TC–RAG: Turing–Complete RAG's Case study on Medical LLM Systems

Xinke Jiang[1,2*], Yue Fang[1,2*], Rihong Qiu[1,2*], Haoyu Zhang[7], Yongxin Xu[1,2]
Hao Chen[8], Wentao Zhang[9], Ruizhe Zhang[1,2], Yuchen Fang[10], Xinyu Ma[1,2]
Xu Chu[1,2,4,6], Junfeng Zhao[1,2,5†], Yasha Wang[2,3,6†]

[1]School of Computer Science, Peking University, Beijing, China
[2]Key Lab of HCST (PKU), MOE; SCS, Peking University, China
[3]National Engineering Research Center For Software Engineering, Peking University, China
[4]Center on Frontiers of Computing Studies, Peking University, Beijing, China
[5]Big Data Technology Research Center, Nanhu Laboratory, Jiaxing, China
[6]Peking University Information Technology Institute, Tianjin Binhai, China
[7]City University of Hong Kong (Dongguan) [8]University of Chinese Academy of Sciences
[9]ShanghaiTech University [10]University of Electronic Science and Technology of China

{xinkejiang, fangyue, rihongqiu}@stu.pku.edu.cn, {zhaojf, wangyasha}@pku.edu.cn

Abstract

In the pursuit of enhancing domain-specific Large Language Models (LLMs), Retrieval-Augmented Generation (RAG) emerges as a promising solution to mitigate issues such as hallucinations, outdated knowledge, and limited expertise in highly specialized queries. However, existing approaches to RAG fall short by neglecting system state variables, which are crucial for ensuring adaptive control, retrieval halting, and system convergence. In this paper, we introduce the **T**uring– **C**omplete–**RAG** (TC–RAG) through rigorous proof, a novel framework that addresses these challenges by incorporating a Turing Complete System to manage state variables, thereby enabling more efficient and accurate knowledge retrieval. By leveraging a memory stack system with adaptive retrieval, reasoning, and planning capabilities, TC–RAG not only ensures the controlled halting of retrieval processes but also mitigates the accumulation of erroneous knowledge via **Push** and **Pop** actions. In the case study of the medical and general domain, our extensive experiments on seven real-world healthcare and general-domain datasets demonstrate the superiority of TC–RAG over existing methods in accuracy by over 7.20%. Our code, datasets and RAG resources have been available at https://github.com/Artessay/TC-RAG .

1 Introduction

Large Language Models (LLMs), such as ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023), have achieved remarkable strides in pivotal areas, demonstrated exceptional performance across a variety of downstream tasks (Kaplan et al., 2020; Vu et al., 2024; Ma et al., 2025, 2024). In the medical domain—those medical LLMs (Wang et al., 2023a; Zhang et al., 2023; Yang et al., 2023; Zhu et al., 2023; Pal and Sankarasubbu, 2023)—exhibit great promise in the healthcare field, where accountability and trustworthiness are paramount (Ji et al., 2023a; Song et al., 2024). By incorporating comprehensive medical knowledge through pre-training (Kaplan et al., 2020), they can support physicians in making accurate diagnoses and formulating treatment plans (Jiang et al., 2023a), as well as enhance medical resource allocation (Wang et al.; Xu et al.; Liu et al., 2024). Despite medical LLMs' advancements, significant conundrums still remain, including the difficulty in avoiding factual inaccuracies (*i.e.*, hallucinations) (Ji et al., 2023a; Cao et al., 2020; Ji et al., 2023b; Jiang et al., 2024b), outdated knowledge (He et al., 2022), and a lack of highly specialized expertise (Kandpal et al., 2023). Consequently, **R**etrieval-**A**ugmented **G**eneration (RAG) (Edge et al., 2024; Asai et al., 2024; Yang et al., 2024b; Xu et al., 2025b; Jiang et al., 2024a), which uses the external knowledge base to provide medical knowledge as contextual information to enhance content generation, combined with medical-LLMs with their massive parameterized knowledge can be likened to the expertise of doctors, is deemed as a promising and necessary solution to the aforementioned difficulties.

However, while current approaches in enhancing LLMs with external knowledge through RAG show promise (Su et al., 2024; Asai et al., 2024; Jiang et al., 2023c), they have consistently overlooked the introduction of system state variables—**an essential component for ensuring adaptive control, retrieval halting, and system convergence**. Moreover, these existing RAG methods are not Turing Complete, lacking the ability to dynamically manage and monitor the retrieval process in a way that guarantees convergence to a reliable conclusion. In complex medical scenarios, where decisions often require intricate, multi-step reasoning and adaptive responses (Mustafa et al., 2023), the absence of Turing Completeness (Turing, 1936) significantly limits a system's effectiveness and reliability. This gap moti-

[*]Xinke, Yue and Rihong are equal contribution.
[†]Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11400–11426
July 27 - August 1, 2025

vates our approach: **to construct a Turing Complete RAG System** that effectively manages state variables, using a finite logical framework to enhance the RAG process. However, how to effectively construct a Turing Complete RAG system remains unexplored and faces substantial challenges:

❶ ***C1. How to design a Turing Complete RAG System with the monitored state variable.*** Designing a Turing Complete RAG system requires the integration of monitored state variables that dynamically track and control the retrieval process—something that existing RAG methods lack. Current approaches (Jeong et al., 2024; Su et al., 2024) do not have an explicit mechanism to assess whether the system has converged to a reliable conclusion, which is a critical gap. A significant challenge lies in leveraging the forward propagation of the large model to accurately compute these state variables in real-time. This involves ensuring that the state variables effectively reflect the system's evolving context, guiding crucial decisions on whether to continue, halt, or refine the retrieval process. Managing these variables within the model's forward pass, while maintaining adaptability to complex and varied medical queries, is essential for achieving both efficiency and accuracy, ensuring that the retrieval process finally reliably converges to an optimal conclusion.

❷ ***C2. Whether to, What to, and How to plan retrieval for efficient and accurate knowledge to maintain optimal state.*** With the ability to assess the state, how to dynamically manage it to achieve the expected state is significant. For the real-life consultation case, doctors will decide whether to perform and what to search based on their state of mastery of this problem (Cox et al., 2021), instead of the regardless search–which can lead to redundant information that the model already possesses, potentially causing confusion or even misleading the LLMs. Moreover, an experienced doctor can systematically analyze and plan further steps with access to a vast medical knowledge base, while a layperson might struggle in a dilemma as to choose where to start or which tools to use (Yao et al., 2022a; Zhu et al., 2024). The medical LLM is analogous to a medical expert (DIS, 2023), and the challenge lies in effectively utilizing the LLM's internal parameterized knowledge to retrieve to maintain an optimal state.

❸ ***C3. How to avoid irrelevant noise affecting system state during RAG.*** Since the traditional RAG's retrieval process is typically driven by query keywords (Soman et al., 2023b,a; Sen et al., 2023; Kim et al., 2023) rather than the model's specific needs, it may introduce extensive irrelevant and noisy context. And the erroneous knowledge will continue to accumulate with the retrieval and reasoning process (Yao et al., 2022a; Shinn et al., 2024), which can cause to waste token resources (Jiang et al., 2024b), accumulate invalid memories, and encounter the "lost in the middle" (Liu et al., 2023) problems. Therefore, how to effectively remove erroneous knowledge is crucial for maintaining system state.

To address these challenges, we propose **T**uring **C**omplete-**RAG** (**TC–RAG**), a Turing Complete System for domain-specific LLMs to provide reliable and trustworthy medical analysis. ❶ For ***C1***, we designed a Turing Complete RAG system with a memory stack that monitors intermediate states, ensuring the retrieval process reliably converges to an optimal conclusion. ❷ For ***C2***, we extensively collected medical data and pretrained a medical LLM, elevating its understanding from layperson to expert level, thus enhancing its reasoning and planning abilities. The model's reasoning ability is leveraged to decide whether and what to retrieve adaptively, and its planning capacity guides tool usage and action planning, akin to how medical professionals solve complex problems. ❸ For ***C3***, TC–RAG incorporates a memory stack system with backtrack and summary operations to timely remove errors and condense redundant knowledge, mitigating accumulation of erroneous information and noise. In summary, our contributions are as follows:

- To the best of our knowledge, TC–RAG is the first RAG framework to introduce the system state variable and the Turing Completeness mechanism, which could make the retrieval process controllable and halt.
- By introducing state variable, we theoretically prove the Turing Completeness of our white-box approach.
- TC–RAG establishes a stack memory system, capable of adaptive retrieval, incorporating composed actions to effectively manage memory, particularly in handling harmful or noisy knowledge.
- We open-source a meticulously curated Chinese medical pretraining dataset, along with extensive medical documents and a comprehensive knowledge graph.
- Our thorough experimental evaluation on seven real-world Medical and general Q&A datasets demonstrates the superior performance of TC–RAG over existing baselines, underscoring its accuracy and explainability. Furthermore, TC–RAG has been deployed on the online hospital platform (anonymous).

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG), introduced by (Lewis et al., 2020), enhances LLM performance on knowledge-intensive tasks by integrating relevant information from external knowledge bases through prompt engineering. RAG not only mitigates hallucination issues during LLM inference but also provides up-to-date, task-specific knowledge, significantly boosting both interpretability and performance on downstream tasks (Izacard et al., 2022; Asai et al., 2023b,a). In the biomedical field, RAG has been widely used to improve LLMs' reasoning and analytical capabilities by leveraging external medical knowledge from sources such as medical papers, textbooks, Wikipedia (Jin et al., 2023; Lála et al., 2023; Zakka et al., 2024; Wang et al., 2023c; Xiong et al., 2024), and knowledge graphs (Soman et al., 2023b; Matsumoto et al., 2024).

Naive & Advanced RAG. Naive RAG typically

follows a simple *retrieve-and-read* approach, where relevant information is retrieved based on the initial user query, and the answer is generated using that content (Soman et al., 2023b,a; Sen et al., 2023; Khandelwal et al., 2020; Borgeaud et al., 2022; Ram et al., 2023). Advanced RAG, however, incorporates more sophisticated components such as retrievers (Qu et al., 2021; Ma et al., 2023a), rerankers (Cheng et al., 2021; Yu et al., 2022), filters (Jiang et al., 2024b), and readers (Yoran et al., 2024; Fang et al., 2024), to improve the quality of both retrieval and generation. However, neither naive nor advanced RAG considers whether the LLM already possesses the necessary knowledge. This often leads to the retrieval of excessive, redundant information, which can mislead the model and cause a "lost in the middle" dilemma (Liu et al., 2023). Our method addresses this by determining whether to retrieve and what to retrieve based on the model's internal parameterized knowledge, resulting in more efficient and accurate retrieval.

Adaptive RAG. Recent research has focused on developing adaptive RAG strategies, enabling LLMs to determine whether and when to retrieve and to select the most appropriate retrieval tools from huge knowledge base. FLARE (Jiang et al., 2023d) predicts the next sentence and uses the generated low-confidence tokens as query to re-retrieve relevant documents. DRAGIN (Su et al., 2024) leverages the LLM's uncertainty in its generated content to decide when to trigger retrieval based on the internal self-attention weights and corresponding keywords. Adaptive-RAG (Jeong et al., 2024) uses a smaller LLM as a classifier to query complexity and subsequently selects the most appropriate retrieval strategy—ranging from simple to advanced. However, these existing adaptive RAG methods are not Turing Complete, lacking the ability to dynamically manage and monitor the retrieval process in a way that guarantees convergence to a reliable conclusion. Moreover, they have yet to fully harness the step-by-step planning and tool-use abilities of LLMs in conjunction with RAG. Our approach addresses these limitations by integrating a Turing Complete framework that optimizes retrieval through advanced planning and tool-use strategies, ensuring more reliable and accurate outcomes.

2.2 Reasoning and Planning Capabilities

Recent advancements have focused on enhancing the reasoning and planning capabilities of LLMs (Zhu et al., 2024; Yao et al., 2022a; Shinn et al., 2024; Koh et al., 2024). One notable approach is Chain-of-Thought (CoT) (Wei et al., 2022; Xia et al., 2024), which demonstrates how LLMs can construct structured "thought processes" to solve complex problems. ReAct(Yao et al., 2022b) integrates reasoning traces with task-specific actions, enabling LLMs to plan, adjust actions, and manage exceptions while gathering information from external sources such as knowledge bases. Reflexion (Shinn et al., 2024) further improves LLMs by using linguistic feedback, allowing them to reflect on and store task feedback, enhancing decision-making in future trials. Despite these advancements, the reasoning and planning processes of LLMs often lead to the accumulation of errors and redundant information. While these methods introduce new decision trials, they frequently fall short in managing previous memory—particularly in removing ineffective decisions or refining historical records. To address these challenges, TC–RAG incorporates a memory stack system with backtrack and summary operations, allowing for timely error correction and condensation of redundant knowledge. This ensures that the model's reasoning process remains efficient and accurate, leading to more reliable outcomes.

3 Prelinmary

Definition 1. (Stack of Memory). In TC–RAG, the Memory of LLMs is conceptualized as a White Box Turing Machine (Turing, 1936), with the rigorous theoretical proof in **Appendix 8.2**. Let $\text{TC} = (\mathcal{S}, \mathcal{A}, \mathcal{M}, \delta, s_0, \mathcal{F}, \sigma)$ represent the stack memory of the LLMs, where:

- $\mathcal{S}$ denotes the set of possible states the LLMs occupy. Specifically, we use numerical values like perplexity (Jelinek et al., 1977) & uncertainty (Peng et al., 2024) to define system's state.
- $\mathcal{A}$ represents the set of actions that the LLMs can perform. Following stack theory, we define the fundamental meta-actions: **push** and **pop**. Additionally, we have composite actions composed of the two meta-actions: "`Thought`", "`Tool_Observation`", "`Plan`", "`Backtrack`", "`Summary`", and "`Conclusion`".
- $\mathcal{M}$ is the infinite stack memory, which includes the set of actions $\mathcal{A}$ and potentially other symbols may be used in stack operations, such as the initiation state "`User_Query`".
- $\delta : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times (0, +\infty)$ is the state transition function. It defines how the LLM transitions to a new state, selects an action, updates the stack memory, and calculates a new system state value based on the current state and stack memory. The transition involves selecting an action $a \in \mathcal{A}$ (which includes push and pop operations), leading to a new stack top and an updated system state value $s \in \mathcal{S}$ within the range $(0, \$\text{Large\_Value}\$)$. δ is decided by LLMs and state calculation function.
- $s_0 \in \mathcal{S}$ is the initial state, with its initial and upper bound value defined as $s_0 = \$\text{Large\_Value}\$$ (where a lower value indicates a more desirable system state).
- $\mathcal{F} \subseteq \mathcal{S}$ is the set of final states. The state $f \in \mathcal{F}$ is deemed final if the current action is "`Conclusion`" and the current state value f is less than the threshold σ. Otherwise, the system will either reset the state value to the threshold σ or change the action from "`Conclusion`" to "`Thought`" for further analysis.
- $\sigma \in (0, +\infty)$ is the hyper-threshold for the final state.

