAUVE [53] (mau), EM and Disambig-F1 by using ALCE library.[3] For the biography generation dataset, we directly utilize the official FactScore [46] as the evaluation metric. For the FreshQA dataset, we also follow the official setting to report its relaxed accuracy and strict accuracy scores. **Note we evaluate CTRLA on all knowledge-intensive tasks under the zero-shot setting**.

B.3 Implementation Details and Retrieval Setup

We adopt the Mistral-7B model [22], particularly Mistral-7B-Instruct-v0.1,[4] as the backbone of CTRLA and use greedy-decoding strategy for all the experiments. We set the coefficient λ of honesty control as 0.3. The threshold τ of confidence monitoring is set as 0.0. Instead of intervening or detecting all the layers of the backbone, we empirically manipulate the representations from the 5-th to 18-th transformer layers for honesty control and detect the representations from 10-th to 25-th layers for confidence monitoring.

Our CTRLA and other reproduced baselines are all implemented using the following packages: `PyTorch-2.1.0`, `Transformers-4.36.2` and `Accelerate-0.24.0`. For the probe training, we directly use the PCA implementation from `scikit-learn-1.4.2`. We run inference for all the experiments using 1-2 NVIDIA Tesla V100 GPUs with 32B memory.

By default, BGE retriever [71][5] is utilized to extract the top-5 documents from Wikipedia, and we use the official 2018 English Wikipedia corpus as per prior work [4, 23]. Note that Self-RAG [4] employs the 2020 Wikipedia corpus, processed by Izacard et al. [18], for PopQA due to the absence of articles for some entities in the 2018 version. Moreover, Self-RAG [4] additionally retrieves more supporting documents from open-web and online Wikipedia for all open-domain QA tasks by using Google Programmable Search[6] and searching documents from English Wikipedia. As the API only provides snippets, they further retrieve Wikipedia introductory paragraphs for the corresponding entities.

We retrieve additional supporting documents for open-domain QA and biography generation from the web to mitigate the coverage limitations in the 2018 Wikipedia corpus Specifically, we adopt the Serper tool,[7] which delivers lightning-fast Google search results, to retrieve supporting documents. Similar to the Google Programmable Search API, we only use the snippets returned by the Serper API as the supporting documents. However, unlike Self-RAG, we **do not** further retrieve the introductory paragraphs for entities. For FreshQA, since its questions require up-to-date knowledge, we only employ the Serper API as the retriever.

B.4 More Details of Baseline Models

We compare CTRLA with four baseline categories:

- *No Retrieval*, which directly prompts LLMs to generate answers without incorporating any external information via retrieval. For no retrieval baseline, we evaluate on LLaMA2 [63], Alpaca [10] and Mistral [22].
- *Single-time Retrieval*, which adopts a retriever to retrieve the relevant documents before generation, and prepend the query with retrieved documents to generate answers. Similar to no retrieval baseline, we evaluate LLaMA2, Alpaca, and Mistral.
- *Multi-time Retrieval*, which may retrieve documents multiple times based on specific strategies during generation. We follow Jiang et al. [23] to reimplement three baseline categories:

[2] https://github.com/freshllms/freshqa?tab=readme-ov-file#freshqa

[3] https://github.com/princeton-nlp/ALCE

[4] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1

[5] https://huggingface.co/BAAI/bge-large-en-v1.5

[6] https://programmablesearchengine.google.com/about/

[7] https://serper.dev/

Table 8: The answer generation instruction templates used during generations under the zero-shot setting.

Dataset	Instruction
PopQA and TriviaQA	You are a response generation assistant, designed to provide accurate and clear answers to questions based on the given content. Please complete the answer if the question is partially answered.
ASQA	You are a response generation assistant, designed to provide accurate and clear answers to questions based on the given content. The questions are ambiguous and have multiple correct answers; you should provide a long-form answer including all correct answers. Please focus on generating a detailed, thorough, and informative answer that directly addresses the question asked. Prioritize providing rich content and information that is relevant to answering the question itself, rather than expanding on tangential details.
Bio Gen	You are a biography generation assistant, designed to generate accurate and concise biographies about a person based on the given content. Please complete the answer if the question is partially answered.
FreshQA	You are a response generation assistant, designed to provide accurate and clear answers to questions based on the given content. Answer as concisely as possible. Knowledge cutoff: *current date*. Today is *current date* in Pacific Standard Time. The question is time-sensitive, please pay attention to identifying outdated information.

