Can Hallucinations Be Useful? Solving Multi-Hop Questions With SLMs By Chaining System-I/II Reasoning

Anonymous ACL submission

Abstract

Recently, there has been increased interest in Small Language Models (SLMs), which are fast, show good performance, and have lower hardware demands than large language models (LLMs). However, SLMs hallucinate more frequently than LLMs, impacting their ability to solve complex multi-step reasoning problems as early mistakes cascade to the final response. To address this, existing works *think-first* followed by iterative retrieval to reduce hallucination. We argue that the *think-first* strategy is not always necessary as we find that: (i) SLMs are often accurately confident in their initial answer and, (ii) *hallucinations can actually be beneficial* for honing in on the true answer. As such, we position our work as an *inversion* of this strategy, i.e., *answer first-reason later*. We propose a *cognitively-inspired* framework where the model is first allowed to quickly answer the question (System-I (zero-shot)) and then resorts to deeper thinking (System-II) based on evidence retrieved from a knowledge source using the initial hypothesis. By combining System-I and System-II style thinking, we show that our method can outperform prior work that takes the traditional think-first route on various multi-step question-answering benchmarks.

1 Introduction

Multi-hop (or step) question answering (MHQA) requires a model to chain multiple pieces (hops) of information to answer a question (Lan et al., 2023). For example, the question "*Who was the top goal scorer in the first season of the team that won back-to-back Champions League titles in the last 10 years?*" needs a model to (i) find out which teams won in the last 10 years, (ii) determine which team won back-to-back, and (iii) find the first season's top scorer. Large Language Models (LLMs) (Naveed et al., 2025) have demonstrated substantial potential in solving such questions owing to their strong *reasoning* abilities (Xu et al., 2025). However, deep reasoning is generally observed in truly large models such as GPT-5 (OpenAI, 2025) and K2 (Team et al., 2025). Developing such models is non-trivial, making their wider adoption difficult.

Recently, there has been an increasing interest in *small* language models (SLMs) (Wang et al., 2025), which offer attractive properties, e.g., lower hardware demands and reasonably good performance. However, SLMs can hallucinate (plausible but incorrect response) (Huang et al., 2025) more frequently than LLMs (Li et al., 2024b). This hurts performance as early mistakes might propagate through reasoning, leading to incorrect answers.

An obvious solution to mitigate hallucinations is to train SLMs with specialized data (Li et al., 2024a; Tian et al., 2024). However, as acquiring high-quality training data is non-trivial, research (Srivastava et al., 2025) has shifted to more efficient reasoning techniques. Test-Time Scaling (TTS) (Zhang et al., 2025a) is one such technique that has gained recent popularity. It involves allocating more compute during inference by promoting models to generate long chains-of-thought (CoT) (Wei et al., 2022) before answering. TTS is generally applied to *logic* problems, such as math or coding (Tanay Varshney, 2025), which *do not require external information* and can be solved by a model using only its parametric knowledge.

Although TTS has shown promise for logic problems, recent work (Zhao et al., 2025) suggests that it is infeasible for knowledge-heavy tasks and can even exacerbate hallucinations. This is because a model cannot reason its way into an answer if it fundamentally lacks background information on a topic. To alleviate this, they are coupled with external knowledge via retrieval-augmented generation (RAG) (Gao et al., 2023). In this work, we view advanced RAG, i.e., beyond one-shot retrieve-then-generate, as a form of `knowledge-based TTS` (KB-TTS) where scaling happens across reasoning tokens (depth) and a knowledge source (breadth).

Most KB-TTS approaches (Li et al., 2025b)

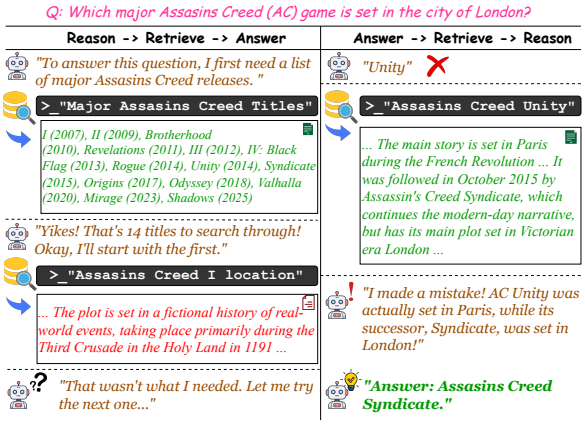

Figure 1: Current methods alternate between reasoning and retrieval, leading to the model getting flooded with unnecessary information, which ultimately misses the answer. Our method asks a model to give a fast, but might be wrong answer, which serves as an anchor to locate evidence for/against it, leading to the true answer.

follow a *plan-retrieve-reason-answer* cycle for MHQA, i.e., iteratively decomposing a question, retrieving background information, verifying the correctness of each step, and finally generating the answer. This is done to prevent a model from hallucinating as much as possible by breaking down a question into simpler and more manageable sub-questions. While this works, we argue that it is not always necessary for three reasons: (i) In cases where the model is accurately confident in itself. For example, *without thinking*, an SLM can accurately answer questions like "*Which continent is the Eiffel Tower located in?*". Iteratively solving questions like this is costly whereas an early stop would be favorable; (ii) Due to their size, SLMs struggle to reason at the level of LLMs as their breakdown of queries may be imperfect (Kim et al., 2025; Li et al., 2025c); (iii) When the model cannot find the needed information, it spirals into a chain of retrievals that continue to grow its context window (c.f. Table 2), finally resulting in noisy contexts and wrong answer. For example, in Fig. 1 (left), the SLM correctly identifies the search target (c.f. App. B for empirical validation). However, if the queries do not return anything useful, the model will keep exploring till either (i) it exhausts its context window or (ii) it exceeds the maximum requests allowed by the framework.

To address this, we suggest an *inversion* of the paradigm, i.e., instead of think-first, we propose *answer-first-reason-later*, framing our approach as cognitively inspired System-I (zero-shot: fast/instinctive) and System-II (post evidence gathering: slow/deliberate) type thinking (Kahneman, 2011). We observe this with many human experiences. For example, when a physician examines a complicated case, they may first offer a *gut-feel* hypothesis (System-I) based on the immediate information. However, to confirm it (System-II), they perform tests, study patient history and consider plausible alternatives (Rotgans, 2015). This dual-thinking model informs the design of our approach.