Definition 2. (Meta-Actions). In this part, we define the meta-actions in TC–RAG as follows:

- `Push`: When the medical-LLM processes new information or obtains tool observations, a new action is

determined and pushed onto the stack $\mathcal{M}$:

$$(s_{t+1}, a_{t+1}) \leftarrow \delta(s_t, \mathcal{M}_t), \mathcal{M}_{t+1} \leftarrow push(\mathcal{M}_t, a_{t+1}),$$

where s_{t+1} represents LLM's next system state value, a_{t+1} is the next action to be pushed onto stack $\mathcal{M}$.

- `Pop`: If an error or inconsistency is detected, the LLM needs to revert to the previous state by popping the top element from the stack:

$$(s_{t+1}, a_{t+1}) \leftarrow \delta(s_t, \mathcal{M}_t), \quad \mathcal{M}_{t+1} \leftarrow pop(\mathcal{M}_t).$$

Task Definition. Given the parameterized knowledge Θ embedded within LLM, and non-parameterized knowledge base $\mathcal{D}$ which comprises diverse types of knowledge, the objective is to generate a reliable medical `Response` given the natural language `User_Query`:

$$\texttt{Response} \leftarrow \Theta(\texttt{User\_Query}, \mathcal{D} \mid \mathcal{P}), \tag{1}$$

where $\mathcal{P}$ is the task-specific prompt, *i.e.* CoT's (Wei et al., 2023), ReACT's (Yao et al., 2022a), Trigger Prompt's (Xu et al., 2025a), etc.

Reasons for Stack Memory. The stack design in TC–RAG offers several advantages: ❶ ensures the purity of historical information by removing irrelevant knowledge via pop; ❷ provides $O(1)$ access efficiency for fast retrieval of past information; ❸ memorizes both parameterized and external knowledge, enhancing reasoning capabilities; ❹ allows dynamic control of the reasoning process with options for backtracking and summarizing; ❺ and mimics human-like reasoning, where information is accumulated and conclusions are adjusted as new evidence arises. (cf **Appendix** 8.1 for details).

4 TURING-COMPLETE

We define a stack-based memory system incorporating a system state variable and prove its equivalence to a universal Turing machine T through a series of formal definitions, lemmas, and a main theorem in 5 steps. The detailed proofs are provided in **Appendix 8.2, 8.3**. We prove this by showing that for any Turing machine T, there exists TC that can simulate T.

Step ❶: Construct Turing System For a Turing machine $T = (\mathcal{S}_T, \mathcal{A}_T, \mathcal{M}_T, \delta_T, st_0, st_{accept}, st_{reject})$, we interpret TC $= (\mathcal{S}, \mathcal{A}, \mathcal{M}, \delta, s_0, \mathcal{F}, \sigma)$ with the following components:

1. $\mathcal{S} = \mathcal{S}_T \cup f$, where $f \notin \mathcal{S}_T$ is the end state.
2. $\mathcal{A} = \mathcal{A}_T$, $\mathcal{M} = \mathcal{M}_T \cup \mathcal{A}$, $s_0 = st_0$, $\mathcal{F} = \{f\}$, σ is the hyper-threshold to decide accept or reject.
3. The transition function δ is defined as:

$$\delta(s,a) = \begin{cases} (s', push, s \geq \sigma) & \text{if } \delta_T(st,a) = (st', a, R) \\ (s', pop, s \geq \sigma) & \text{if } \delta_T(st,a) = (st', a, L) \\ (f, no\_op, a, s < \sigma) & \text{if } st \in \{st_{accept}, st_{reject}\} \end{cases}$$

where a is the executing action for T or TC. The operations $push$ and pop are corresponding to the right (R) and left (L) movements in T. Transition function $\delta(\cdot,\cdot)$ is for TC and $\delta_T(\cdot,\cdot)$ is for T. no_op indicates no operation is performed when the system is halting.

Step ❷: Configuration Mapping Next, we define a bijective mapping function h that maps each configuration of T to TC. We now define configuration c of T as $c = (st, w_1 a w_2)$, w_1, w_2 represents the sequence of actions to the left / right of the tape head, where $st \in \mathcal{S}_T$, $a \in \mathcal{A}_T$ is the tape head/action. Configuration of TC c_{TC} is composed of the tuple as (system state, remaining actions, stack memory actions, whether to halt) as:

$$h(st, w_1 a w_2) = (s, w_2, w_1^R a, s \geq \sigma) = c_{\text{TC}}$$

where w_1^R is the reverse of w_1, which is necessary due to Last-In-First-Out (LIFO) principle of $\mathcal{M}$.

Step ❸: Simulation of Computation Steps We prove that TC can simulate each step of T (Lemma 3):

Lemma 1 *If $c_1 \vdash_T c_2$ in T, then $h(c_1) \vdash^*_{\text{TC}} h(c_2)$ in TC, where $*$ denotes one or multiple steps of derivation, $\vdash$ is the action shift operator.*

Proof 4.1 *Let $c_1 = (st_1, w_1 a_1 w_2)$ and $c_2 = (st_2, w_1' a_2 w_2')$ be configurations of T with $a_t = \{L, R\}$. We prove Lemma 3 by considering the two cases $push$ and pop corresponding to the possible directions of tape head movement in T.*

- ***Case 1**: If $\delta_T(st_1, a_1) = (st_2, a_2, R)$, then:*

$$\begin{aligned} h(c_1) &= (st_1, w_2, w_1^R a_1, st_1 \geq \sigma) \\ &\vdash_{\text{TC}} (s_2, w_2', w_1^R a_1 a_2, s_2 \geq \sigma) \quad (\textbf{\textit{push}}\ a_2) \\ &= h(st_2, \underline{w_1 a_1} a_2 w_2') = h(st_2, \underline{w_1'} a_2 w_2') \\ &= h(c_2) \end{aligned}$$

- ***Case 2**: If $\delta_T(st_1, a_1) = (st_2, a_2, L)$, then:*

$$\begin{aligned} h(c_1) &= (st_1, w_2, w_1^R a_1, st_1 \geq \sigma) \\ &\vdash_{\text{TC}} (s_2, w_2, w_1^R, s_2 \geq \sigma) \quad (\textbf{\textit{pop}}\ a_1) \\ &\vdash_{\text{TC}} (s_2, w_2', w_1'^R a_2, s_2 \geq \sigma) \\ &\quad (\textbf{\textit{push}}\ a_2, \text{if } a_2 \neq null) \\ &= h(st_2, w_1' a_2 w_2') = h(c_2) \end{aligned}$$

It's noted that the two cases correspond to meta operations of `push` (Case 1) and `pop` (Case 2) in TC–RAG.

Step ❹: Preservation of Acceptance and Rejection

Lemma 2 *T accepts (rejects) whole input w if and only if TC reaches a configuration $(f, NULL, \mathcal{M}, f < \sigma)$ from the initial configuration $(s_0, w, NULL, s_0 \geq \sigma)$.*

Proof 4.2 *We prove the consistency of reaching the termination state from both $T \rightarrow$ TC and TC $\rightarrow T$ perspectives.*

*(**Forward** $\Rightarrow$) Assume T accepts (rejects) input w. Then: $\exists t \in \mathbb{N}$ such that after t steps, T enters state st_{accept} (or st_{reject}). Then, Let (st_t, w_t) be the configuration of T at step t, where $st_t \in \{st_{accept}, st_{reject}\}$. By our construction of δ, $\forall a \in \mathcal{A}$, for $\forall st \in \{st_{accept}, st_{reject}\}$ we have:*

$$\delta(s,a) = (f, no\_op, a, f < \sigma)$$

Therefore, if $h(st_t, w_t) = (s_t, w_2, w_1^R a, s_t \geq \sigma)$ *is the corresponding configuration in* TC*, then:*

$$(s_t, w_2, w_1^R a, s_t \geq \sigma) \vdash_{\text{TC}}^{t} (f, NULL, \mathcal{M}, f < \sigma)$$

Thus, TC *achieves* f *with state value lower than* σ*.*

(Backward $\Leftarrow$*)** Assume* TC *reaches configuration* $(f, NULL, \mathcal{M}, f < \sigma)$ *from* $(s_0, w, NULL, s_0 \geq \sigma)$*. Then:* $\exists t \in \mathbb{N}$ *such that:*

$$(s_0, w, NULL, s_0 \geq \sigma) \vdash_{\text{TC}}^{t} (f, NULL, \mathcal{M}, f < \sigma).$$

Let $(s_{t-1}, w_2, w_1^R a, s_{t-1} \geq \sigma)$ *be the configuration of* TC *at step* $t-1$*. Then, by the construction of* δ*, the only way to reach* f *is if* $s_{t-1} \in \{st_{accept}, st_{reject}\}$*. Therefore, the corresponding configuration of* T *at step* $t-1$ *must be* $(st_{t-1}, w_1 a w_2)$*, where* $st_{t-1} \in \{st_{accept}, st_{reject}\}$*. Thus,* T *must have entered* st_{accept} *or* st_{reject}*, implying that* T *accepts or rejects* w*.*

From both the ***Forward*** *and* ***Backward*** *proofs,* T *accepts (or rejects) the input* w *if and only if* TC *reaches a configuration* $(f, NULL, \mathcal{M}, f < \sigma)$ *from the initial configuration* $(s_0, w, NULL, s_0 \geq \sigma)$*.*

Step ❺: Convergence of the State Value to σ As detailed in Appendix 8.3, under the *Action Validity Assumption*, we have shown that the Lyapunov function $V(st_j) = \max(0, st_j - \sigma)$ satisfies the following key properties: $V(st_{j+1}) - V(st_j) \leq -\epsilon, \epsilon > 0$, which ensures that the system state s_t decreases monotonically. As a result, the system state will eventually reach the threshold σ, satisfying $st_T < \sigma$ for some finite T. This guarantees finite-time convergence. And the convergence time T satisfies the following bound with approximate $\frac{\log(\delta / V(s_{t_0}))}{\log(1-\beta)}$. Thus, the system halts reliably after a finite number of steps.

By the above lemmas, we show that: ❶ TC can simulate every move of T. ❷ TC halts if and only if T achieve acceptance or rejection behavior. **Thus, TC can simulate any computation of any Turing machine** T**, which by definition makes TC Turing complete**. Theoretically, TC can be considered a **Tape Model**, however, considering the issue of accumulated erroneous and disordered context, in implementation, we allow the observation to the entire memory stack $\mathcal{S}$ (not just the top), but restrict LLMs to modify any position of $\mathcal{S}$.

5 Memory Stack and State Monitor

In this section, we present our innovative framework, TC–RAG, as illustrated in Figure 1. Traditional methods, such as those based on thought chain (Yu et al., 2023; Li et al., 2023a; Ma et al., 2023a; Wei et al., 2023) and reasoning and acting approaches (Yao et al., 2022a; Shinn et al., 2024; Zhu et al., 2024), often suffer from lack of state management, retrieval halting, accumulated erroneous knowledge, token inefficiency, and the issue of being "lost in the middle". To address these challenges, we introduce a stack-based memory system that leverages `push` and `pop` actions for efficient memory management for ***C1, C3***. Additionally, to further improve the LLMs' knowledge comprehension and adaptability, we incorporate an expert knowledge pre-training module based on the general LLM, which enhances the LLMs' understanding and reasoning capabilities (as detailed in **Appendix 8.5**) for ***C2***. Next, in subsection 5.1, we define the stack memory structure and the composite actions such as "`Thought`", "`Tool_Observation`", "`Plan`", "`Backtrack`", "`Summary`", and "`Conclusion`". We then outline the stack states—initiation, intermediate, and final—and provide specific state calculations in subsection 5.2. In **Appendix 8.6**, we describe the prompting strategy and the full algorithm.

5.1 Stack Memory and Composed Actions

In managing memory for LLMs, memory can be conceptualized as a stack structure adhering to the "last-in, first-out" (LIFO) principle. Unlike Markov processes where future actions depend solely on the current actions, in TC–RAG, each new memory entry builds upon the accumulated memory (Pimentel and Silva, 2018; Miller and Chomsky, 2003). To maintain the orderliness and consistency of the memory, erroneous entries are ejected sequentially from the top of the stack. This approach prevents "contamination" and confusion that might arise from directly removing intermediate memories, thereby mitigating unreasonable interference during the reasoning process.

Consequently, as outlined in Definition 1, we leverage the stack $\mathcal{M}$ to simulate the LLM's memory. Based on meta stack operations of push & pop in Section 4, the following composite actions have been devised to enhance reasoning and planning capabilities of LLMs:

- `Thought`: This action represents LLM's analytical process. Once the model has processed and generated an insight, this analysis is **pushed** onto stack $\mathcal{M}$.
- `Tool_Observation`: When the LLM interacts with external tools to gather additional data or insights, the result of this interaction is captured as an observation. Both the tool's name and the resulting observation are then **pushed** onto the stack $\mathcal{M}$.
- `Plan`: Triggered when the LLM identifies a relevant or helpful task, this action occurs when the model makes a decision to outline subsequent actions or strategies. The model **pushed** a new element to the stack $\mathcal{M}$, representing a planned step, decision, or strategy that will guide the next phase of the task.
- `Backtrack`: Triggered by the LLM's reflection mechanism, this action occurs when the top of the stack is found to be irrelevant or harmful to the ongoing task. The model **pops** the top element off the stack $\mathcal{M}$, removing the irrelevant or harmful memory.
- `Summary`: When the content at the top of the stack $\mathcal{M}$ becomes too lengthy or cluttered with irrelevant information, the LLM initiates a summarizing process. The content is first **popped** off the stack, and then the concise summary is **pushed** back onto the $\mathcal{M}$, ensuring that the stack remains focused on the most

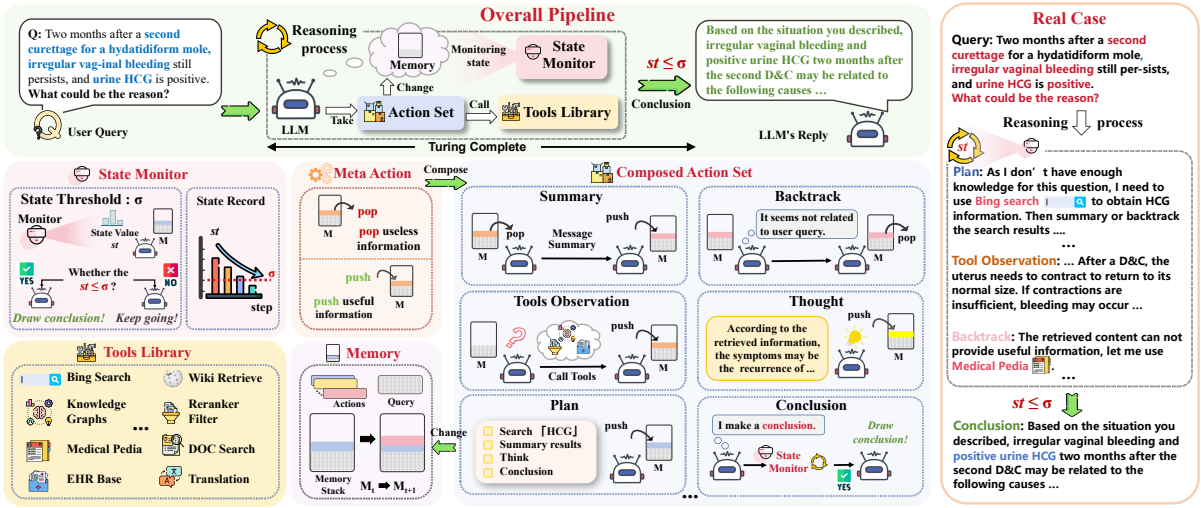

Figure 1: Overall framework of TC–RAG.

pertinent information.