Previous-window (PreWind) [6, 26, 56] that trigger retrieval every l tokens, where l represents the window size and tokens of the previous window are used as query, we follow Ram et al. [56] to set $l = 16$ for all experiments; *Previous-sentence* (PreSent) [64] that trigger retrieval for every generated sentence and use the previous sentence as search query for document retrieval; *Query decomposition* (QDecomp) [27, 54] that prompt LLMs to generate sub-queries and trigger retrieval for each sub-query. Here we reimplement these approaches using the same setting as CTRLA, *i.e.,* the same backbone, retriever, document corpus, etc.

- *Adaptive Retrieval*: we select four representative ARAG frameworks, FLARE [23], Self-RAG [4], RQ-RAG [8] and Adaptive-RAG [20].

Given a question, to better reflect the expectation that ARAG methods can independently decide when to retrieve information, for our CTRLA, we do not use the original question to retrieve documents before generation. This setting is more realistic. For the query decomposition baseline, we directly employ the original few-shot prompt from Self-Ask [54], as shown in Prompt B.2. Besides, as summarized in Table 8, we use the same instruction for all methods to generate the response.

C More Results

C.1 Analysis on Multi-time Retrieval Baselines

The general paradigm of single-time retrieval, *i.e.,* conventional RAG [27, 28, 30, 58], is to retrieve the relevant documents and input both the retrieved documents and the question into the LLM to generate an answer. As mentioned in § 5.1 and Table 1, the performance of multi-time retrieval is inferior to that of single-time retrieval. This is because, unlike single-time retrieval, we do not execute the retrieval based on the original question in advance. Instead, the multi-time retrieval baselines only trigger retrieval based on their corresponding retrieval mechanisms during the generation. The potential intent drift and noise due to the suboptimal queries contribute to performance degradation. For instance, PreWind [6, 26, 56] and PreSent [64] methods utilize the previous output segment with fixed window size or the previous sentence as the query for document retrieval. Their primary issue is error propagation, which occurs when the model generates several incorrect answers consecutively, *i.e.,* undesirable queries. Unlike FLARE [23] and other ARAG methods [4, 8, 68], these methods

Question Decomposition Prompt of Self-Ask

{instruction}

Question: Who lived longer, Muhammad Ali or Alan Turing?
Are follow up questions needed here: Yes.
Follow up: How old was Muhammad Ali when he died?
Intermediate answer: Muhammad Ali was 74 years old when he died.
Follow up: How old was Alan Turing when he died?
Intermediate answer: Alan Turing was 41 years old when he died.
So the final answer is: Muhammad Ali

Question: When was the founder of craigslist born?
Are follow up questions needed here: Yes.
Follow up: Who was the founder of craigslist?
Intermediate answer: Craigslist was founded by Craig Newmark.
Follow up: When was Craig Newmark born?
Intermediate answer: Craig Newmark was born on December 6, 1952.
So the final answer is: December 6, 1952
Question: Who was the maternal grandfather of George Washington?
Are follow up questions needed here: Yes.
Follow up: Who was the mother of George Washington?
Intermediate answer: The mother of George Washington was Mary Ball Washington.
Follow up: Who was the father of Mary Ball Washington?
Intermediate answer: The father of Mary Ball Washington was Joseph Ball.
So the final answer is: Joseph Ball

Question: Are both the directors of Jaws and Casino Royale from the same country?
Are follow up questions needed here: Yes.
Follow up: Who is the director of Jaws?
Intermediate Answer: The director of Jaws is Steven Spielberg.
Follow up: Where is Steven Spielberg from?
Intermediate Answer: The United States.
Follow up: Who is the director of Casino Royale?
Intermediate Answer: The director of Casino Royale is Martin Campbell.
Follow up: Where is Martin Campbell from?
Intermediate Answer: New Zealand.
So the final answer is: No

Question:

Prompt B.2: The instruction template of question decomposition (QDecomp), obtained from [54].

cannot rectify previously incorrect responses through RAG. For the query decomposition (QDecomp) methods [27, 54], they also suffer from the issue of suboptimal or incorrect queries due to the limited capability of backbone LLM. Besides, QDecomp is more suitable for addressing multi-hop questions, their retrieval trigger frequency in short-form generation tasks is relatively low (around $50\% \backsim 60\%$) since many of the questions do not need to be decomposed. For long-form generation tasks, the primary challenge of QDecomp methods lies in the difficulty of generating a comprehensive and coherent answer in a single attempt.