To this end, given a question, we first ask an SLM to answer quickly, without thinking. We notice that even if the answer is wrong, there exists some element of truth to its rationale. For instance, in Fig. 1 (right), with System-I thinking, an SLM might answer *Unity* (App. B). However, *Unity* is set in *Paris* (Wikipedia contributors, 2026), while its successor, *Syndicate*, is set in *London*. The model learns this by studying the initial answer's background. If we imagine the context as a graph, this mistake is plausible as the initial answer is a neighbor of the true answer. The observation here is that while the initial answer is incorrect, it is in the vicinity of the truth, leading us to argue that *hallucinations might not always be detrimental* and, on the contrary, can provide an initial clue leading to the correct answer. We use this insight for System-II, where we treat the initial answer as a *hypothesis* to initialize reasoning. Here, the SLM is first asked to generate knowledge-graph style triples (subject-predicate-object) from the initial answer and question (Fig. 2). These triples, containing entities and relationships relevant to the question (§3.2.1), are considered as a condensed summary of the SLM's reasoning and would be useful for retrieval. We use the triples as anchors to search a knowledge base for supporting or counterfactual contexts, if the initial answer is incorrect. The SLM studies all of the information, identifies its mistakes, and finally provides a refined answer.

Overall, our **contributions** are: (i) We identify the utility of hallucinations as an effective starting point in searching for information related to a query; (ii) We challenge the existing paradigm of *reason-first* KB-TTS with a simple and unique *answer-first-reason-later* strategy; (iii) Results indicate the effectiveness of our approach against more sophisticated approaches that generate detailed reasoning traces and prune invalid trajectories.

2 Related Work

To ensure comprehensiveness, we discuss training-free and training-based approaches to MHQA.

SFT on Distilled LLM CoT: Generally, LLMs craft superior reasoning traces (CoT) than SLMs. Observing this, prior work (Kang et al., 2025; Bi et al., 2025a; Lee et al., 2025; Piao and Park, 2025) distills synthetic reasoning data from LLMs to train SLMs to mimic them through supervised fine-tuning (SFT). However, as pointed out by Gudibande et al. (2024), synthetic CoT makes SLMs mimic an LLM's *style*, but not their *factuality*. This means that a trained SLM CoT follows similar steps as an LLM to break down a problem, but its core content is factually incorrect.

Reinforcement Learning (RL): As shown by (Yang et al., 2025; Guo et al., 2025), SFT followed by RL builds good reasoning as it encourages creative exploration of a solution space. Indeed, this has shown promise in various MHQA tasks (Yu et al., 2026; Luo et al., 2025a; Song et al., 2025). A critical RL requirement is the design of proper reward functions to guide the learned policy (trained SLM). Unfortunately, none of the mentioned works employ rewards to check *what* is being retrieved and only rely on surface-level response format checks. Even when they do evaluate retrieval quality (Luo et al., 2026a), the checks are imperfect, leading to degraded system performance. Considering the importance of retrieved content in RAG (Wu et al., 2025), this class of methods appears sub-optimal for MHQA.

Test-Time Scaling: As an alternative to training, TTS explores techniques to improve model performance at inference time by having it *think* harder (generate reasoning tokens) before answering. Early attempts at TTS include Tree (Yao et al., 2023), Graph (Besta et al., 2024), and Forest (Bi et al., 2025b) - of Thoughts, all building on the idea of exploring multiple reasoning trajectories before converging on the most promising one. These methods were developed for math or logical reasoning tasks, i.e., not requiring external information. For non-math/logic tasks, exhaustive reasoning cannot bridge the gap between knowing and not-knowing information on a given entity (Zhao et al., 2025).

KB-TTS methods (Pan et al., 2025; Liu et al., 2026; Luo et al., 2025b; Jiao et al., 2026; Shi et al., 2026; Li et al., 2026) address the above limitation. Here, a retriever provides the necessary context for a query in line with the model's reasoning. These methods are typically *reason and retrieve*, i.e., iteratively break down a question into sub-questions and retrieve context to answer them. However, progressive question solving relies on strong decomposition abilities, which are not always reliable for SLMs (Kim et al., 2025; Li et al., 2025c).

We propose an inversion of this strategy, i.e., instead of reasoning first, we use an SLM's initial guess to guide evidence acquisition used to inform its final response. Our approach hinges on the observation that an SLM's initial answer, although potentially hallucinated, is often in the vicinity of the ground truth. We leverage this to zero in on the true answer, instead of exploring dead-end reasoning paths. To the best of our knowledge, we are the first to make use of hallucinations in the proposed manner. The only other related work we find is Hosseini et al. (2024), who **trains** a reward model based on an LLM's correct/incorrect trajectories to better inform its decision making. While our methods share the same spirit, they are ultimately orthogonal as we approach MHQA through a **training-free** framework.

3 Methodology

An overview of our method is shown in Figure 2. At a high-level: given a question, the SLM first generates a quick, initial answer, serving as its hypothesis (System-I). Using this, it produces a condensed reasoning trace in the form of knowledge-graph style triples, which are used as search queries to retrieve context that may support or refute the initial claim. Finally, the SLM deliberates over the initial answer, triples, and retrieved context to identify its mistakes and come up with the final correct answer (System-II).

3.1 System-I: Direct Answer

Our framework begins by simply having the SLM try its best to provide an answer to the question by leveraging its parameterized knowledge. Here, the model is deliberately instructed not to reason (**no CoT**) before answering. Formally, given a question Q, the SLM generates an initial answer A^* following prompt P_{Sys-1} (in Fig. 6) as

$$A^* = SLM(Q, P_{Sys-I}) \quad (1)$$

The goal in this step is to harness the SLMs *parametric knowledge* to guide further evidence acquisition. As noted previously, an SLM is often capable of either producing a correct or near-correct answer to a question. We use this insight here to narrow the search space of a question instead of having it reason at the onset, which has a potentially higher degree of being error-prone due to their weaker question-decomposition abilities (§2).