- `Conclusion`: This action occurs when the LLM believes it has reached a final conclusion or solution in a black-box manner. However, if the current state does not satisfy the necessary stopping conditions, the `Conclusion` action is replaced with `Thought`, allowing the model to continue refining its reasoning.

5.2 Stack Status and Memory Management

5.2.1 Pipeline

We will systematically describe the changes in the memory stack and system status.

- **Stack and State Initialization**: Initially, the overall system state value is assigned a large constant as $Large_Value$. The memory stack, $\mathcal{M}$, begins with only the "`User_Query`" action. This entry is immutable, meaning it cannot be popped and permanently resides at the bottom of the stack.
- **State Value Evaluation**: During analytical process, outputs tagged with actions "`Conclusion`" and "`Thought`" are monitored. For each output, a system state value is computed and carried forward in subsequent iterations. This value can be determined using metrics like conditional perplexity and uncertainty, with lower values indicating higher confidence.
- **Conclusion Validation**: If the LLMs output a `Conclusion`, the system evaluates whether the associated state value meets predefined stopping conditions. If not, action `Conclusion` is reclassified as `Thought`, signaling the need for further analysis.
- **State Restoration**: If a `Thought` is popped from the stack (*e.g.*, via `Backtrack` action), the system restores the system state value to its previous level before the addition of that `Thought`. This management strategy ensures the system remains dynamic and responsive to memory, allowing for iterative refinement of LLM's outputs. By continuously evaluating and adjusting based on system state value, LLMs can effectively navigate toward accurate and reliable conclusions.

5.2.2 System Value Calculation

To ensure that the system is progressing toward a reliable conclusion, a system state value is calculated by comparing the content generated by the LLMs at the top of the stack (from either a `Thought` or `Conclusion`) with the bottom "`User_Query`". This comparison, based on metrics such as conditional perplexity and uncertainty, quantifies how closely the system's current reasoning aligns with the original intent of the task:

- **Conditional Perplexity:** Conditional Perplexity is a metric used to evaluate the predictability of a language model given prior context (Jelinek et al., 1977). For the sequence of text $\mathcal{M}_{\text{top}}$ at the top of the $\mathcal{M}$, and the original query at the bottom (denoted as $\mathcal{M}_{\text{bottom}}$), the conditional perplexity $cppl$ is calculated as:

$$cppl(\mathcal{M}_{\text{top}} \mid \mathcal{M}_{\text{bottom}}) = \exp\Big(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_1, \ldots, w_{i-1}, \mathcal{M}_{\text{bottom}})\Big), \quad (2)$$

  where N is the number of tokens in $\mathcal{M}_{\text{top}}$ and $P(w_i \mid w_1, \ldots, w_{i-1}, \mathcal{M}_{\text{bottom}})$ represents the probability of the i-th token w_i in $\mathcal{M}_{\text{top}}$ given the preceding tokens and the context $\mathcal{M}_{\text{bottom}}$.

- **Uncertainty:** uct measures the confidence (Jiang et al., 2023b) in the context $\mathcal{M}_{\text{top}}$ and is derived from the entropy of the output probability distribution:

$$uct(\mathcal{M}_{\text{top}}) = -\sum_{i=1}^{N} P(w_i \mid w_1, \ldots, w_{i-1}) \times \log P(w_i \mid w_1, \ldots, w_{i-1}), \quad (3)$$

  where higher entropy indicates greater uncertainty and less reliability.

The system will then select either $cppl$ or uct as the system state value s_t at step t. If $s_t < \sigma$, it suggests

that the LLM's output is both predictable and confident, indicating alignment with the original query. Vice versa, a high-value signal that the output may not be reliable or relevant, necessitating further refinement. The system will continuously update s_t during reasoning process, using it to guide decisions on whether to finalize or continue refining the output.

5.3 Acceleration Strategy

To mitigate the time overhead associated with multi-turn interactions with LLMs, we employ two acceleration strategies: ❶ **Retrieval and Interaction in the Decoder Stage.** As multi-turn interactions accumulate more contextual information, the increasing length of the context results in repeated encoding of historical context (Key and Value matrices) during the decoding process leading to significant inefficiencies. To address this, we incorporate a retrieval and interaction mechanism during the decoder stage, which interrupts the decoding process after each action is generated by LLMs (Jain et al., 2024; Asai et al., 2024). ❷ **Token Speculative Sampling.** The decoding stage in LLMs is computationally intensive due to the token-by-token generation process. To alleviate this, we fine-tune a smaller language model (with only 1.5B parameters) as a draft model and employ speculative sampling to predict the next token (Leviathan et al., 2023; Chen et al., 2023).

6 Experiments

In this section, we conduct a series of experiments on three medical datasets to answer the following research questions:

- **RQ1** (**Section** 6.2, **Appendix** 8.8.1): Does TC–RAG outperform SOTA RAG methods under the same database source?
- **RQ2** (**Section** 6.3, 6.4): Is the stack framework we designed effective? What impact does each component have on the overall performance?
- **RQ3** (**Appendix** 8.8.2, 8.8.3): Can TC–RAG really pop up erroneous execution memory and noise injection and achieve memory management?

Moreover, we also anonymously rank TC–RAG on the largest Chinese medical CMB Leaderboard, which includes detailed medical scores for each subcategory (c.f. **Appendix 8.10**)[1].

6.1 Experimental Setup

❶ **Datasets.** Our experiments are conducted on two multi-task medical datasets MMCU-Medical (Zeng, 2023) and CMB-Exam (Wang et al., 2023b), and one open-domain Q&A medical dataset CMB-Clin (Wang et al., 2023b). ❷ **RAG Tools.** We incorporate various RAG tools including knowledge graph search, document search, web and pedia search, and electronic medical record database. ❸ **LLM Turbo.** The general-domain LLM Qwen-32B (Yang et al., 2024a) was selected as the base model, which we further pretrained for the medical domain.❹ **Baselines.** To assess the effectiveness of our approach, we compare TC–RAG with twelve baselines. (1) Without RAG: **Base** and **CoT** (Wei et al., 2023). (2) Naive RAG: **SR-RAG** (Soman et al., 2023b), **FL-RAG** (Khandelwal et al., 2020) and **FS-RAG** (Trivedi et al., 2023). (3) Advanced RAG: **CoK** (Li et al., 2023b), **SuRe** (Kim et al., 2024), **HyKGE** (Jiang et al., 2024b). (4) Adaptive RAG: **ReACT** (Yao et al., 2022a), **Self-RAG** (Asai et al., 2024), **FLARE** (Jiang et al., 2023c) and **DRAGIN** (Su et al., 2024). ❺ **Evaluation Metrics.** We use **EM** (Zhu et al., 2021; Karpukhin et al., 2020) metric for multi-task medical choice questions and **ROUGE-R** (Xu, 2023) and **BLEU-1, BLEU-4** (Xu, 2023) for open-domain Q&A tasks. Detailed experimental settings are in **Appendix 8.7**.

6.2 Main Result Analysis (RQ1)

To answer RQ1, we conduct experiments and report results of the accuracy on the MMCU-Medical, CMB-Exam and CMB-Clin datasets with two LLMs in Table1. From the reported accuracy, we can find the following observations:

Comparison of RAG methods and Base LLMs. Given the complexity of patient queries, we observe that the performance of the Naive RAG methods shows little to no improvement compared to the No RAG baselines. In contrast, the effectiveness of Adaptive RAG is significantly higher than that of Advanced RAG and substantially outperforms other approaches. This underscores the necessity of employing more sophisticated submodules for advanced and adaptive retrieval in domain-specific scenarios.

Comparison of TC–RAG and other RAG methods. Our model, TC–RAG, clearly outperforms the baselines across all datasets, with an average performance gain of all metrics of up to **7.20%**. For instance, the EM and BLEU-4 scores improve by approximately 2.60%-5.62% and 8.23%-13.72%, respectively. These results highlight the effectiveness of our modules in system state & memory management, as well as adaptive retrieval. Additionally, it is worth noting that domain-specific LLMs significantly outperform general LLMs, further demonstrating the importance of pre-training a medical LLM to support TC–RAG, in line with ***C2***.

Strong Generalization Ability. Table 4 summarizes the experimental results. TC–RAG demonstrated superior performance compared to the baseline methods in the general domain. These findings reinforce TC–RAG's strong **generalization ability across diverse domains, tasks, languages, model sizes, and model types**, making it a robust framework for RAG tasks.

[1] https://cmedbenchmark.llmzoo.com/static/leaderboard.html

Table 1: Performance comparison (%) on *CMB*, *MMCU* and *CMB-Clin* datasets.

Method	LLM Turbo		Qwen-32B					Pretrained Qwen-32B				
	Type	Dataset	CMB	MMCU	CMB-Clin			CMB	MMCU	CMB-Clin		
		Metric	EM	EM	BLEU-1	BLEU-4	ROUGE	EM	EM	BLEU-1	BLEU-4	ROUGE
Baselines	Without RAG	Base	67.33	80.92	06.84	10.82	24.10	68.34	80.70	06.93	11.24	23.75
		CoT	70.88	80.53	07.23	11.64	24.30	69.94	80.06	07.37	11.67	23.54
	Naive RAG	SR-RAG	66.73	80.31	09.48	14.18	22.23	68.54	83.15	11.56	16.72	22.38
		FL-RAG	67.94	75.17	10.67	15.32	22.96	69.48	78.61	10.95	16.20	23.13
		FS-RAG	64.65	71.62	05.12	08.63	20.29	67.20	75.20	05.76	09.20	21.34
	Advanced RAG	CoK	76.25	79.04	13.96	25.33	31.49	79.18	81.31	14.92	27.84	46.90
		SuRe	76.96	83.08	14.88	27.90	31.06	78.55	85.56	15.26	28.01	44.38
		HyKGE	82.17	86.45	19.02	41.84	38.92	84.58	87.09	22.85	43.79	50.23
	Adaptive RAG	ReACT	82.20	87.34	19.15	48.55	41.22	84.08	88.19	22.37	49.96	50.45
		Self-RAG	81.88	85.78	18.23	46.82	44.70	84.32	86.77	20.06	47.12	48.90
		FLARE	82.89	87.26	19.85	49.19	51.15	85.14	87.37	20.91	50.38	52.96
		DRAGIN	82.78	85.28	19.48	44.18	45.23	84.80	85.46	19.56	46.72	47.38
Ours	Adaptive RAG	**TC–RAG**-*cppl*	84.90	**89.61**	20.86	53.04	53.29	87.33	92.80	24.65	56.94	**57.46**
		TC–RAG-*uct*	**85.63**	89.46	**21.03**	**53.24**	**54.98**	**87.95**	**93.15**	**25.89**	**57.29**	56.59
Ablations	Adaptive-RAG	`w/o Backtrack`	83.44	88.61	20.04	51.38	52.40	86.24	90.67	22.06	52.92	53.88
		`w/o Summary`	84.82	89.00	20.72	52.11	53.07	84.79	88.47	24.48	54.39	54.72
		`w/o State Monitor`	83.75	88.79	19.82	49.44	51.07	85.27	89.04	21.40	48.42	51.86

6.3 Ablation Study (RQ2)

We perform ablation studies to evaluate the impact of each component within TC–RAG, as detailed in Table 1, with three variants: (1) TC–RAG without `Backtrack` action (denoted as `w/o Backtrack`), (2) TC–RAG without `Summary` action (denoted as `w/o Summary`), (3) TC–RAG without State Monitor, relying solely on the LLM's 'Final Answer' action to determine termination, transforming into a black-box system (denoted as `w/o State Monitor`).

The results reveal that each component contributes positively to the overall performance of TC–RAG. The exclusion of any component leads to a noticeable reduction in effectiveness. Particularly, the absence of the State Monitor results in significant performance degradation, highlighting the critical importance of the system state variable in monitoring the process, in line with ***C1***, which is essential for preventing overconfidence and ensuring appropriate termination, thereby avoiding excessive or inadequate retrieval. Moreover, the removal of the `Backtrack` and `Summary` actions underscores the necessity of effective memory management. These actions are crucial for mitigating irrelevant noise and maintaining an optimal system state, aligning with the challenges outlined in ***C3***.

6.4 Effiency Study (RQ2)

To illustrate the effectiveness of our TC–RAG, we conducted a comparative analysis of the time and token overhead between TC–RAG and other RAG approaches based on the Qwen-32B and CMB-Exam datasets, as presented in Table 2.

Our findings reveal that except for models like the Base, which do not require RAG, and those requiring only 1-2 interactions such as CoT and HyKGE, most iterative RAG methods exhibit long average interaction times. Notably, CoK, which lacks adaptive capabilities, is particularly hindered by unrestricted interactions and retrievals, negatively impacting their efficiency. In contrast, methods like ReACT, Self-RAG, FLARE, DRAGIN, and TC–RAG show significantly shorter overheads. For TC–RAG, the combination of **Retrieval and Interaction in the Decoder Stage** and the **Token Speculative Sampling** strategy results in much lower overhead compared to other adaptive RAG models. This reduced overhead is particularly important, as excessive waiting times can prove to be considerably restrictive, especially in real-world medical Q&A scenarios where time is a critical factor. Therefore, achieving a balance between time overhead and model accuracy is crucial. In this regard, TC–RAG emerges as the most efficient and high-performing framework. In summary, since RAG involves a process of continuous trial and error (Barnett et al., 2024), we experimented with numerous strategies and ultimately arrived at TC–RAG.