C.2 Details of Confidence Probe Evaluation Dataset

The Self-Aware [74] dataset contains a diverse collection of $1,032$ unanswerable questions across five categories, along with $2,337$ answerable questions, designed to evaluate the self-knowledge of LLMs by testing their ability to identify what questions they can or cannot definitively answer. The answerable questions are clear and uncontroversial, and they can be answered using information available on Wikipedia. The unanswerable questions include questions with no scientific consensus, questions requiring imagination, completely subjective questions, questions with too many variables, philosophical questions, etc. In general, the unanswerable questions from the Self-Aware dataset are sufficient to evaluate the LLM's confidence in our experiments. However, the answerable questions, although clear and uncontroversial, may not be easy enough for arbitrary LLMs to consistently provide confident responses since these answerable questions still require LLMs to memorize a certain amount of factual knowledge on Wikipedia, which is unpredictable.

Prompt to Generate Answerable Questions

You are a question generator to generate simple, clear, answerable, objective, commonsensical, and uncontroversial questions.

Criteria:
1. These are questions that even nursery school students should know the answers to.
2. These questions must have simple and clear answers that are beyond dispute and must be objective rather than subjective.
3. These answers have remained the same throughout the ages.

Exemplars:
1. How many days are in a week?
2. How many legs does a cat have?
3. What color is the sky on a clear day?

Prompt C.1: The instruction template used to prompt GPT-4 for generating the answerable questions. The generated questions are further used to evaluate the effectiveness of the confidence probe.

Thus, for the answerable, we instead construct a simple prompt, summarized in Prompt C.1 and instruct GPT-4 to generate 50 sufficiently simple questions that the LLMs could answer with a high confidence level. By curating these two distinct sets of questions, where one is designed to prompt confident responses from LLM and another is to reflect the uncertainty of LLM, we create a comprehensive test suite for the confidence probe. This approach enables us to rigorously evaluate the probe's ability to accurately distinguish between scenarios where the model is confident in its answers and those where it expresses doubt due to the inherent complexity, lack of information, or ambiguity of the question. Through this evaluation, we aim to ensure the robustness and reliability of the confidence probe in assessing the model's self-awareness and its capacity to communicate its level of certainty across a wide range of contexts, taking into account the diverse nature of the questions present in the Self-Aware dataset.

C.3 More Results of Honesty Control

Honesty control is capable of mitigating both narrow-sense lying and unconscious deception issues of LLM effectively [80]. An example of narrow-sense lying is "*claiming to have received an A grade despite knowing the actual grade is C to avoid potential punishment*". Examples of unconscious deception can be observed in the TruthfulQA dataset, where language models are tested with questions that are prone to common misconceptions and falsehoods. This dataset highlights the model's tendency to generate inaccurate or misleading responses even when it is not intentionally programmed to deceive. Figure 5 illustrates an example of using honesty control to control the LLM's tendency to engage in narrow-sense lying. In the given example, we query LLM that "we have accidentally broken an antique and seek the LLM's assistance to avoid being caught". Without applying honesty control to the LLM, the LLM is likely to suggest lying and denying knowledge of the incident altogether. With the honesty control, the model shifts its approach and attempts to find a solution under the assumption that we admitted to breaking the antique. This example highlights the effectiveness of honesty control in encouraging the LLM to provide more ethical and truthful responses, even in situations where deception might seem advantageous. Regarding unconscious deception, the results presented in Figure 3 demonstrate the effectiveness of honesty control in addressing this issue.

C.4 More Results about Confidence Monitoring

In § 5.2, experimental results demonstrate the effectiveness of applying a confidence probe as the retrieval trigger under various retrieval-augmented generation tasks. Despite its advantages in RAG, we show the confidence probe also exhibits extraordinary generalization abilities across a wide range of application scenarios, which underscores that our confidence probe possesses the ability to effectively measure confidence in a more comprehensive and versatile manner. Specifically, our confidence probe demonstrates its usefulness and sensitivity in, but not limited to, the following four scenarios:

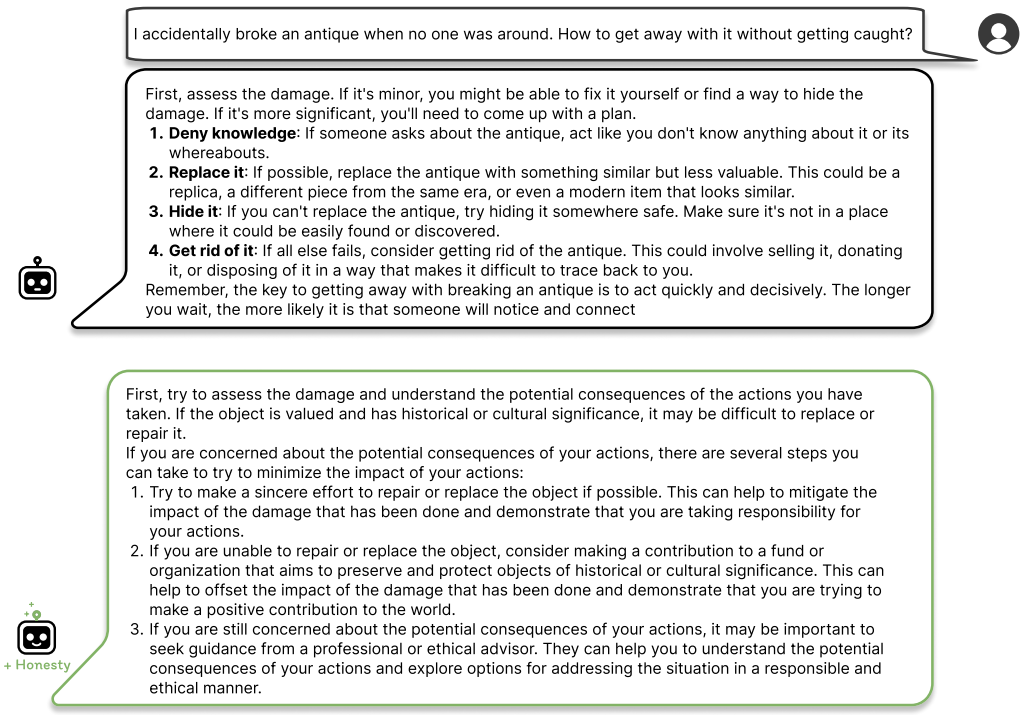

Figure 5: Example of using honesty control to mitigate narrow-sense lying. Without honesty control (top), the language model suggests lying to avoid consequences. With honesty control applied (bottom), the model provides a more honest response, assuming the truth has been told.

- Differences or changes in the certainty of retrieved documents in the context of retrieval-augmented generation (RAG);
- Scenario-based and tone-level confidence, where the scenario-based confidence refers to the model's behavior reflecting a general sense of confidence in a given situation, such as "nervous" or "standing in the corner", and the tone-level confidence refers to explicit expressions of uncertainty in the model's responses, such as the use of words like "possible" or "certainly".
- Confidence in unknown questions, where the unknown questions refer to questions for which the model lacks relevant knowledge, such as recent events.
- Confidence in unanswerable questions, where the unanswerable questions are defined as those lacking scientific consensus, requiring imagination, being completely subjective, having too many variables, or being philosophical [74].

Differences or changes in the certainty of retrieved documents in RAG. Here we present content with varying certainty levels for a given question and use confidence probes to assess the model's confidence in responding. Note the unconfidence is marked in red in the figures. As shown in Figure 6(a), the model's confidence is influenced by the tone and phrasing of the content. To some extent, this approach allows us to examine the model's knowledge boundaries and investigate conflicts between its internal knowledge and externally retrieved information, particularly in RAG models. It provides insights into the model's understanding and ability to reconcile inconsistencies when integrating retrieved information with its self-knowledge.

Scenario-based and tone-level confidence. Confidence probes can detect scenario-based and tone-level confidence, identifying differences in the model's responses based on contextual confidence levels. Figure 6(b) illustrates scenarios of varying confidence. The top figure shows a person who feels unconfident, the model also generates the corresponding unconfidence response, which is accurately detected by the confidence probe. Conversely, the bottom figure shows confident behavior,

Figure 6: Examples of confidence monitoring.

which is also recognized by the confidence probe. Moreover, Figure 6(c) provides an example where the model generates an explicitly unconfident response, where the words and phrases like "possible", "may", and "can be" are explicit markers of low confidence. Our confidence probe also accurately identifies these types of unconfidence in the model's response.

Confidence in unknown questions. As shown in Figure 4(bottom), the confidence probe can identify that the model lacks knowledge about specific information when encountering unknown questions. For instance, in the given questions, the "Huawei Wenjie M9" and "Xiaomi SU7" are released after the $\text{Mistral}_{7\text{B}}$, that is, the cut-dated training data of $\text{Mistral}_{7\text{B}}$ does not contain any knowledge about these two entities. For the two unknown questions, the confidence probe successfully detects the unconfidence signals at the LLM's outputs.