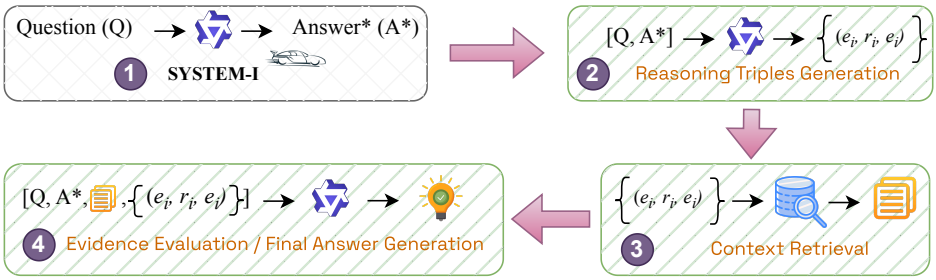

Figure 2: Method Overview. Step 1 (gray provides System-I answer). Steps 2-4 (green) involve System-II operations.

3.2 System-II: Reasoning With Evidence

System-II requires a more deliberate and well-rationed thought process to answer a question. As such, the **bulk of the reasoning budget is spent here**. This phase consists of three steps: (i) *Reasoning Triple Generation*: Given the initial answer, the SLM explains, using knowledge-graph style triples how it arrived at the answer; (ii) *Evidence Collection*: The reasoning triples are used as anchors to locate context relevant to the question from a knowledge base; (iii) *Final Answer*: The SLM reviews everything, investigates if/where there are flaws are and forms its final judgment.

3.2.1 Reasoning Triple Generation

System-II starts with the SLM generating a set of *reasoning triples*. Given the question (Q) and its System-I answer (A^*), the model is asked to explain how it arrived at the answer in the form of (subject-predicate-object) triplets using the prompt P_{triple_gen} (in Fig. 7), which can be written as

$$\{(e_i, r_i, e_i)\} = SLM(Q, A^*, P_{triple_gen}) \quad (2)$$

where e_i denotes entity and, r_i the relationship between the two entities. This is done for two reasons: (i) when explaining multi-hop logic, we view triples as a form of condensed reasoning that is easier for the SLM to handle than a verbose explanation; and (ii) each triple acts as a search key for locating a particular slice of information relevant to the question (§3.2.2).

We realize that the triples might contain hallucinated entities, as the model is expected to form connections using only its parametric knowledge. For example, given the question *What league does Luka Modric play in?*, an SLM might answer *La Liga* by yielding triples as, *Modric* | *plays_for* | *AC Milan* and *AC Milan* | *competes_in* | *La Liga* (correct - *Serie A*). Such mistakes are plausible due to a model's knowledge cutoff (Gao et al., 2025) or popularity biases (Luo et al., 2026b). Even though the second triple is incorrect, it contains relevant information, i.e., the concerned team, that can help a retriever perform *approximate neighbor matching* to locate related context. We reason that, as long as the key entities in the triples are related to the question, there is a high chance of acquiring the correct or, at least, connected information.

3.2.2 Context Retrieval

Each generated triple describes a particular relationship associated with a question. They act as search keys to locate context relevant to their highlighted relationship. For example, the triple *Modric* | *plays_for* | *AC Milan* can be used to retrieve chunks related to the player and his team. The retrieved context then serves as **grounding evidence** which may support or refute the System-I answer, depending on whether it was correct. This context is necessary for the SLM in the last step (§3.2.3) where it must consider all of the provided information to reach its final decision. Specifically, given a reasoning triple, generated in §3.2.1, a *Retriever* fetches the top-k relevant document chunks from the provided knowledge base as

$$D = Retriever(\{(e_i, r_i, e_i)\}) \quad (3)$$

where $D = \{d_i\}_{i=1}^{k}$ is the set of retrieved documents. Implementation-wise, the retriever performs simple semantic similarity (via approximate neighbor matching) between the search queries (triples) and the knowledge index. Although a triple is expressed in a pseudo-language (e.g., *plays_in*), a relatively strong retriever is capable of distilling the underlying meaning of the phrase. For each triple, we retrieve their top-5 document chunks and unify the results by removing repeated entries.

3.2.3 Final Generation

After all of the evidence has been gathered, the SLM is in a position to make an informed judgment about the question. Formally, the SLM utilizes

the question (Q), System-I answer (A^*), generated triples ($\{(e_i, r_i, e_i)^R\}$) to support that answer (§3.2.1) and the context retrieved (D) using the triples (§3.2.2) to form its final answer ($\mathcal{A}$) as

$$\mathcal{A} = SLM(Q, A^*, D, \{(e_i, r_i, e_i)\}, P_{Sys-II}) \quad (4)$$

where P_{Sys-II} is the prompt used (see Fig.8). We provide the initial answer and reasoning trace alongside the new evidence to the model to give it a bird's-eye view of all the data it needs to provide the final answer. With these, the SLM can (i) contrast retrieved context with its initial line of reasoning (triples) that led it to the System-I answer; (ii) identify potential flaws in its initial reasoning; (iii) update its rationale in light of the retrieved evidence, and (iv) provide a final System-II answer to the question. As reasoning is naturally critical to this step, we use a higher generation limit to promote the model to think more before answering.

4 Experiments

Through our experiments, we aim to answer three research questions (RQ) - (i) How well does our method perform against prior work? (ii) What is the overall impact of System-I v/s II answering? and (iii) What does a qualitative analysis of the framework reveal about the success and failure modes?

4.1 Experiment Setup

We first describe the datasets, models and baselines used in our work.

Datasets: We evaluate our framework on three widely used MHQA datasets, i.e., 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), and MuSiQue (Trivedi et al., 2022). The questions in these datasets require a model to assemble information from connected contexts to arrive at a solution. Following Jiao et al. (2026); Liu et al. (2026), we randomly sample 500 examples from each dataset's validation split for evaluation.