Table 2: Analysis Comparison of RAG methods. The average duration is computed based on Qwen-32B for the CMB-Exam Dataset.

Method	Avg. Interactions	Avg. Retrievers	Avg. Time (s)	Avg. Token
Base	1	0	7.29	128.89
CoT	1	1	14.37	196.68
HyKGE	2	1	19.76	120.37
CoK	4.31	4.31	125.84	753.02
ReACT	6.84	5.72	130.87	874.29
Self-RAG	5.84	4.22	61.09	567.98
FLARE	4.96	4.96	126.74	785.46
DRAGIN	6.12	6.12	167.95	998.64
TC–RAG	4.78	3.37	50.91	458.82

7 Conclusions and Future Works

We introduce the first Turing-Complete RAG system for medical LLMs, named TC–RAG. By incorporating a monitored state variable, we have developed a stack memory framework that enables more dynamic

and adaptive retrieval processes, effectively addressing the endless and inaccurate retrieval challenges. The TC–RAG framework, with its memory stack system for backtracking and summarizing, effectively reduces the accumulation of erroneous knowledge and irrelevant noise. Our experiments suggest that TC–RAG outperforms existing baselines across multiple real-world medical datasets, showing potential improvements in accuracy and reliability. Furthermore, the successful deployment of TC–RAG on an online platform also highlights its practical value in real-world applications. Future efforts will focus on incorporating more complex composed actions, enabling multi-LLM interactions, and exploring its application in other specialized domains.

Limitations

Despite the promising results, our framework does have some limitations that need to be addressed. First, the computational overhead introduced by the memory stack system and dynamic retrieval processes can be relatively high. While these components are essential for managing complex queries and minimizing error accumulation, they do contribute to increased computational costs. In large-scale or real-time applications, this could present challenges related to processing speed and efficiency. Although we have proposed solutions like speculative sampling to alleviate some of this overhead, further optimization is still needed to enhance performance and scalability in time-sensitive environments. Another limitation is that the current framework is primarily optimized for general medical applications. While it performs well across a variety of tasks, there remains significant potential for improvement in more specialized medical domains. Future work will focus on integrating more complex composed actions, enabling multi-LLM interactions, and expanding the framework's capabilities to accommodate a broader range of specialized medical subfields. This will further enhance its adaptability and performance in these targeted areas.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.U23A20468).

References

Autogpt: Build, deploy, and run ai agents. https://github.com/Significant-Gravitas/AutoGPT.

2023. Bridging general large language models and real-world medical consultation.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023a. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.

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

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. *Preprint*, arXiv:2401.05856.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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 W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.

Odmaa BYAMBASUREN, Yunfei YANG, Zhi-fang SUI, Damai DAI, Baobao CHANG, Sujian LI, and Hongying ZAN. 2020. Preliminary study on the construction of chinese medical knowledge graph. *Journal of Chinese Information Processing*.

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. 2020. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, Online. Association for Computational Linguistics.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *Preprint*, arXiv:2302.01318.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. Unitedqa: A hybrid approach for open domain question answering. *Preprint*, arXiv:2101.00178.

Caitríona L Cox, Benjamin M Miller, Isla Kuhn, and Zoë Fritz. 2021. Diagnostic uncertainty in primary care: what is known about its communication, and what are the associated ethical issues? *Family practice*, 38(5):654–668.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. *Preprint*, arXiv:2401.14887.

Hongxin Ding, Yue Fang, Runchuan Zhu, Xinke Jiang, Jinyang Zhang, Yongxin Xu, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024. 3ds: Decomposed difficulty data selection's case study on llm medical domain adaptation. *arXiv preprint arXiv:2410.10901*.

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

Feiteng Fang, Yuelin Bai, Shiwen Ni, Min Yang, Xiaojun Chen, and Ruifeng Xu. 2024. Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training. *Preprint*, arXiv:2405.20978.

Mehrdad Farahani and Richard Johansson. 2024. Deciphering the interplay of parametric and non-parametric memory in retrieval-augmented language models. *Preprint*, arXiv:2410.05162.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1137–1147, Online. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Preprint*, arXiv:2101.02235.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew

Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *ArXiv preprint*, abs/2306.11644.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *Preprint*, arXiv:2301.00303.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zan Hongying, Li Wenxin, Zhang Kunli, Ye Yajuan, Chang Baobao, and Sui Zhifang. 2020. Building a pediatric medical corpus: Word segmentation and named entity annotation. In *Workshop on Chinese Lexical Semantics*, pages 652–664.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models. *Preprint*, arXiv:2208.03299.

Palak Jain, Livio Baldini Soares, and Tom Kwiatkowski. 2024. From rag to riches: Retrieval interlaced with sequence generation. *Preprint*, arXiv:2407.00361.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *J. Acoust.*

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

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023b. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Xinke Jiang, Rihong Qiu, Yongxin Xu, Wentao Zhang, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024a. Ragraph: A general retrieval-augmented graph learning framework. *NeurIPS 2024*.

Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, and Yasha Wang. 2023a. Think and retrieval: A hypothesis knowledge graph enhanced medical large language models.

Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024b. Hykge: A hypothesis knowledge graph enhanced framework for accurate and reliable medical llms responses. *Preprint*, arXiv:2312.15883.

Xinke Jiang, Dingyi Zhuang, Xianghui Zhang, Hao Chen, Jiayuan Luo, and Xiaowei Gao. 2023b. Uncertainty quantification via spatial-temporal tweedie model for zero-inflated and long-tail travel demand prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23. ACM.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023c. Active retrieval augmented generation. *Preprint*, arXiv:2305.06983.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023d. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Qiao Jin, Robert Leaman, and Zhiyong Lu. 2023. Retrieve, summarize, and verify: how will chatgpt affect information seeking from the medical literature? *Journal of the American Society of Nephrology*, 34(8):1302–1304.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Sci. Data*, 10(1):1.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Sci. Data*, 3(1):1–9.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. *Preprint*, arXiv:1911.00172.

Jaehyung Kim, Sangwoo Mo Jaehyun Nam, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. In *ICLR*.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *ACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *Preprint*, arXiv:2407.01476.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. *Preprint*, arXiv:2211.17192.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation

for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2021. Unified named entity recognition as word-word relation classification. *Preprint*, arXiv:2112.10070.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023a. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *Preprint*, arXiv:2305.13269.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023b. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *Preprint*, arXiv:2305.13269.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023c. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

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

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Zijian Zhang, Feng Tian, and Yefeng Zheng. 2024. Large language model distilling medication recommendation model. *Preprint*, arXiv:2402.02803.

Jiayuan Luo, Songhua Yang, Xiaoling Qiu, Panyu Chen, Yufei Nai, Wenxuan Zeng, Wentao Zhang, and Xinke Jiang. 2024. Kuaiji: the first chinese accounting large language model. *Preprint*, arXiv:2402.13866.

Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2023a. Chain-of-skills: A configurable model for open-domain question answering. *Preprint*, arXiv:2305.03130.

Xinyu Ma, Xu Chu, Zhibang Yang, Yang Lin, Xin Gao, and Junfeng Zhao. 2024. Parameter efficient quasi-orthogonal fine-tuning via givens rotation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 33686–33729. PMLR.

Xinyu Ma, Yasha Wang, Xu Chu, Liantao Ma, Wen Tang, Junfeng Zhao, Ye Yuan, and Guoren Wang. 2023b. Patient health representation learning via correlational sparse prior of medical features. *TKDE*.

Xinyu Ma, Yifeng Xu, Yang Lin, Tianlong Wang, Xu Chu, Xin Gao, Junfeng Zhao, and Yasha Wang. 2025. DRESSing up LLM: Efficient stylized question-answering via style subspace editing. In *The Thirteenth International Conference on Learning Representations*.

Nicholas Matsumoto, Jay Moran, Hyunjun Choi, Miguel E Hernandez, Mythreye Venkatesan, Paul Wang, and Jason H Moore. 2024. Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. *Bioinformatics*, 40(6).

George A. Miller and Noam Chomsky. 2003. Memory and the processing of complex sentences. *Cognitive Science*, 27(4):607–646.

Eslam M Mustafa, Mahmoud M Saad, and Lydia Wahid Rizkallah. 2023. Building an enhanced case-based reasoning and rule-based systems for medical diagnosis. *Journal of Engineering and Applied Science*, 70(1):139.

OpenAI. 2022. Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Ankit Pal and Malaikannan Sankarasubbu. 2023. Gemini goes to med school: Exploring the capabilities of multimodal large language models on medical challenge problems and hallucinations.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. *Preprint*, arXiv:2401.12087.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is turing complete. *J. Mach. Learn. Res.*, 22(1).

Daniel S. Pimentel and Marco A. Silva. 2018. A review of memory management techniques in computing systems. *ACM Computing Surveys (CSUR)*, 51(3):1–35.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eicu collaborative research database, a freely available multi-center database for critical care research. *Sci. Data*, 5(1):1–13.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2010.08191.

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

Priyanka Sen, Sandeep Mavadia, and Amir Saffari. 2023. Knowledge graph-augmented language models for complex question answering. *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Karthik Soman, Peter W Rose, John H Morris, Rabia E Akbas, Brett Smith, Braian Peetoom, Catalina Villouta-Reyes, Gabriel Cerono, Yongmei Shi, Angela Rizk-Jackson, Sharat Israni, Charlotte A Nelson, Sui Huang, and Sergio E Baranzini. 2023a. Biomedical knowledge graph-enhanced prompt generation for large language models. *Preprint*, arXiv:2311.17330.

Karthik Soman, Peter W Rose, John H Morris, Rabia E Akbas, Brett Smith, Braian Peetoom, Catalina Villouta-Reyes, Gabriel Cerono, Yongmei Shi, Angela Rizk-Jackson, et al. 2023b. Biomedical knowledge graph-enhanced prompt generation for large language models. *arXiv preprint arXiv:2311.17330*.

Inhwa Song, Sachin R. Pendse, Neha Kumar, and Munmun De Choudhury. 2024. The typing cure: Experiences with large language model chatbots for mental health support. *Preprint*, arXiv:2401.14362.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. *arXiv preprint arXiv:2403.10081*.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. 2025. Understanding chain-of-thought in LLMs through information theory.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *Preprint*, arXiv:2212.10509.

Alan Mathison Turing. 1936. On computable numbers, with an application to the entscheidungsproblem. *Journal of Mathematics*, 58(345-363):5.

Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. 2024. Gptvoicetasker: Llm-powered virtual assistant for smartphone. *Preprint*, arXiv:2401.14268.

Haochun Wang, Chi Liu, Nuwa Xi, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. 2023a. Huatuo: Tuning llama model with chinese medical knowledge.

Sheng Wang, Zihao Zhao, Xi Ouyang, Qian Wang, Dinggang Shen, and Lesion Segmentor. Chatcad: Interactive computer-aided diagnosis on medical image using large language models.

Xidong Wang, Guiming Hardy Chen, Dingjie Song, Zhiyi Zhang, Zhihong Chen, Qingying Xiao, Feng Jiang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. 2023b. Cmb: A comprehensive medical benchmark in chinese. *Preprint*, arXiv:2308.08833.

Yubo Wang, Xueguang Ma, and Wenhu Chen. 2023c. Augmenting black-box llms with medical textbooks for clinical question answering. *arXiv preprint arXiv:2309.02233*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

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

Yuwei Xia, Ding Wang, Qiang Liu, Liang Wang, Shu Wu, and Xiao-Yu Zhang. 2024. Chain-of-history reasoning for temporal knowledge graph forecasting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16144–16159, Bangkok, Thailand. Association for Computational Linguistics.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2402.13178*.

Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, Julian Mcauley, and Diego Diego. Small models are valuable plug-ins for large language models.

Yongxin Xu, Xinke Jiang, Xu Chu, Rihong Qiu, Yujie Feng, Hongxin Ding, Junfeng Zhao, Yasha Wang, and Bing Xie. 2025a. Dearllm: Enhancing personalized healthcare via large language models-deduced feature correlations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 941–949.

Yongxin Xu, Xinke Jiang, Xu Chu, Yuzhen Xiao, Chaohe Zhang, Hongxin Ding, Junfeng Zhao, Yasha Wang, and Bing Xie. 2024. Protomix: Augmenting health status representation learning via prototype-based mixup. In *SIGKDD*, pages 3633–3644.

Yongxin Xu, Ruizhe Zhang, Xinke Jiang, Yujie Feng, Yuzhen Xiao, Xinyu Ma, Runchuan Zhu, Xu Chu, Junfeng Zhao, and Yasha Wang. 2025b. Parenting: Optimizing knowledge selection of retrieval-augmented language models with parameter decoupling and tailored tuning. *ACL 2025*.

Zhichao Xu. 2023. Context-aware decoding reduces hallucination in query-focused summarization. *Preprint*, arXiv:2312.14335.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize

Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Songhua Yang, Xinke Jiang, Hanjie Zhao, Wenxuan Zeng, Hongde Liu, and Yuxiang Jia. 2024b. Faima: Feature-aware in-context learning for multi-domain aspect-based sentiment analysis. In *COLING*.

Songhua Yang, Hanjia Zhao, Senbin Zhu, Guangyu Zhou, Hongfei Xu, Yuxiang Jia, and Hongying Zan. 2023. Zhongjing: Enhancing the chinese medical capabilities of large language model through expert feedback and real-world multi-turn dialogue.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022a. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. *Preprint*, arXiv:2310.01558.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *Preprint*, arXiv:2110.04330.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *Preprint*, arXiv:2311.09210.

Cyril Zakka, Rohan Shad, Akash Chaurasia, Alex R Dalal, Jennifer L Kim, Michael Moor, Robyn Fong, Curran Phillips, Kevin Alexander, Euan Ashley, et al. 2024. Almanac—retrieval-augmented language models for clinical medicine. *NEJM AI*, 1(2):AIoa2300068.

Hui Zeng. 2023. Measuring massive multitask chinese understanding. *Preprint*, arXiv:2304.12986.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Jianquan Li, Guiming Chen, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, and Benyou Wang. 2023. Huatuogpt, towards taming language model to be a doctor.