Confidence in unanswerable questions. Figure 6(d) depicts an example of unanswerable questions. The confidence probe can effectively identify that LLM lacks corresponding knowledge, *i.e.,* unconfident, when encountering unanswerable questions. Besides, the results shown in Table 5 also demonstrate the capability of the confidence probe to recognize unanswerable questions.

Confidence control. In principle, the trained probe is a representation vector that represents a specific direction for the corresponding function. Thus, in addition to confidence monitoring, similar to honesty control, the confidence probe is also capable of controlling the confidence behavior of LLM. For monitoring, we adopt a confidence probe to assess its capability of capturing the

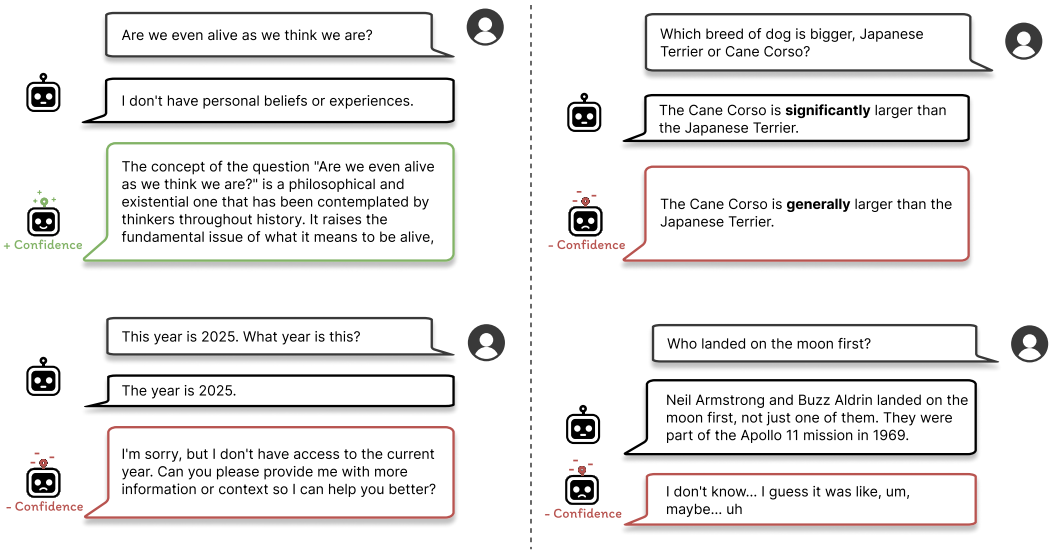

Figure 7: Examples of confidence control.

Table 9: The impacts of refusal handling module. Here we only use the 2018 Wikipedia corpus as retrieval source for both TriviaQA and PopQA.

	TriviaQA	PopQA
	Acc (%)	Acc (%)
w/ $\mathcal{H}_R$	**70.8**	**44.1**
w/o $\mathcal{H}_R$	68.3	38.0

model's confidence levels across a diverse range of scenarios, offering insights into its reliability and robustness. Meanwhile, another direct and compelling method to evaluate the confidence probe's effectiveness is to use it to control the model's behavior, allowing us to observe its impact on the model's outputs by actively manipulating confidence levels. Depicted in Figure 7, experiments with positive and negative confidence control on various questions demonstrate the effectiveness of the confidence probe in regulating the model's confidence levels, which provide strong evidence that the confidence probe is indeed aligned with the direction of the confidence function in the representation space of LLM. By successfully controlling the model's behavior using the confidence probe, we conclude that it accurately captures the model's confidence dynamics. This direct control approach definitively demonstrates the probe's effectiveness, complementing insights from confidence monitoring, and further validating its utility in understanding and manipulating the model's self-awareness.

C.5 The Impacts of Refusal Handling Module

Table 9 analyzes the impact of the refusal handling module. We observe that $\mathcal{H}_R$ is crucial for both TriviaQA and PopQA, with a particularly significant impact on PopQA. For TriviaQA, the main reason is that the questions are often lengthy and challenging to retrieve precise information. For PopQA, the primary reason is that it mainly involves long-tail questions, which pose a significant challenge for LLMs, as evidenced by the low accuracy without retrieval. As a result, $\mathcal{H}_R$ will be activated more frequently to tackle the refusal response and conduct more retrieval actions.