Index: To provide background for the questions, each dataset contains Wikipedia-sourced, abstract-style contexts, relevant or irrelevant to the question. Following prior work (Wang et al., 2026; Pan et al., 2025), we construct our knowledge retrieval index (§3.2.2) by combining all of these contexts, for each dataset. As the maximum context length ranges between ~900 and 2K tokens, we create standard (Tuychiev, 2026) 400 token chunks with a stride (context overlap) of 50 tokens for continuity.

Models: Following (most) prior work, we use `Qwen2.5-7B-Instruct` (Qwen et al., 2024) as the SLM for all our experiments. Qwen2.5 is not a native reasoning model, i.e., does not default to long-form thinking using `<think>` tags. Thus, it is ideal for isolating the strength of reasoning frameworks without the influence of pre-trained biases. For our retriever, we use a lightweight `Qwen3-0.6B` (Zhang et al., 2025b) embedding model, which offers high retrieval accuracy and low-compute requirements.

Baselines: As our framework does not involve post-training of the SLM, we compare it against similar methods. For competitiveness, we select the latest methods published near the time of writing this paper. Unfortunately, as many papers, including *several* prior published works, have not released code, we could not test them. That said, we tried our best to be fair by selecting the following categories of methods: (i) **One-shot retrieval** that performs context retrieval once based on the search query. This includes **Standard RAG** (Lewis et al., 2020), i.e., retrieving top-5 document chunks for a question, and **HyperGraphRAG** (Luo et al., 2025b) that uses *knowledge-hypergraphs*, i.e., graph structures where edges connect multiple vertices, as their knowledge index; (ii) **Iterative retrieval** that performs multiple rounds of retrieval based on the decomposed query. This includes **GraphAnchor** (Liu et al., 2026) which uses an *evolving* knowledge-graph; **PruneRAG** (Jiao et al., 2026) and **RT-RAG** (Reasoning Tree Guided RAG) (Shi et al., 2026), both of which explore a solution space by rejecting search-paths with low-quality answers; **PAGER** (Li et al., 2026) which uses *placeholder clues* generated by decomposing the question to inform the retrieval context; and **Search-o1** (Li et al., 2025a), which weaves search and an evolving knowledge-store to guide an agent. Further details are provided in App. C.

Metrics: We report standard MHQA metrics (App. F): (i) Exact Match (EM), 0/1 measure of exact string match; (ii) F1, harmonic mean of token overlap between the predicted and gold answer.

4.2 RQ1: Performance Against Prior Work

We present our main results in Table 1. As we can see, our proposed approach dominates existing *reason-then-retrieve* strategies. This reinforces our hypothesis that smaller models struggle to generate cogent plans (§2) and thus suffer in performance when subjected to such frameworks.

HyperGraphRAG and GraphAnchor are variations of GraphRAG (Peng et al., 2025), i.e., use knowledge graphs instead of documents as their in-

Method	Dataset							
	2WikiMultiHopQA		HotPotQA		Musique		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
Standard RAG	26.8	33.16	16.4	25.08	8.6	15.9	17.27	24.71
HyperGraphRAG∗	0.40	12.82	0.40	8.93	0.60	10.03	0.47	10.59
GraphAnchor†	22.4	28.23	22.2	29.44	9	14.62	17.87	24.1
PruneRAG	19.68	23.35	15.12	21.56	2.60	7.21	12.47	17.37
RT-RAG	15.40	22.05	9.40	14.89	6.20	12.33	10.33	16.42
PAGER	11.80	24.06	8.80	20.31	2.80	9.42	7.80	17.93
Search-o1	19.4	26.44	12.8	22.3	4.8	12.22	12.33	20.32
Ours (Proposed)	**29.60**	**34.71**	**20.80**	**28.75**	**10.20**	**19.42**	**20.20**	**27.63**

Table 1: Comparing frameworks. ∗Uses Gold Context only, †Unable to complete ~43% questions (App. C).

dex. The former's poor performance highlights major limitations with GraphRAG approaches (Peng et al., 2025), including the complexity of developing knowledge-graphs (App. C). Although they perform a one-time retrieval, they are ultimately let down by the complex nature of their retrieval structure (Table 2). GraphAnchor augments our text index with an evolving graph index, used for relational information. However, the complexity of parallelly decomposing questions, building graphs, and reasoning about contexts, overloads an SLM, preventing it from even being able to process all questions (App. C). In comparison, we use simple text chunks, similar to traditional RAG (Gao et al., 2023) and avoid complex operations such as weaving graph construction with question reasoning.

The remaining four methods utilize a text index and perform multiple rounds of reasoning and retrieval. Empirically, these approaches work better than the previous ones. However, they are quite complex, often requiring a model to verify its thought process by backtracking on faulty reasoning trajectories. As explained before, such requirements are difficult for smaller models to obey due to their limited instruction-following abilities. In fact, methods such as PAGER are seen to shine when models are scaled to 32B or 70B parameters. This is in contrast with our framework, which harnesses the strength of smaller models by avoiding heavy reasoning demands from them.

Finally, we notice how well Standard RAG performs against all methods. This further motivates the need for developing simpler frameworks that are more suitable for the capabilities of SLMs.

4.2.1 Information Efficiency

To understand the computational demands for each method, we measure the average number of tokens in the *retrieved contexts*. We do this to (i) gauge the core retrieval operation, i.e., the amount of grounding context needed for a framework to perform as designed, and (ii) strip out the variation due to prompt design, which can be succinct or verbose, and output token limits that can be controlled as a hyperparameter.

Framework	Avg. Context Tokens (Payload)	Relative Overhead	EM/F1
HGRAG	~17K	~13K%	0.47 / 10.59
Std. RAG	141.33	6.60%	17.27 / 24.71
Search-o1	141.32	6.54%	12.33 / 20.32
GA	135.98	2.52%	17.87 / 24.10
Ours	**132.64**	-	**20.20 / 27.63**

Table 2: Comparing average retrieved context (payload) tokens across all datasets. Relative Overhead indicates percentage increase in token consumption over our method. HGRAG (HyperGraphRAG), GA (GraphAnchor), Std. RAG (Standard RAG).