Ningyu Zhang, Mosha Chen, Zhen Bi, Xiaozhuan Liang, Lei Li, Xin Shang, Kangping Yin, Chuanqi Tan, Jian Xu, Fei Huang, Luo Si, Yuan Ni, Guotong Xie, Zhifang Sui, Baobao Chang, Hui Zong, Zheng Yuan, Linfeng Li, Jun Yan, Hongying Zan, Kunli Zhang, Buzhou Tang, and Qingcai Chen. 2022. CBLUE: A Chinese biomedical language understanding evaluation benchmark. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7888–7915, Dublin, Ireland. Association for Computational Linguistics.

Ruizhe Zhang, Yongxin Xu, Yuzhen Xiao, Runchuan Zhu, Xinke Jiang, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024. Knowpo: Knowledge-aware preference optimization for controllable knowledge selection in retrieval-augmented language models. *Preprint*, arXiv:2408.03297.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *Preprint*, arXiv:2101.00774.

He Zhu, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2023. A medical domain visual question generation model via large language model.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *Preprint*, arXiv:2403.03101.

8 Appendix

8.1 The Reasons of Choosing Stack Memory

The stack design in TC–RAG has the following five key advantages:

- **Ensuring the Purity of Historical Information**: The stack uses the pop operation to eliminate incorrect or irrelevant knowledge, ensuring that each reasoning step is based on accurate information, preventing the accumulation of erroneous data. This way, the stack remains pure, guaranteeing the reliability and consistency of the reasoning process.

- **Efficient Access**: The stack provides $O(1)$ access efficiency, meaning that accessing historical information is done in constant time. In complex query processing, the stack can quickly provide the most recent context or reasoning step, ensuring that the reasoning process is not delayed by information retrieval.

- **Storing Multiple Types of Knowledge**: The stack can store both the model's parameterized knowledge via `Thought` or `Conclusion` and external non-parameterized knowledge via `Tool_Observation` or `Summary` (Farahani and Johansson, 2024). This design makes the stack a flexible, multi-level storage structure that integrates multiple sources of knowledge, enhancing the model's reasoning capabilities.

- **Flexible Reasoning Control**: The stack dynamically manages the reasoning process. The model decides whether to continue reasoning or conclude based on the state of the information in the stack. By controlling the stack, the model can backtrack, summarize, or terminate reasoning when necessary, improving the flexibility and controllability of the reasoning process and avoiding over-reasoning or premature conclusions.

- **Human-like Reasoning**: The stack's "Last In, First Out" (LIFO) nature effectively simulates the way human experts (i.e. doctors) reason during medical diagnoses or decision-making. Experts typically accumulate information step by step and adjust their conclusions when new evidence arises. This design of the stack makes the model's reasoning process more aligned with real-world decision-making, enabling it to handle multi-step reasoning and complex decisions more effectively.

8.2 Full Proof of Turing-Complete

We prove this by showing that for any Turing machine T, there exists our TC that can simulate T:

Let $\mathbb{N}$ denote the set of natural numbers, and $\mathbb{R}$ the set of real numbers. We consider computations over finite alphabets and prove the Turing completeness of our proposed model.

Definition 1. (Standard Turing Machine). A standard Turing Machine is a 7-tuple Turing machine $T = (\mathcal{S}_T, \mathcal{A}_T, \mathcal{M}_T, \delta_T, st_0, st_{accept}, st_{reject})$ where:

- $\mathcal{S}_T$ is a finite set of states.
- $\mathcal{A}_T \subset \mathcal{M}_T$ is the input alphabet.
- $\mathcal{M}_T$ is the tape alphabet, with $\sqcup \in \mathcal{M}_T \setminus \mathcal{A}_T$ as the blank symbol.
- $\delta_T : \mathcal{S}_T \times \mathcal{M}_T \to \mathcal{S}_T \times \mathcal{M}_T \times \{L, R\}$ is the transition function.
- $st_0 \in \mathcal{S}_T$ is the initial state, and its initial value and upper bound is $Large_Value$.
- $st_{accept}, st_{reject} \in \mathcal{S}_T$ are the accept and reject states, respectively.

Definition 2. (T Configuration). A configuration of T is a tuple $c_T = (st, w_1 a w_2)$, where:

- $st \in \mathcal{S}_T$ is the current system state value,
- $w1, w2$ represents the sequence of actions $a \in \mathcal{A}_T$ to the left / right of the tape head.

Definition 3. (TC Configuration). A configuration of TC is a tuple $c_{\text{TC}} = (s, w, \mathcal{M}, \{halt \mid\mid continue\}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \{halt, continue\}$, where:

- $s \in \mathcal{S}$ is the current system state value, and its initial value and upper bound is $Large_Value$.
- w is the remaining input actions,
- $\mathcal{M}$ is the current stack content,
- $\{halt \mid\mid continue\}$ is the conditions that determine whether the system will continue or halt.

Definition 4. (Computation Step). For configurations of TC at step t $c_t = (s_t, w_t, \mathcal{M}_t, s_t \geq \sigma)$ and $c_{t+1} = (s_{t+1}, w_{t+1}, \mathcal{M}_{t+1}, \cdot)$, we say $c_t \vdash_{\text{TC}} c_{t+1}$ if and only if:

1. $w_t = a w_{t+1}$ for some $a \in \mathcal{A} \cup \{\varepsilon\}$
2. $\delta(s_t, a) = (s_2, op, b)$
3. $\mathcal{M}_{t+1} = \begin{cases} push(\mathcal{M}_t, b) & \text{if } op = push \\ pop(\mathcal{M}_t) & \text{if } op = pop \end{cases}$
4. The computation terminates if $s_{t+1} < \sigma$

where $top(\mathcal{M})$ returns the top element of the stack $\mathcal{M}$.

Next, we will break down this proof process into the following steps:

Step 1: Construction of the Turing System. Given a Turing machine $T = (\mathcal{S}_T, \mathcal{A}_T, \mathcal{M}_T, \delta_T, st_0, st_{accept}, st_{reject})$, we interpret TC $= (\mathcal{S}, \mathcal{A}, \mathcal{M}, \delta, s_0, \mathcal{F}, \sigma)$ with the following components:

1. $\mathcal{S} = \mathcal{S}_T \cup f$, where $f \notin \mathcal{S}_T$ is the end state.
2. $\mathcal{A} = \mathcal{A}_T$, $\mathcal{M} = \mathcal{M}_T \cup \mathcal{A}$, $s_0 = st_0$, $\mathcal{F} = \{f\}$, σ is the hyper-threshold to decide accept or reject.
3. The transition function δ is defined as:

$$\delta(s,a) = \begin{cases} (s', push, s \geq \sigma) & \text{if } \delta_T(st,a) = (st', a, R) \\ (s', pop, s \geq \sigma) & \text{if } \delta_T(st,a) = (st', a, L) \\ (f, no\_op, a, s < \sigma) & \text{if } st \in \{st_{accept}, st_{reject}\} \end{cases}$$

where a is the executing action for T or TC. The operations $push$ and pop are corresponding to the right (R) and left (L) movements in T. Transition function $\delta(\cdot,\cdot)$ is for TC and $\delta_T(\cdot,\cdot)$ is for T. The operation no_op indicates no operation is performed when the system is halting. This $\delta(\cdot,\cdot)$ decides the next state value and action (R, L, no_op).

Step 2: Configuration Mapping Next, we define a bijective mapping function h that maps each configuration of T to TC. We now define the configuration c of T as $c = (st, w_1 a w_2)$, w_1, w_2 represents the sequence of actions to the left / right of the tape head, where $st \in \mathcal{S}_T$, $a \in \mathcal{A}_T$ is the tape head/action. The configuration of TC c_{TC} is composed of the tuple as (system state, remaining actions, stack memory actions, whether to halt) as:

$$h(st, w_1 a w_2) = (s, w_2, w_1^R a, s \geq \sigma) = c_{\text{TC}}$$

where w_1^R is the reverse of w_1, which is necessary due to Last-In-First-Out (LIFO) principle of $\mathcal{M}$.

Step 3: Simulation of Computation Steps We now prove that TC can simulate each step of T as in Lemma 3:

Lemma 3 *If $c_1 \vdash_T c_2$ in T, then $h(c_1) \vdash^*_{\text{TC}} h(c_2)$ in* TC*, where $*$ denotes one or multiple steps of derivation, $\vdash$ is the action shift operator.*

Proof 8.1 *Let $c_1 = (st_1, w_1 a_1 w_2)$ and $c_2 = (st_2, w_1' a_2 w_2')$ be configurations of T with $a_t = \{L, R\}$. We prove Lemma 3 by considering the two cases $push$ and pop corresponding to the possible directions of tape head movement in T.*

- ***Case 1**: If $\delta_T(st_1, a_1) = (st_2, a_2, R)$, then:*

$$\begin{aligned} h(c_1) &= (st_1, w_2, w_1^R a_1, st_1 \geq \sigma) \\ &\vdash_{\text{TC}} (s_2, w_2', w_1^R a_1 a_2, s_2 \geq \sigma) \quad (\textbf{\textit{push}}\ a_2) \\ &= h(st_2, \underline{w_1 a_1} a_2 w_2') = h(st_2, \underline{w_1'} a_2 w_2') = h(c_2) \end{aligned}$$

- ***Case 2**: If $\delta_T(st_1, a_1) = (st_2, a_2, L)$, then:*

$$\begin{aligned} h(c_1) &= (st_1, w_2, w_1^R a_1, st_1 \geq \sigma) \\ &\vdash_{\text{TC}} (s_2, w_2, w_1^R, s_2 \geq \sigma) \quad (\textbf{\textit{pop}}\ a_1) \\ &\vdash_{\text{TC}} (s_2, w_2', w_1'^R a_2, s_2 \geq \sigma) \quad (\textbf{\textit{push}}\ a_2, \textit{if } a_2 \neq \textit{null}) \\ &= h(st_2, w_1' a_2 w_2') = h(c_2) \end{aligned}$$

It is noted that the two cases correspond to the meta operations of push (Case 1) and pop (Case 2) in TC–RAG.

Step 4: Preservation of Acceptance and Rejection

Lemma 4 *T accepts (rejects) the whole input w if and only if* TC *reaches a configuration $(f, NULL, \mathcal{M}, f < \sigma)$ from the initial configuration $(s_0, w, NULL, s_0 \geq \sigma)$.*

Proof 8.2 *We prove the consistency of reaching the termination state from both $T \rightarrow$ TC and TC $\rightarrow T$ perspectives.*

*(**Forward** $\Rightarrow$) Assume T accepts (rejects) input w. Then: $\exists t \in \mathbb{N}$ such that after t steps, T enters state st_{accept} (or st_{reject}). Then, Let (st_t, w_t) be the configuration of T at step t, where $st_t \in \{st_{accept}, st_{reject}\}$. By our construction of δ, $\forall a \in \mathcal{A}$, for $\forall st \in \{st_{accept}, st_{reject}\}$ we have:*

$$\delta(s,a) = (f, no\_op, a, f < \sigma)$$

Therefore, if $h(st_t, w_t) = (s_t, w_2, w_1^R a, s_t \geq \sigma)$ is the corresponding configuration in TC*, then:*

$$(s_t, w_2, w_1^R a, s_t \geq \sigma) \vdash^t_{\text{TC}} (f, NULL, \mathcal{M}, f < \sigma)$$

Thus, TC *achieve f with system state value lower than σ.*

*(**Backward** $\Leftarrow$) Assume* TC *reaches configuration $(f, NULL, \mathcal{M}, f < \sigma)$ from $(s_0, w, NULL, s_0 \geq \sigma)$. Then: $\exists t \in \mathbb{N}$ such that:*

$$(s_0, w, NULL, s_0 \geq \sigma) \vdash^t_{\text{TC}} (f, NULL, \mathcal{M}, f < \sigma).$$

Let $(s_{t-1}, w_2, w_1^R a, s_{t-1} \geq \sigma)$ be the configuration of TC *at step $t-1$. Then, by the construction of δ, the only way to reach f is if $s_{t-1} \in \{st_{accept}, st_{reject}\}$. Therefore, the corresponding configuration of T at step $t-1$ must be $(st_{t-1}, w_1 a w_2)$, where $st_{t-1} \in \{st_{accept}, st_{reject}\}$. Thus, T must have entered st_{accept} or st_{reject}, implying that T accepts or rejects w.*

*From both the **Forward** and **Backward** proof, T accepts (rejects) input w if and only if* TC *reaches a configuration $(f, NULL, \mathcal{M}, f < \sigma)$ from the initial configuration $(s_0, w, NULL, s_0 \geq \sigma)$.*

By the above lemmas, we have shown that: (1) TC can simulate every move of T. (2) TC halts if and only if T achieve acceptance or rejection behavior. **Thus, TC can simulate any computation of any Turing machine T, which by definition makes TC Turing complete**, which is consistent with the LLM's backbone——transformers are Turing-Complete (Pérez et al., 2021).

8.3 Convergence Analysis of State Value

In this section, we rigorously analyze the convergence behavior of the TC–RAG model. We focus on two key aspects: the role of information gain in reducing uncertainty and the application of Lyapunov stability theory to establish finite-time convergence. To rigorously prove the finite-time convergence of the TC–RAG process, we analyze its dynamics using Lyapunov stability theory.

Definition of the Lyapunov Function We define a Lyapunov function that quantifies the deviation of the system state st_t from a convergence threshold σ:

$$V(st_j) = \max(0, st_j - \sigma), \tag{4}$$

where σ is a predefined threshold indicating sufficient certainty about the final answer Y. When $st_j \leq \sigma$, the system is considered to have converged.

Properties of the Lyapunov Function The function $V(st_j)$ possesses the following properties:

1. **Non-negativity:** By definition, $V(st_j) \geq 0$ for all $j \geq 0$, and $V(st_j) = 0$ if and only if $st_j \leq \sigma$.
2. **Monotonic Decrease:** Our objective is to prove that $V(st_j)$ decreases over time, i.e.,

$$V(st_{j+1}) - V(st_j) \leq 0. \tag{5}$$

Next, first we will provide the *Action Validity Assumption*, and then we will prove these two properties respectively.