The results are shown in Table 2. As we can see, **the average payload (retrieved context) for our method is the lightest**, i.e., the least number of tokens. This is in stark contrast with HyperGraphRAG, which bloats context with hyperedges, leading to unmanageable API costs if used in production. Search-o1 and GraphAnchor also consume more tokens than ours, as both perform iterative retrieval across steps to answer questions. We are unable to measure PAGER, PruneRAG, and RT-RAG as PAGER has many questions that do not undergo retrieval, while the others do not provide a way to parse the collected chunks.

4.3 RQ2: Analyzing Framework Components

In Table 3, we present the results of our investigation of the different facets of our strategy. Here, we ask: (i) How well does System-I alone work? (ii) What happens if the retrieved context is irrelevant and the model has no fall-back option? (iii) Is there a way to gauge model *confidence* to balance System-I/II thinking?

Setting	2WikiMultiHopQA		HotpotQA		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
Full Framework	29.60	34.71	20.80	28.75	10.20	19.42	**20.20**	**27.63**
– System-II	25.40	29.73	14.20	21.35	2.60	10.03	14.07 $^{-30.3}$	20.37 $^{-26.3}$
– Initial Reasoning	23.00	30.63	13.00	21.82	7.20	16.32	14.40 $^{-28.7}$	22.92 $^{-17}$
+ Threshold	30.40	35.47	18.13	25.52	7.20	15.25	18.58 $^{-8}$	25.41 $^{-8}$

Table 3: Investigating impact of framework components (ablation). "– System-II" indicates only System-I answering, i.e., without involving any reasoning/retrieval. "– Initial Reasoning" components refer to answering without looking at the System-I answer and initial reasoning triples. "+ Threshold" uses confidence scores from the SLM to reduce the number of questions passing on to System-II processing.

4.3.1 Removing System-II

Firstly, we see that System-I alone (row 2), i.e., zero-shot, can provide reasonable accuracy. This makes sense, as even at 7B parameters, SLMs have sufficient parametric knowledge to at least make educated guesses. However, we do observe a drastic decline when compared with the full pipeline, highlighting the importance of grounding context to steer the model towards the correct answer.

4.3.2 Excluding Initial Reasoning

Next, we consider the impact of the System-I hypothesis and its associated reasoning (generated triples) on the final answer. When they are absent, the model does not have an answer to fall back on or understand where it went wrong when comparing the retrieved evidence with its generated trace (triples). This is reflected by reduced performance in row 3, highlighting their importance, where, if the retrieved context is inconclusive, the model can still rely on its initial hypothesis as an answer.

4.3.3 Balancing System-I v/s II

In addition to the main framework, we were interested to see if there is a way to *balance System-I/II*, i.e., decide which questions are complete, from System-I, and which need further processing (System-II). We do this for two reasons: (i) System-II processing of questions with already correct answers would be redundant, and (ii) noise in the retrieved context can sway the SLM from its initial answer, if it is already correct. To implement this, we utilize **model confidence**, i.e., answer logits. Confidence scores are calculated by a standard mean of log-softmax probabilities for the decoded answer tokens. Questions whose answer confidence is above an empirically determined threshold are marked as complete, with the remaining proceeding to System-II. Prior work (Mahaut et al., 2024) explains that logit scores are an imperfect measure of a model's confidence as they reflect token probability over the actual content of the output. However, we treat them as a *rough signal* to avoid the overhead of training a separate verification model (Zhou et al., 2025).

We notice that in general, model confidence for the majority of answers is quite low. This makes sense, as the questions require chaining interconnected pieces of information, and we are asking the model to answer without any form of thinking. Naturally, while it might know the answer, from its parametric knowledge, it is not confident in it. This is similar to how humans behave when asked a question that we potentially know the answer to, but lack the assertiveness in our judgment (Xie and Wang, 2026; Yang et al., 2019). Thus, we set our threshold, as empirically observed, to .05, which yielded the best performance overall. Under this value, we notice that the majority of questions move to System-II. However, around 200 questions per dataset were marked complete, indicating a sizable proportion (App. A). Row 4 shows the results for this experiment. As we can see, even a lightweight filter can deliver performance close to the full pipeline but by being more efficient (App. A). This is beneficial in settings where the compute budget needs to be managed carefully.

4.3.4 Quantifying System-I/II Advantage

Finally, we study the net overall effect System-II reasoning has over System-I. For this, we calculate the number of questions whose answer changes between the two modes, either for the better or worse. Results are shown in Table 4. We compute the number of questions whose answers are correct in either mode, going from being incorrect in System-I to correct in System-II (gain) and vice versa (loss). Correctness is measured using the same strict exact match metric used for the rest of our results. As such, the absolute numbers are generally low. Overall, we notice a **net positive impact of System-II**

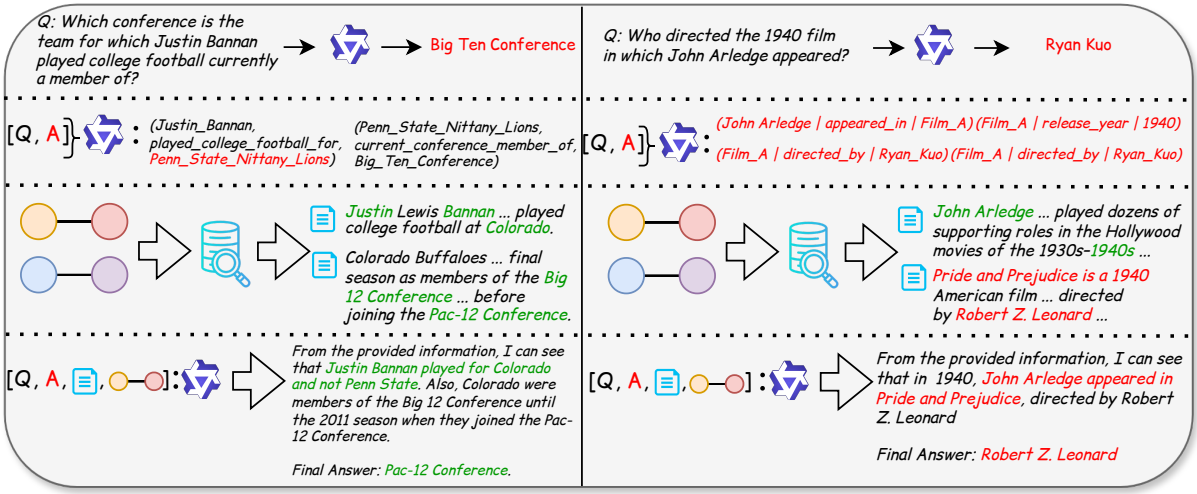

Figure 3: Success/failure (left/right) cases of our framework. Further examples contrasting with others in App. G.

reasoning with the proposed framework, especially for MuSiQue, which is claimed to be more difficult than the other two (Trivedi et al., 2022).

Dataset	Both Correct ($C \rightarrow C$)	Gained ($I \rightarrow C$)	Lost ($C \rightarrow I$)	**Net Gain** (Δ)
2Wiki	82	66	45	21
HTPT	49	55	22	33
MUS	9	42	4	38

Table 4: Answer changes between System-I v/s System-II processing. 2wiki = 2WikiMultiHopQA, HTPT = HotpotQA, MUS = MuSiQue, C/I = Correct/Incorrect. $\Delta = (Gain - Lost)$.

4.4 RQ3: Qualitative Analysis

Lastly, we perform a qualitative analysis of success and failure cases to understand the conditions under which the framework excels or struggles. We study two examples from HotpotQA, as shown in Fig. 3.

On the left, our system works as expected. The model produces an incorrect, but plausible answer. However, from its generated triples, we learn why it believes *Bannan* could be connected to the *Big Ten Conference*, as it incorrectly reasons that he played for *Penn State*. That said, these triples are useful anchors for locating counterfactual evidence. Using the first triple, the retriever finds an excerpt detailing where he played college football. Although the second triple does not contain direct links to *Colorado football*, it describes elements such as team names, conferences, etc. that help the retriever locate top-k relevant items to it, out of which is an excerpt detailing the transition of the team to the *Pac-12 conference*. By utilizing this counter-evidence for its initial claim, the model realizes its error and resolves to the correct answer.

The right side showcases an example where our framework struggles. When the model is *truly uncertain* in its initial answer, its reasoning trace gets reflected accordingly. Here, the SLM is unable to recall which *1940 film* the actor starred in. As such, it uses ineffective placeholder variables to justify its reasoning, which in turn leads to the wrong context being retrieved. Based on the retrieved context, the SLM reconciles the wrong *1940 film* with the actor, ultimately producing a flawed answer.

Overall, we conclude that when the model is reasonably confident in its initial answer, its subsequent reasoning will drive it towards the correct answer. However, in cases where it simply does not have enough knowledge to make an educated guess, it will try its best to generate a faithful reasoning chain, which can ultimately harm its ability to find the true answer. As such, efforts at detecting model confidence (§4.3.3) can make a difference here in building a more robust framework.

5 Conclusion

In this paper, we propose a cognitively inspired KB-TTS method, where we show that model *hallucinations can be beneficial* as an initial clue to locate the real answer. We show that, against prior work, which requires multiple rounds of reasoning/retrieval, our method recognizes the limited capabilities of SLMs but uses that to our advantage. However, as seen in Sec. 4.4, future work needs to balance cases where the SLM truly struggles with an initial answer with a more deliberate stepwise reasoning approach (Fig. 1) for a best-of-both-worlds setting.

Limitations

We identify two limitations with our work: (i) The datasets that we use, like all MHQA studies, are factoid-style (requiring short and objective answers) questions based on general-domain topics. As such, it remains to be seen how well our method works across more complex domains such as medical or finance. Future work will aim to address these spaces; and (ii) Our method does not perform iterative reasoning/retrieval, which, in cases such as Fig. 3, can be useful to mitigate incorrect reasoning trajectories. Although this does not impact the overall system performance, as observed empirically (Table 1), future work will explore how to trade off zero-shot retrieval with a multi-stage pipeline to achieve even more robust performance.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Jing Bi, Yuting Wu, Weiwei Xing, and Zhenjie Wei. 2025a. Enhancing the reasoning capabilities of small language models via solution guidance fine-tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9074–9084, Abu Dhabi, UAE. Association for Computational Linguistics.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025b. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*.

Xin Gao, Ruiyi Zhang, Daniel Du, Saurabh Mahindre, Sai Ashish Somayajula, and Pengtao Xie. 2025. Can prompts rewind time for LLMs? evaluating the effectiveness of prompted knowledge cutoffs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20777–20788, Suzhou, China. Association for Computational Linguistics.

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

Arnav Gudibande, Eric Wallace, Charlie Victor Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2024. The false promise of imitating proprietary language models. In *The Twelfth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

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.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Shuguang Jiao, Xinyu Xiao, Yunfan Wei, Shuhan Qi, Chengkai Huang, Quan Z. Sheng, and Lina Yao. 2026. Prunerag: Confidence-guided query decomposition trees for efficient retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 1923–1934, New York, NY, USA. Association for Computing Machinery.

Daniel Kahneman. 2011. Thinking, fast and slow. *Farrar, Straus and Giroux*.

Aditya K. Kamath, Ramya Prabhu, Jayashree Mohan, Simon Peter, Ramachandran Ramjee, and Ashish Panwar. 2025. Pod-attention: Unlocking full prefill-decode overlap for faster llm inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, page 897–912, New York, NY, USA. Association for Computing Machinery.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. 2025. Distilling LLM agent into small models with retrieval and code tools. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yujin Kim, Euiin Yi, Minu Kim, Se-Young Yun, and Taehyeon Kim. 2025. Guiding reasoning in small language models with llm assistance. *arXiv preprint arXiv:2504.09923*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Kyumin Lee, Minjin Jeon, Sanghwan Jang, and Hwanjo Yu. 2025. StepER: Step-wise knowledge distillation for enhancing reasoning ability in multi-step retrieval-augmented language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29501–29523, Suzhou, China. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Beibin Li, Yi Zhang, Sébastien Bubeck, Jeevan Pathuri, and Ishai Menache. 2024a. Small language models for application interactions: A case study. *CoRR*, abs/2405.20347.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024b. The dawn after the dark: An empirical study on factuality hallucination in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10879–10899, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China. Association for Computational Linguistics.