8.3.1 Assumption of Action Validity

Assumption 1 (Action Validity): *Leveraging the robust task comprehension capabilities of LLMs (Ton et al., 2025), we assume that every correctly executed action—such as* push *and* pop *operations—yields meaningful information gain. This gain enhances the system's ability to accurately predict the final answer. In contrast, when an incorrect action occurs, the system employs a* pop *operation to discard the erroneous information. This mechanism ensures that only relevant and accurate content is retained throughout the retrieval-augmented and reasoning processes.*

8.3.2 Uncertainty Reduction

To formalize the reduction in uncertainty over time, we analyze the system's behavior through the lens of mutual information. Let Y denote the final answer and st_j represent the memory state at step j. The mutual information between Y and st_j is defined as:

$$I(Y; st_j) = \mathbb{E}_{p(y,st_j)}\left[\log \frac{p(y, st_j)}{p(y)p(st_j)}\right]. \tag{6}$$

Our goal is to show that after each action a_t, the uncertainty regarding Y is reduced. More precisely, we aim to demonstrate:

$$\mathbb{E}\left[\log p(Y \mid st_j)\right] - \mathbb{E}\left[\log p(Y \mid st_{j-1})\right] \tag{7}$$
$$= I(Y; st_j \mid st_{j-1}). \tag{8}$$

The expectation $\mathbb{E}\left[\log p(Y \mid st_j)\right]$ measures how well the model fits the true labels given the state st_j, representing the expected log-likelihood. This value reflects the information gain of the content at step j with respect to Y. If we can show that this value at step j is higher than at step $j-1$, it would indicate that the model's inference is progressively approaching the true distribution. Thus, we assume that the state transitions follow a Markov process, where st_j depends only on st_{j-1} and the previous state could be observed $p(st_{j-1}|st_j) = 1$. This Markov structure also implies that the observation Y is conditionally independent of the previous state st_{j-1} given the current state st_j, expressed formally as $Y \perp st_{j-1} \mid st_j$. This leads to the following:

Proof:

$$\mathbb{E}\left[\log p(Y \mid st_j)\right] - \mathbb{E}\left[\log p(Y \mid st_{j-1})\right] \tag{9}$$
$$= \mathbb{E}\left[\log \frac{p(Y \mid st_j)}{p(Y \mid st_{j-1})}\right]$$
$$= \mathbb{E}\left[\log \frac{p(Y, st_j \mid st_{j-1})}{p(Y \mid st_{j-1})\, p(st_j \mid st_{j-1})}\right]$$
$$= I(Y; st_j \mid st_{j-1}). \tag{10}$$

In summary, the behavior of the conditional mutual information after each action a_t is as follows:

$$I(Y; st_j \mid st_{j-1}) = \begin{cases} > 0, & \text{if } a_t = \textsf{push}, \\ \geq 0, & \text{if } a_t = \textsf{pop}. \end{cases} \tag{11}$$

Thus, with every action, mutual information either increases or remains non-decreasing, ensuring a progressive reduction in uncertainty. Specifically, push operations incorporate task-relevant knowledge, while pop operations remove noisy or irrelevant information.

8.3.3 Lyapunov Stability and Finite-Time Convergence of TC–RAG

Impact of Actions on the Lyapunov Function Each action a_t (whether push or pop) influences the system's uncertainty, and consequently, the Lyapunov function, which implies that the Lyapunov function satisfies:

$$V(st_{j+1}) - V(st_j) \tag{12}$$
$$= \max(0, st_{j+1} - \sigma) - \max(0, st_j - \sigma) \tag{13}$$
$$\leq st_{j+1} - st_j \leq 0. \tag{14}$$

Thus, the Lyapunov function is non-increasing, ensuring that the system state progressively approaches the convergence threshold.

Proof of Finite-Time Convergence To establish finite-time convergence, we assume that the decrease in $V(st_j)$ after each action is bounded below by a positive constant. That is, there exists an $\epsilon > 0$ such that for all t:

$$V(st_{j+1}) - V(st_t) \leq -\epsilon. \tag{15}$$

Theorem 1 (Finite-Time Convergence). There exists a finite time T such that: $V(st_T) = 0$.

Proof. (By contradiction) Suppose that $V(st_j) > 0$ for all t. Then, due to the bounded decrease, for any $n \in \mathbb{N}$,

$$V(st_n) \leq V(st_0) - n\epsilon. \tag{16}$$

Choosing $n > \frac{V(st_0)}{\epsilon}$ yields: $V(st_n) < 0$, which contradicts the non-negativity of $V(st_j)$. Hence, there must exist a finite T such that $V(st_j) = 0$.

Convergence Time Bound The convergence time T satisfies:

$$T \leq \left\lceil \frac{V(st_0)}{\epsilon} \right\rceil. \tag{17}$$

Alternatively, if the decrease is proportional, i.e., $V(s_{t_{j+1}}) \leq (1-\beta)V(s_{t_j})$, for some $\beta > 0$, then by repeated application we obtain $V(s_{t_j}) \leq V(s_{t_0})(1-\beta)^T$. To guarantee that $V(s_{t_j}) \leq \delta$ for a small threshold δ, it is necessary that

$$T \geq \frac{\log(\delta/V(s_{t_0}))}{\log(1-\beta)}.$$

However, experimental observations indicate that the behavior of $V(s_t)$ actually unfolds in two stages. Initially, the decay is exponential in nature, which can be modeled as $V(s_t) = V(s_{t_0})e^{-\lambda t}$, so that ensuring $V(s_t) \leq \delta$ requires

$$T \geq \frac{\ln(V(s_{t_0})/\delta)}{\lambda}.$$

After this initial phase, the decay tends to become nearly linear. Therefore, the overall time cost T is estimated to be composed of these two parts: an initial exponential decay phase followed by a subsequent nearly linear decay phase.

Conclusion By defining the Lyapunov function $V(st_j) = \max(0, st_j - \sigma)$ and carefully analyzing the effects of both push and pop operations, we have demonstrated that the uncertainty in the TC–RAG process decreases over time. Moreover, under the assumption of a bounded decrease per step, the system is guaranteed to converge to the target state within a finite number of steps. This theoretical framework underscores the robustness of the TC–RAG and highlights the importance of effectively managing information gain to achieve both efficiency and accuracy.

8.4 Review of LLMs

Definition (Large Language Models). Generative LLMs are powerful language models capable of generating coherent and contextually relevant text. Through pretraining on large-scale text corpora and alignment fine-tuning to follow human instructions, they can generate human-like text based on given prompts or inputs. Typically, LLMs Θ model the probability of a sentence (*i.e.*, a sequence of word tokens) $l = (w_1, w_2, \ldots, w_n)$ as $P(s;\Theta) = \prod_i^n P(w_i \mid w_{<i};\Theta)$, where w_i denotes the i-th token w_i of the sentence l and $w_{<i}$ denotes the partial word token sequence before the i-th step.

8.5 Model Pretrain

8.5.1 Pretrain LLMs

The high-quality pre-training corpus can greatly improve the performance of LLM and even break the scaling laws to some extent (Gunasekar et al., 2023; Ding et al., 2024). Among them, continuous pre-training is a crucial phase where the language model undergoes extensive training on vast and diverse unlabeled datasets. This process spans multiple iterations, each aimed at refining the model's language understanding capabilities (Luo et al., 2024). Initially, the LLM is initialized with the pre-trained weights of basic LLMs and learns to predict missing words or segments within sentences using self-supervised learning objectives such as masked language modeling (MLM) and next-sentence prediction (NSP). Through exposure to a wide variety of textual data sources, the model gradually acquires a rich domain understanding of language structure, semantics, and context in the medical domain. Additionally, techniques like attention mechanisms and multi-layer architectures are employed to capture complex linguistic patterns and dependencies.

It should be noted that we did not use supervised fine-tuning because it would lead to overconfidence in the large model, resulting in the large model often receiving diagnostic results directly without planning and calling professional medical knowledge retrieval tools.

Table 3: Medical pre-train data statistics

Type	Dataset	Size
Dialogues	RealHospital-QA	2318 MB
	Web-QA	567 MB
Knowledge graphs	Medical-KG	379 MB
Exams	Medical-Exam	443 MB
Textbooks	Chinese-Textbook	52 MB
	English-Textbook	212 MB
Guidelines	Med-Guidelines	878 MB
Encyclopedia	Med-Encyclopedia	798 MB
Total		**5647 MB**

8.5.2 Data Preparement

To build a diverse medical corpus, we compiled data from multiple sources, ensuring broad coverage across various medical domains.

- **Dialogues**: The **RealHospital-QA** dataset includes real-world clinical conversations, while **Web-QA** provides Q&A pairs from online health forums, capturing common public inquiries.
- **Knowledge Graphs**: **Medical-KG** organizes medical knowledge into entities and relationships, integrating data from clinical guidelines, research papers, and textbooks.
- **Exams**: The **Medical-Exam** dataset consists of questions from medical exams, aiding the model in handling complex diagnostic scenarios.
- **Textbooks**: We included **Chinese-Textbook** and **English-Textbook** datasets to provide foundational knowledge in both languages.

- **Guidelines**: **Med-Guidelines** comprises official medical guidelines, essential for evidence-based practice.
- **Encyclopedia**: The **Med-Encyclopedia** offers concise explanations of medical terms and conditions.

These datasets span multiple specialties, giving the model a comprehensive understanding of medical knowledge. The total corpus size is 5647 MB, as shown in Table 3.

8.5.3 Pretrain Loss Function

The pertaining loss function is defined as follows:

$$\mathcal{L}_{\textbf{pretrain}} = \sum_{t \in \text{masked}} \log P(w_t \mid l_{\setminus t}; \Theta), \qquad (18)$$

where $l_{\setminus t}$ represents the remaining part of the sequence l after masking the t-th word.

8.5.4 Training Setup

We performed the continuous pre-training on Qwen1.5-32B-Chat using the aforementioned medical datasets. The pre-training was conducted on eight H100 GPUs for one epoch with a learning rate of 1e-4, and the entire training process spanned 4 days. This configuration was chosen to balance computational efficiency with the need for thorough learning, allowing the model to effectively internalize the extensive medical knowledge embedded in the training data. Through this process, we aimed to enhance the model's reasoning and planning capabilities, ensuring it can provide accurate and reliable medical analysis in real-world applications.

8.6 Prompt and Algorithm

8.6.1 Prompt Format.

In this module, we will provide a detailed introduction to the Prompt used in our entire model in the following Prompt:

8.6.2 TC–RAG Algotithm.

Algorithm 1 describes the reasoning loop of TC–RAG for generating a final answer based on user query using a pre-trained Medical LLM. The process begins by initializing the stack memory and the initial state (Lines 1-2). The user query is then pushed into the stack memory (Line 3).

Within the loop, which continues until the action limit is reached or the system state value drops below the threshold σ, the model generates an action based on the current memory stack (Lines 5-6). If the action type is identified as a `Conclusion`, the system state is recalculated; if it remains within acceptable bounds, the final answer is confirmed and pushed into the stack, terminating the loop (Lines 7-11). Otherwise, the final answer is reclassified as a `Thought`, and processing continues (Lines 12-14).

If the action type is a `Thought`, the system state is updated, and the thought result is pushed into the stack (Lines 15-17). For `Tool_Use`, the model retrieves relevant observations using specified tools or a knowledge base, which are then pushed into the stack (Lines 18-20). When the action type is `Backtrack`, the top of the stack is popped, and the previous system state is restored (Lines 21-23). If the action is a `Summary`, the stack is adjusted by popping irrelevant content, summarizing it, and pushing the summary back onto the stack (Lines 24-26).

Finally, after exiting the loop, the top of the stack is returned as the final answer (Lines 27-28).

Algorithm 1 TC–RAG Inference Algorithm

Require: $User\_Query$, pretrained Medical LLM Θ, knowledge base $\mathcal{D}$, system value threshold σ, TC–RAG Prompt $\mathcal{P}$
Ensure: $Conclusion$
1: Initialize stack memory $\mathcal{M}$, state $s \leftarrow$ $Large_Value$
2: Push $User\_Query$ into $\mathcal{M}$
3: $actions\_taken \leftarrow 0$
4: **while** $actions\_taken < max\_loop$ **and** $s \geq \sigma$ **do**
5: $Action \leftarrow \Theta(\mathcal{M} \mid \mathcal{P})$
6: **if** $Action$ is `Conclusion` **then**
7: $f \leftarrow$ calculate_state($User\_Query, Action$)
8: **if** $f < \sigma$ **then**
9: Push $Action$ into $\mathcal{M}$
10: **Break**
11: **else**
12: $action\_type \leftarrow$ `Thought`, $s \leftarrow f$
13: Push $Action$ into $\mathcal{M}$
14: **end if**
15: **else if** $Action$ is `Thought` **then**
16: $s \leftarrow$ calculate_state($User\_Query, Action$)
17: If $s < \sigma$ Then $s \leftarrow \sigma$
18: Push $Action$ into $\mathcal{M}$
19: **else if** $Action$ is `Plan` **then**
20: Push $Action$ into $\mathcal{M}$
21: **else if** $Action$ is `Tool_Use` **then**
22: $observation \leftarrow$ call($tool\_type$) & search $\mathcal{D}$
23: Push $observation$ into $\mathcal{M}$
24: **else if** $Action$ is `Backtrack` **then**
25: old_action = Pop from $\mathcal{M}$
26: If old_action is `Thought` then reset s to the previous state
27: **else if** $Action$ is `Summary` **then**
28: Pop from $\mathcal{M}$
29: Push $Action$ into $\mathcal{M}$
30: **end if**
31: $actions\_taken \leftarrow actions\_taken + 1$
32: **end while**
33: **Return** $Conclusion \leftarrow$ top($\mathcal{M}$)

TC–RAG Prompt

Answer the following questions as best you can. You have access to the following tools:

[Insert tool_descriptions here]
i.e. DOC_RAG: You can obtain medical knowledge from authoritative documents via this tool to help you reply.

Please think strictly according to the provided way of thinking without omission, and use the following format:

[Actions Pipeline Format]
`User_Query`: User's questions or observed information,
`Thought`: You should think about what to do, whether to answer questions based on the results of the tool or decide which tool to use.
`Tool_Use`: The tool to be used must be one of [tool_name], do not add any extra characters or symbols! Only the name of the tool can be output!
`Tool_Observation`: The answer provided by the tool (not generated by you)
`Plan`: Make decisions to outline subsequent actions or strategies.
`Summary`: When the previous action outputs a large amount of vocabulary and you need to summarize it, please output a detailed summary based on your knowledge.
`Backtrack`: When the result of the previous action is meaningless to your task and you need to re-execute it, apply this.
...(`Thought/Tool_Use/Summary/Backtrack` here can be repeated zero or more times)
`Thought`: I now know the final answer.
`Conclusion`: the final answer to the original input question.

[Start Conversation]
Begin!
`User_Query:`...

8.7 Detailed Experimental Setup

8.7.1 Dataset.

Our experiments are conducted on two open-source query sets: MMCU-Medical (Zeng, 2023) and CMB-Exam (Wang et al., 2023b) datasets, which are designed for multi-task Q&A and encompass single and multiple-choice questions in the medical field, and one open-domain Q&A dataset CMB-Clin (Wang et al., 2023b) which is the inaugural multi-round question-answering dataset based on real, complex medical diagnosis and treatment records. For MMCU-Medical, the questions are from the university medical professional examination, covering the three basic medical sciences, pharmacology, nursing, pathology, clinical medicine, infectious diseases, surgery, anatomy, etc., with a total of 2,819 questions. The CMB-Exam dataset utilizes qualifying exams as a data source in the four clinical medicine specialties of physicians, nurses, medical technicians, and pharmacists, with a total of 269,359 questions. The CMB-Clin dataset contains 74 high-quality, complex, and real patient cases with 208 medical questions.

8.7.2 RAG Tools.

We have involved several types of RAG tools:

(1) Knowledge Graph Search. *CMeKG* (Clinical Medicine Knowledge Graph)[2][3] (BYAMBASUREN et al., 2020), *CPubMed-KG* (Large-scale Chinese Open Medical Knowledge Graph) [4] and *Disease-KG* (Chinese disease Knowledge Graph)[5] are open-source medical KGs, which integrates extensive medical text data, including diseases, medications, symptoms and diagnostic treatment technologies. The fused KG has 1,288,721 entities and 3,569,427 relations. However, due to the lack of medical entity descriptions in its entities, we collect relevant entity knowledge from Wikipedia[6], Baidu Baike[7], and Medical Baike[8], and store them as entity descriptions. As mentioned by HyKGE (Jiang et al., 2024b) and GraphRAG (Edge et al., 2024), we choose the reasoning chains and knowledge communities as knowledge carriers.

(2) Documents Search. We have collected over 2 million medical documents of 3B size from drug instructions, medical textbooks, medical encyclopedias, clinical diagnosis and treatment guidelines, medical papers, and medical electronic medical records. To be specific, we utilize the GTE embedding model (Li et al., 2023c) "gte_sentence-embedding"[9] to obtain the embedding for each document, which is currently the top-performing model for text vector embedding in the retrieval field. We set the document chunk size to 128 with overlap size 50.

(3) Web and Pedia Search. In order to support the implementation of online retrieval, we also support searching on encyclopedias such as Wikipedia and MedNet (for this purpose, we pre-trained a W2NER model (Li et al., 2021) for medicine to extract medical entities). In addition, we also use search engines such as Bing and Google for web retrieval.

(4) Electronic Medical Record Database Search. In order to support the retrieval of similar patient information, we also apply MIMIC-III (Johnson et al., 2016), MIMIC-IV (Johnson et al., 2023), and eICU (Pollard et al., 2018) datasets, following the code translation

[2]https://cmekg.pcl.ac.cn/
[3]https://github.com/king-yyf/CMeKG_tools
[4]https://cpubmed.openi.org.cn/graph/wiki
[5]https://github.com/nuolade/disease-kb
[6]https://www.wikipedia.org/
[7]https://baike.baidu.com/
[8]https://www.yixue.com/
[9]https://www.modelscope.cn/models/damo/nlp_gte_sentence-embedding

"ICD9-CM" (Ma et al., 2023b; Xu et al., 2024) principle to support the retrieval of similar patients.

8.7.3 LLM Turbo.

To fairly verify whether TC–RAG can effectively enhance LLMs, we selected the general-domain large model and the medical-domain large model as the base models and explored the gains brought by TC–RAG: Qwen-1.5-32B-chat. In the general domain, we select Llama3.2-1B-Instruct, Llama3.1-8B-Instruct, Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct for comparison.å

8.7.4 Compared Methods.

In order to explore the advantages of the TC–RAG, we compare the TC–RAG results against twelve other models: (1) **Base Model (Base)** servers as the model without any external knowledge, used to check the improvement effect of different RAG methods. We use Qwen and pre-trained ones as base models. (2) **CHAIN-OF-THOUGHT (CoT) (Wei et al., 2023)** generates a series of intermediate reasoning steps to perform complex reasoning. (3) **Single Round-RAG (SR-RAG)** is selected with the combination of KGRAG (Soman et al., 2023b,a; Sen et al., 2023), embedding-based Document RAG and web search based on the initial question. (4) **Fix Length RAG (FL-RAG) (Khandelwal et al., 2020; Borgeaud et al., 2022; Ram et al., 2023)** triggers the retrieval module every n tokens and the tokens generated in the previous token window are utilized as the query. (5) **Fix Sentence RAG (FS-RAG) (Trivedi et al., 2023)**: Similar to FL-RAG, we retrieves based on every sentence. (6) **CHAIN-OF-NOTE (CoK) (Li et al., 2023b)** generates sequential thoughts after retrieved knowledge, enabling a thorough evaluation of their relevance to the given question and integrating these thoughts to formulate the final answer. (7) **Summarizing Retrievals (SuRe) (Kim et al., 2024)** constructs summaries of the retrieved passages for each of the multiple answer candidates and confirms the most plausible answer from the candidate set by evaluating the validity and ranking of the generated summaries. (8) **Hypothesis Knowledge Graph Enhanced Framework (HyKGE) (Jiang et al., 2024b)** leverages the hypothesis output and knowledge graph to enhance model inference. (9) **Reasoning And Acting (ReACT) (Yao et al., 2022a)** overcomes issues of hallucination and error propagation prevalent in chain-of-thought reasoning ahd actions. (10) **Self-Reflective RAG (Self-RAG) (Asai et al., 2024)** enhances an LM's quality and factuality through API retrieval and self-reflection. (11) **Forward-Looking Active RAG (FLARE) (Jiang et al., 2023c)** iteratively uses a prediction of the upcoming sentence to anticipate future content and retrieve relevant documents to regenerate the sentence if it contains low-confidence tokens. (12) **Dynamic Retrieval Augmented Generation based on the Information Needs (DRAGIN) (Su et al., 2024)** is designed to make decisions on when and what to retrieve based on real-time information needs. Note that we strictly follow the prompts for the baselines as stated.

8.7.5 Evaluation Metrics.

To evaluate the performance of LLMs on multi-task medical choice questions, we instruct the models to provide only the correct answer and adopt the widely-used metric, **Exact Match (EM)**, as recommended by prior work (Zhu et al., 2021; Karpukhin et al., 2020). An answer is deemed correct under the EM metric if its form exactly matches all the correct answers listed in the ground truth. The EM score is computed as follows:

$$EM = \frac{\text{Number of Correct Answers}}{\text{Total Number of Answers}} \times 100\%.$$

For open-domain medical Q&A tasks, we employ **ROUGE-R** (Xu, 2023; Jiang et al., 2024b) and **Bilingual Evaluation Understudy (BLEU)** to assess the quality of the LLMs' responses.

Specifically, for **BLEU**, **BLEU-1** is used to measure answer precision, and **BLEU-4** evaluates answer fluency by considering higher-order n-gram consistency. **BLEU** evaluates the similarity of generated responses to the ground truth using the following formula:

$$\text{BLEU-N} = BP \cdot \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right),$$

where p_n is the precision of n-grams, BP is the Brevity Penalty, calculated as:

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right), & \text{if } c \leq r \end{cases}.$$

Here c is the length of the generated response, and r is the length of the reference response.

ROUGE-R quantifies the recall of retrieved knowledge in the LLMs' responses, emphasizing their ability to comprehensively cover the information relevant to the query. For a generated response R and a reference G, ROUGE-R is computed as:

$$\text{ROUGE-R} = \frac{|R \cap G|}{|G|},$$

where $|R \cap G|$ denotes the number of overlapping n-grams between the generated response and the reference, and $|G|$ is the total number of n-grams in the reference.

8.7.6 Experimental Implementation.

In TC–RAG, $\sigma = 10$ for $cppl$ and 20 for uct. Moreover, for all the baselines and TC–RAG, we set the maximum number of returned tokens for LLMs to 500 and the temperature to 0.6. For a fair comparison, we apply the same W2NER, GTE (Li et al., 2023c) models for all baselines. Moreover, the parameters of W2NER are optimized with Adam optimizer (Kingma and Ba, 2015) with L_2 regularization and dropout on high-quality medical dataset (Zhang et al., 2022; Hongying et al., 2020), the learning rate is set to 1e-3, the hidden unit is set to

1024 and weight decay is 1e-4. Implementations are done using the PyTorch 1.9.0 framework (Paszke et al., 2019) in Python 3.9, on an Ubuntu server equipped with 8 A100 GPU and an Intel(R) Xeon(R) CPU.

8.8 Additional Experiments

8.8.1 Generalization Experiments (RQ1)

To evaluate the generalization ability of TC–RAG, we conducted additional experiments on two MultihopQA datasets 2WikiMultihopQA (Ho et al., 2020) and HotpotQA (Yang et al., 2018) that require multihop reasoning, a reading comprehension dataset IIRC (Ferguson et al., 2020) and a commonsense reasoning dataset StrategyQA (Geva et al., 2021) in the general domain. All experiments utilized a unified Wiki-based knowledge repository as the RAG tool. Specifically, we employed the following LLMs: Llama3.2-1B-Instruct, LLaMA3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct. For the two MultihopQA datasets and reading comprehension dataset, we use **Exact Match** at the answer level, along with token-level measurements of **F1-Score**, to assess the quality of responses. For the commonsense reasoning dataset, which consists of true-false questions, **EM** was employed as the primary evaluation metric.

8.8.2 Noise Poisoning Attack (RQ3)

To assess the robustness of TC–RAG against noise poisoning attacks, we conducted two distinct types of attacks: **Partial Attack and Structural Attack**. These attacks simulate scenarios involving excessive redundant information or situations where retrieval mechanisms fail, leaving no effective information. Additionally, we introduced two categories of noise—**Irrelevant Retrieval Noise & Relevant Retrieval Noise** (Yoran et al., 2024; Cuconasu et al., 2024; Fang et al., 2024; Zhang et al., 2024)—to evaluate the effectiveness of the `Backtrack` and `Summary` actions.

In terms of implementation, we artificially constructed 100 pieces of noise based on CMB-Exam for each type of noise. It is worth noting that for partial attacks, we directly concatenate the noise knowledge after the retrieved results. For structural attacks, we replace the retrieved knowledge instead. We conduct the poison attack during the first function call:

Impact of Structural Attack and Partial Attack In the context of **Structural Attack**, the influence of **Partial Attack** is particularly pronounced. Since this type of noise is directly embedded within the sentence, it exerts a greater interference on TC–RAG. The experimental results indicate that under Partial Attack, (1) TC–RAG is more likely to trigger the `Summary` action, attempting to condense and process the excess information. This suggests that the model tends to utilize summarization as a means to handle noise in such scenarios. (2) Nevertheless, TC–RAG still triggers the `Backtrack` action in some cases, though with a lower probability compared to **Structural Attack**. (3) More notably, due to the greater impact of this embedded noise, the model's EM score significantly decreases, indicating that **Partial Missing** noise has the most substantial impact on TC–RAG during **Structural Attack**.

Comparison of Noise Types (1) In contrast, the impact of **Relevant Noise** is even more severe, particularly in the context of **Partial Attack**. Since **Relevant Noise** is highly related to the task, TC–RAG struggles to determine whether the noise contains the required answer, leading to a significantly lower EM compared to when dealing with **Irrelevant Noise**. (2) **Relevant Noise** is more likely to trigger the `Summary` action, indicating that when faced with task-related noise, the model may prefer summarizing the information rather than directly identifying and discarding irrelevant content. (3) In contrast, **Irrelevant Noise** is more easily detected by TC–RAG and effectively removed through the `Backtrack` action. The model handles it more efficiently, with the `Backtrack` execution probability reaching as high as 94%. (4) The results for **Mixed Noise** fall between the two, but since it contains **Irrelevant Noise**, which is easier for the model to detect, its performance is closer to that of **Irrelevant Noise**.

Overall Robustness under Attack Overall, under the attacks, TC–RAG demonstrates strong robustness, with the execution probabilities of `Summary` and `Backtrack` remaining above 81%, and sometimes reaching as high as 95%. This clearly illustrates the effectiveness of TC–RAG in managing the memory stack, effectively preventing the introduction of erroneous knowledge and irrelevant information, thereby maintaining the purity of the memory stack, in line with ***C3***.

8.8.3 Case Study (RQ3)

In this case study, we examine the effectiveness of two RAG systems: one without memory and state management–ReACT, and TC–RAG. We evaluate how each system impacts the state management, retrieval, and reasoning process, particularly when dealing with irrelevant or incorrect noise.

The results indicate that the ReACT-based approach struggles with accumulating irrelevant noise, leading to overconfidence and inaccurate conclusions, simply due to the unit conversion in Figure 2. In contrast, TC–RAG effectively manages its memory and utilizes `Summary` & `Backtrack` actions to prune incorrect retrievals, resulting in more concise and accurate conclusions, which underscores TC–RAG's superiority in handling complex tasks (for ***C3***). Furthermore, we found that the ReACT-based approach tends to prematurely settle on answers when the system state value is high, due to the lack of state management. On the other hand, TC–RAG dynamically monitors the RAG process, ensuring that the system state value meets the termination condition, which highlights the necessity of constructing a system state, in line with ***C1***.

Table 4: The general experimental results of TC–RAG and other baselines on four benchmarks. The best results are in bold and the second runner are underlined.