Xinze Li, Zhenghao Liu, Haidong Xin, Yukun Yan, Shuo Wang, Zheni Zeng, Sen Mei, Ge Yu, and Maosong Sun. 2026. Structured knowledge representation through contextual pages for retrieval-augmented generation. *arXiv preprint arXiv:2601.09402*.

Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, Chunkit Chan, Yankai Chen, Zhongfen Deng, Yinghui Li, Hai-Tao Zheng, Dongyuan Li, Renhe Jiang, Ming Zhang, Yangqiu Song, and Philip S. Yu. 2025b. A survey of RAG-reasoning systems in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12120–12145, Suzhou, China. Association for Computational Linguistics.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025c. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25366–25394, Vienna, Austria. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zhenghao Liu, Mingyan Wu, Xinze Li, Yukun Yan, Shuo Wang, Cheng Yang, Minghe Yu, Zheni Zeng, and Maosong Sun. 2026. Graph-anchored knowledge indexing for retrieval-augmented generation. *arXiv preprint arXiv:2601.16462*.

Haoran Luo, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, Luu Anh Tuan, and 1 others. 2025a. Graph-r1: Towards agentic graphrag framework via end-to-end reinforcement learning. *arXiv preprint arXiv:2507.21892*.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. 2025b. HypergraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Kangcheng Luo, Tinglang Wu, and Yansong Feng. 2026a. D ^{2} plan: Dual-agent dynamic global planning for complex retrieval-augmented reasoning. *arXiv preprint arXiv:2601.08282*.

Renqiang Luo, Dong Zhang, Yupeng Gao, Wen Shi, Mingliang Hou, Jiaying Liu, Zhe Wang, and Shuo Yu. 2026b. Bridging semantic understanding and popularity bias with llms. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 3656–3665, New York, NY, USA. Association for Computing Machinery.

Matéo Mahaut, Laura Aina, Paula Czarnowska, Momchil Hardalov, Thomas Müller, and Lluis Marquez. 2024. Factual confidence of LLMs: on reliability and robustness of current estimators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4554–4570, Bangkok, Thailand. Association for Computational Linguistics.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Trans. Intell. Syst. Technol.*, 16(5).

OpenAI. 2025. GPT5 system card. `https://cdn.openai.com/gpt-5-system-card.pdf`. [Accessed 10-12-2025].

Jianfeng Pan, Senyou Deng, and Shaomang Huang. 2025. CoAT: Chain-of-associated-thoughts framework for enhancing large language models reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13028–13045, Suzhou, China. Association for Computational Linguistics.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2025. Graph retrieval-augmented generation: A survey. *ACM Trans. Inf. Syst.*, 44(2).

Shengmin Piao and Sanghyun Park. 2025. TinyThinker: Distilling reasoning through coarse-to-fine knowledge internalization with self-reflection. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6069–6087, Albuquerque, New Mexico. Association for Computational Linguistics.

A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Jerome I Rotgans. 2015. It is time to progress beyond the system 1 versus system 2 dichotomy. *Perspectives on Medical Education*, 4(4):163–164.

Yuling Shi, Maolin Sun, Zijun Liu, Mo Yang, Yixiong Fang, Tianran Sun, and Xiaodong Gu. 2026. Reasoning in trees: Improving retrieval-augmented generation for multi-hop question answering. *arXiv preprint arXiv:2601.11255*.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V Vasilakos. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. 2025. ThinkSLM: Towards reasoning in small language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32600–32650, Suzhou, China. Association for Computational Linguistics.

Chris Alexiuk Tanay Varshney, Annie Surla. 2025. An Easy Introduction to LLM Reasoning, AI Agents, and Test Time Scaling | NVIDIA Technical Blog — developer.nvidia.com. https://developer.nvidia.com/blog/an-easy-introduction-to-llm-reasoning-ai-agents-and-test-time-scaling. [Accessed 12-03-2026].

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2024. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Bex Tuychiev. 2026. Best Chunking Strategies for RAG (and LLMs) in 2026 — firecrawl.dev. https://www.firecrawl.dev/blog/best-chunking-strategies-rag. [Accessed 23-05-2026].

Changjian Wang, Weihong Deng, Weili Guan, Quan Lu, and Ning Jiang. 2026. Cross-granularity hypergraph retrieval-augmented generation for multi-hop question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33368–33376.

Fali Wang, Minhua Lin, Yao Ma, Hui Liu, Qi He, Xianfeng Tang, Jiliang Tang, Jian Pei, and Suhang Wang. 2025. A survey on small language models in the era of large language models: Architecture, capabilities, and trustworthiness. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 6173–6183, New York, NY, USA. Association for Computing Machinery.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wikipedia contributors. 2026. Assassin's creed unity — Wikipedia, the free encyclopedia. [Online; accessed 24-May-2026].

Jinyang Wu, Shuai Zhang, Feihu Che, Mingkuan Feng, Pengpeng Shao, and Jianhua Tao. 2025. Pandora's box or aladdin's lamp: A comprehensive analysis revealing the role of RAG noise in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5019–5039, Vienna, Austria. Association for Computational Linguistics.

Chenwei Xie and William Shiyuan Wang. 2026. The tip-of-the-tongue phenomenon: Cognitive, neural, and neurochemical perspectives. *Biomedicines*, 14(2):269.

Fengli Xu, Qianyue Hao, Chenyang Shao, Zefang Zong, Yu Li, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Yuwei Yan, Qinglong Yang, Yiwen Song,

Sijian Ren, Xinyuan Hu, Jie Feng, Chen Gao, and Yong Li. 2025. Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns*, 6(10):101370.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Haiyang Yang, Ziv Carmon, Dan Ariely, and Michael I Norton. 2019. The feeling of not knowing it all. *Journal of Consumer Psychology*, 29(3):455–462.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Chuanyue Yu, Kuo Zhao, Yuhan Li, Heng Chang, Mingjian Feng, Xiangzhe Jiang, Yufei Sun, Jia Li, Yuzhi Zhang, Jianxin Li, and 1 others. 2026. Graphrag-r1: Graph retrieval-augmented generation with process-constrained reinforcement learning. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025a. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

James Xu Zhao, Bryan Hooi, and See-Kiong Ng. 2025. Test-time scaling in reasoning models is not effective for knowledge-intensive tasks yet. *CoRR*, abs/2509.06861.