		2WikiMultihopQA		HotpotQA		StrategyQA	IIRC	
LLM	**RAG Method**	**EM**	**F1**	**EM**	**F1**	**EM**	**EM**	**F1**
Llama3.2-1B-Instruct	**wo-RAG**	0.128	0.1958	0.067	0.1255	0.571	0.056	0.0728
	SR-RAG	0.138	0.2052	0.122	0.2004	0.582	**0.085**	**0.0922**
	FS-RAG	0.143	0.2094	0.120	0.1951	0.589	0.074	0.0891
	FL-RAG	0.134	0.1951	0.105	0.1892	0.548	0.072	0.0882
	FLARE	0.136	0.1883	0.091	0.1575	0.572	0.050	0.0621
	DRAGIN	0.126	0.1824	0.084	0.1334	0.570	0.051	0.0673
	TC–RAG	**0.153**	**0.2146**	**0.134**	**0.2164**	**0.591**	0.066	0.0804
Llama3.1-8B-Instruct	**wo-RAG**	0.260	0.3427	0.233	0.3223	0.747	0.190	0.2212
	SR-RAG	0.339	0.4394	0.246	0.3550	0.726	0.221	0.2811
	FS-RAG	0.370	0.4489	0.316	0.4298	0.745	0.218	0.2790
	FL-RAG	0.336	0.4250	0.253	0.3618	0.722	0.233	0.2917
	FLARE	0.286	0.3571	0.173	0.2710	0.745	0.197	0.2304
	DRAGIN	0.260	0.3390	0.223	0.3167	0.750	0.176	0.2168
	TC–RAG	**0.379**	**0.4654**	**0.340**	**0.4724**	**0.782**	**0.240**	**0.3132**
Qwen2.5-7B-Instruct	**wo-RAG**	0.141	0.2297	0.002	0.0662	0.583	0.068	0.1002
	SR-RAG	0.157	0.2646	0.012	0.1039	0.581	0.156	0.2084
	FS-RAG	0.222	0.3284	0.047	**0.1575**	0.678	0.158	0.2063
	FL-RAG	0.182	0.2856	0.021	0.1360	0.628	0.138	0.1836
	FLARE	0.126	0.2194	0.020	0.0970	0.618	0.095	0.1373
	DRAGIN	0.195	0.2817	**0.053**	0.1339	0.679	0.088	0.1241
	TC–RAG	**0.240**	**0.3478**	0.049	0.1533	**0.690**	**0.170**	**0.2307**
Qwen2.5-14B-Instruct	**wo-RAG**	0.037	0.1496	0.025	0.0595	**0.780**	0.048	0.0774
	SR-RAG	0.073	0.1985	0.016	0.1310	0.731	0.105	0.1440
	FS-RAG	0.148	0.2978	0.117	0.2412	0.777	**0.213**	0.2556
	FL-RAG	0.080	0.2347	0.040	0.1702	**0.780**	0.140	0.2580
	FLARE	0.254	0.3621	0.025	0.1208	0.721	0.071	0.1019
	DRAGIN	0.220	0.3231	0.100	0.2168	0.700	0.120	0.1624
	TC–RAG	**0.370**	**0.4397**	**0.180**	**0.2604**	**0.780**	0.200	**0.2915**

Table 5: Results for Structural and Partial Attacks

Noise Type	`Summary` **Prob.**	`Backtrack` **Prob.**	**EM**
Structural Attack			
Irrelevant Retrieval Noise	01.00%	94.00%	91.00
Relevant Retrieval Noise	12.00%	76.00%	84.00
Mixed Retrieval Noise	04.00%	91.00%	89.00
Partial Attack			
Irrelevant Retrieval Noise	37.00%	49.00%	85.00
Relevant Retrieval Noise	52.00%	29.00%	79.00
Mixed Retrieval Noise	41.00%	47.00%	81.00

8.8.4 In-Depth Analysis of the Trajectories of TC–RAG with Baselines

We expand the analysis with an additional case study on the user query:

User Query: "My CVP indicator is 9 cmH2O, is this normal?"

For this, we provide and compare the complete reasoning trajectories of multiple baselines, including Base, CoT, FLARE, and DRAGIN, in addition to TC–RAG.

Base: *Answer:* "normal" is True, but the CVP normal range is False.

Central Venous Pressure (CVP) is a measure of the pressure in the thoracic vena cava, near the right atrium of the heart. It is an important indicator of fluid status and right heart function. The normal range for CVP is typically between 2 to 6 cmH2O, although some sources may extend this range slightly to 0 to 8 cmH2O. A CVP of 9 cmH2O is slightly above the typical normal range.

FLARE: *Answer:* "normal" is True, the CVP normal range is True.

> Sentence 1: Central Venous Pressure (CVP) is a measure of the pressure in the thoracic vena cava, near the right atrium of the heart. (search not triggered)
>
> Sentence 2: It is an important indicator of fluid status and right heart function. The **normal range for CVP** is typically between 2 to 6 cmH2O. (search triggered)
>
> Search: normal range for CVP
>
> Observation: normal range of CVP is between 5-12 cmH2O.
>
> Sentence 2: It is an important indicator of fluid status and right heart function. The normal range for CVP is typically between 5 to 12 cmH2O. (search not triggered)

Table 6: Performance of Various Approaches on CMB Leaderboard

Method	Average Score	Medical Qualification	Nursing Exam	Pharmacist Exam	Medical Technician	Professional Knowledge	Medical Postgraduate
TC–RAG	**87.95**	**89.50**	**92.63**	**88.00**	**86.75**	**83.56**	**87.25**
HuatuoGPTII-34B	76.80	75.65	82.31	76.81	76.17	74.38	75.56
Qwen-72B-Chat	74.38	78.55	83.56	79.78	77.92	68.25	58.19
Yi-34B-Chat	69.17	71.10	77.56	73.16	73.67	66.56	52.94
Yi-6B-Chat	65.87	67.25	76.38	68.50	67.83	61.75	53.50
GPT-4	59.46	59.90	69.31	52.19	61.50	59.69	54.19
Qwen-14B-Chat	57.64	60.40	65.63	60.94	58.83	54.50	45.56
Baichuan2-13B-chat	39.88	40.04	45.65	40.60	39.25	39.25	34.45
ChatGLM2-6B	38.51	40.25	47.56	36.06	36.58	35.56	35.06
Baichuan-13B-chat	38.20	37.70	44.75	41.22	34.67	37.94	32.94
HuatuoGPT	29.49	29.90	34.00	29.06	30.92	27.38	25.69
ChatMed-Consult	20.23	19.40	21.69	20.00	22.83	18.88	18.56

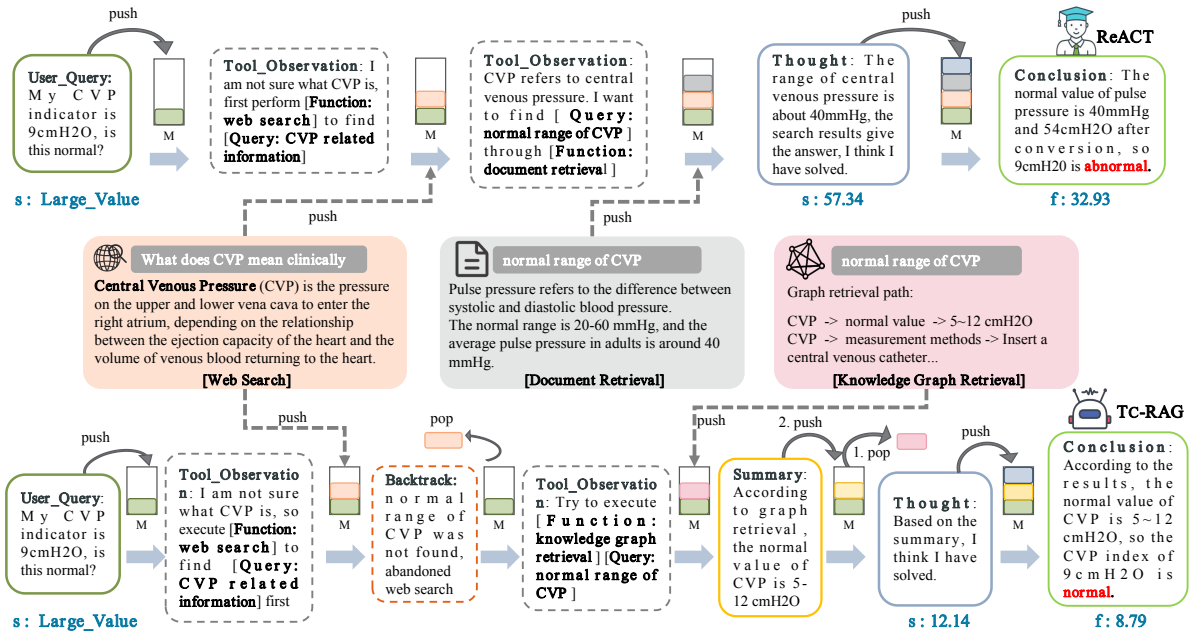

Figure 2: Case Study of ReACT-based and TC–RAG.

Sentence 3: Therefore, having a CVP of 9 cmH2O is normal. (search not triggered, ended)

DRAGIN: *Answer:* "normal" is True, the CVP normal range is True.

Sentence 1: Central Venous Pressure (CVP) is a measure of the pressure in the thoracic vena cava, near the right atrium of the heart. (search not triggered)

Sentence 2: It is an important indicator of fluid status and right heart function. The **normal range for CVP** is typically between 2 to 6 cmH2O. (search triggered)

Search: normal range for Central Venous Pressure

Observation: normal range of CVP is between 5-12 cmH2O.

Sentence 2: Therefore, having a CVP of 9 cmH2O is normal. (search not triggered, ended)

ReACT: *Answer:* "normal" is False, the CVP normal range is False.

Step 1: Tool Observation: I am not sure what CVP is. First perform [Function: web search] to find [Query: CVP related information].

Step 2: Tool Observation: CVP refers to central venous pressure. I want to find [Query: normal range of CVP] through [Function: document retrieval].

Step 3: Thought: The range of central venous pressure is about 40mmHg, the search results give the answer, I think I have solved.

Step 4: Conclusion: The normal value of pulse pressure is 40mmHg and 54cmH2O after conversion, so 9cmH2O is abnormal.

TC–RAG : *Answer:* "normal" is True, the CVP normal range is True.

Step 1 – push: Tool Observation: I am not sure what CVP is, so execute [Function: web search] to find [Query: CVP related information] first.

Step 2 – pop: Backtrack: normal range of CVP was not found, abandoned web search.

Step 3 – push: Tool Observation: Try to execute [Function: knowledge graph retrieval] [Query: normal range of CVP].

Step 4 – pop & push: Summary: According to graph retrieval, the normal value of CVP is 5-12 cmH2O.

Step 5 – push: Thought: Based on the summary, I think I have solved.

Step 6 – push: Conclusion: According to the results, the normal value of CVP is 5-12 cmH2O, so the CVP index of 9cmH2O is normal.

The comparative results reveal that:

- The ReACT approach tends to accumulate irrelevant information across steps, leading to a noisy context, inflated confidence, and ultimately incorrect conclusions. In the original case, even a simple unit mismatch introduces confusion that remains uncorrected due to the lack of memory revision mechanisms.
- FLARE and DRAGIN demonstrate adaptive information retrieval capabilities, which reduces the probability of acquiring noisy information when LLM's internal knowledge suffices to solve the problem. However, if irrelevant or harmful information is inadvertently included in the memory, these models cannot eliminate the impact of such noise.
- In contrast, TC–RAG effectively leverages its stack-based memory, employing Summary and Backtrack actions to eliminate erroneous information and maintain a coherent and accurate reasoning trace, resulting in a more precise conclusion.

8.8.5 The Ability of Backtracking

1. Noise Prevention Through Pre-Insertion Reasoning TC–RAG is based on a simple yet practical assumption: if each piece of knowledge is carefully evaluated before being added to memory, the risk of accumulating noisy or irrelevant information deep in the stack is substantially reduced. This preemptive filtering ensures that only relevant and reliable information is retained, meaning any potential noise typically appears near the top of the stack. As a result, local backtracking is generally sufficient in practice. This assumption underlies our choice of a stack-based memory structure, which enables efficient $O(1)$ access to recent reasoning steps, thereby optimizing both accuracy and computational efficiency.

2. Maintaining Reasoning Consistency Through Sequential Memory Allowing arbitrary modifications to deep memory entries would disrupt the sequential and causal continuity of the reasoning process. As discussed in Section 5 (Lines 433–437), once information is committed to the memory stack, it becomes part of the model's reasoning trajectory, where each new memory is conditioned on the previous ones. Direct edits to earlier entries may introduce inconsistencies or logical contradictions, particularly in domains that demand high levels of traceability and safety. Therefore, TC–RAG deliberately restricts memory modifications to the top of the stack in order to preserve the integrity, coherence, and interpretability of the reasoning flow.

3. Iterative Backtracking as a Practical Fallback In cases where outdated or incorrect information is later found to reside deeper in the stack, TC–RAG supports progressive correction through (1) multiple successive backtracking steps, or (2) higher-level action, such as defining a root-cause tracing action, that internally performs a sequence of pop (> 1) operations. This mechanism enables the system to rewind its reasoning trajectory larger when necessary, without compromising the temporal structure of the memory or the continuity of the decision process.

Here, we do also add a real case example from our log in the CMB dataset in which pop action appears more than once.

Query (push): A patient with a history of type 2 diabetes for 8 years is taking Metformin 1000mg bid and Dapagliflozin 10mg qd. Recently, he has experienced worsening fatigue, frequent nocturia, and foot numbness. The eGFR is 58 ml/min. What's the problem?
Thought (push): The patient's symptoms may be related to diabetes. Let me check it.
Search (push): Symptoms of diabetes
Observation (push): Common symptoms of diabetes include frequent urination, excessive thirst, unexplained weight loss, fatigue... (omitted)
=====> **[Backtrack (pop):]** diabetes symptoms information is not helpful.
=====> **[Backtrack (pop):]** I should change my search question.
Thought (push): The medicine may be relevant to the patient's symptoms.

Search (push): metformin and dapagliflozin
Observation (push): ... When eGFR < 60ml/min, Dapagliflozin needs to be reduced to 5mg qd ... (omitted)
Thought (push): The eGFR of the patient is 58ml/min. I think I found the reason.
Final Answer (push): The patient's eGFR suggests possible renal injury, and the symptoms are due to excessive Dapagliflozin dosage, requiring dose reduction.

It can be observed that when an unrelated message resides deeper within the memory stack, the model will expunge irrelevant memories via sequential popping operations upon detection.

In summary, TC–RAG adopts a cautious and principled approach that balances efficiency, interpretability, and reasoning stability.

8.9 How Does the System Handle Instructions Failing or Formatting Errors

Similar to other multi-step reasoning agent frameworks such as ReAct (Yao et al., 2022a), LangChain, LangGraph, and AutoGPT (aut), the probabilistic nature of LLMs can occasionally lead to outputs that deviate from the specified format or fail to fully comply with instructions. We refer to the methods in (Yao et al., 2022a; aut) for handling exceptions:

1. When the LLM generates an output that does not strictly adhere to the required format, we first employ string manipulation techniques (e.g., regex matching, template-based extraction) to identify and isolate valid components within the response. This allows us to salvage usable information even when the overall structure is imperfect.

2. If parsing still fails after multiple attempts, we leverage a retry mechanism to improve the likelihood of generating a parsable response. The system uses a while loop and try-catch blocks to force the LLM to re-generate output in the required format. Only when it generates parsable output or reaches a predefined maximum limit can the system stop.

8.10 CMB Leaderboard

Here, we compared the specific testing results of our model on CMB with the publicly available rankings on CMB Leaderboard with open-source baselines in Table 6. The experimental results show that TC–RAG has surpassed baselines in all medical indicators.