Yefan Zhou, Austin Xu, Yilun Zhou, Janvijay Singh, Jiang Gui, and Shafiq Joty. 2025. Variation in verification: Understanding verification dynamics in large language models. In *First Workshop on Foundations of Reasoning in Language Models*.

A Balancing System-I/II Details

Table 5 provides the number of samples that were marked "complete" according to the model confidence scores, and those that were deemed necessary for further System-II processing. As we can see, around ~43% ((238+185+227)/1500) of the questions were marked complete, indicating a sizable portion of samples not requiring further processing, which could end up being unnecessary.

To estimate how much compute would be required to process the completed questions, we calculate the average number of input tokens for each of those samples. This includes the prompts along with the retrieved context. As we can see, for each dataset, we save a considerable amount of prefill compute (Kamath et al., 2025), i.e., the pre-decoding phase where the entire input prompt is processed. Thus, if the compute budget is constrained, we can still see reasonable performance (Table 3), but with a lower number of SLM calls.

Dataset	Complete	Incomplete	Avg. I/P Tokens Saved
2Wiki	238	262	389.66
HTPT	185	315	394.36
MUS	227	273	393.17

Table 5: Number of questions marked "complete" from System-I and those that required further processing in System-II (incomplete). Avg. Tokens Saved = Average input (I/P) tokens for each "complete" question, if they had to go to System-II. 2wiki = 2WikiMultiHopQA, HTPT = HotpotQA, MUS = MuSiQue.

B SLM Reasoning

To further lend credence to our *answer-then-reason* hypothesis, we provide empirical justification. For the example question in Figure 1, i.e., *Which major Assasins Creed (AC) game is set in the city of London?*, we ask Qwen-2.5-7B, the same model used in all our experiments, to (i) perform step-by-step reasoning, shown in Figure 4 and, (ii) give a direct, System-I style answer in Figure 5.

As we can see, the *reason-first* strategy indeed forces the model to search through all the games, to determine which title is set in *London*. In theory, this seems reasonable. However, this will lead to at least nine searches (based on the chronological order of the games) to reach *Syndicate*, causing major context bloat, which in turn, can lead to the model getting confused about which portion to pay attention to (Liu et al., 2024). On the other hand, our

> ***Reason First***
>
> *To solve this question, follow these steps:*
>
> 1. *****Identify the Assassin's Creed Games**:*** *List all the major Assassin's Creed games released so far.*
> 2. *****Research Each Game's Setting**:*** *For each game on your list, find out where the game takes place.*
> 3. *****Identify the Game Set in London**:*** *From the research done in step 2, identify which game is set in the city of London.*
> 4. *****Provide the Answer**:*** *Once you have identified the correct game, provide the answer to the question.*
>
> *Following these steps will lead you to the correct answer.*

Figure 4: Step-by-step reasoning for Fig. 1 question.

> ***Answer First***
>
> *The major Assassins Creed game set in the city of London is "Assassin's Creed IV: Black Flag" with a significant side mission set in London. However,* ***the primary game set in London is "Assassin's Creed: Unity," which was released in 2014***

Figure 5: System-I reasoning for Fig. 1 question.

answer-first strategy sees the model, quickly, albeit incorrectly, say *Unity*. We use this as a search key to locate counterfactual information and, in turn, find out from its context that its successor, i.e., *Syndicate*, is actually the title set in London. As such, our method can be considered as a *backtracking* framework where search can be done once we start with a reasonable, educated guess.

C Baseline Details

We provide further details on each baseline as follows,

1. **Standard RAG** (Lewis et al., 2020): This is the original RAG method that is used as a fixed baseline in any related study. The idea is simple: given a question, retrieve the top-k documents from a knowledge base that are relevant to it. These documents are included in the prompt for the model (LLM/SLM) as grounding context for the question. While this showed great promise initially in mitigating hallucinations and providing better answers, its performance was tested when subjected to more complex multi-step reasoning questions, as all of the necessary documents cannot be retrieved at once.

2. **HyperGraphRAG** (Luo et al., 2025b): Inspired by GraphRAG (Peng et al., 2025), this method constructs a *hypergraph*, i.e., a graph where edges connect multiple vertices. The idea behind doing this was to create a knowledge graph that broadly covers as much information as possible. However, in our experiments, we noticed that this method is incredibly inefficient. For instance, it is the only framework for which we had to use the gold contexts, as without it, index construction becomes infeasible. Even with the gold contexts, it takes 4 days to complete all three datasets, and without it, around 40 hours per dataset, which, if using an API-hosted model, incurs massive costs. This is confirmed by the authors: `https://github.com/LHRLAB/HyperGraphRAG/issues/14`.

3. **GraphAnchor** (Liu et al., 2026): Another variant of GraphRAG. Here, the framework uses an *evolving* knowledge-graph, i.e., at each step, a small, focused knowledge-graph, based on the retrieved documents, is developed, which keeps track of the entities and relationships in them. This graph is used to augment the retrieved documents and provide a relational understanding of the data. Unfortunately, the complexity of firing multiple follow-up questions, developing the graph, and keeping track of the reasoning chain, prevented the model from even completing all of the questions (43% across all datasets were incomplete). If we had used their evaluation script to report scores, each dataset would have close to ~2% F1, which would have heavily penalized their method. Thus, to be consistent with other methods, we use the official HuggingFace library to evaluate them, which reports appropriate numbers (App. F).

4. **PruneRAG** (Jiao et al., 2026): Grounded in *tree-based reasoning* (Yao et al., 2023), i.e., exploring multiple search spaces to determine an answer, PruneRAG performs three operations: (i) query decomposition and tree con-

struction, based on a model's reasoning using retrieved context; (ii) backtracking on the branches to aggregate evidence, (iii) token-confidence based answer rejection. While all of this provides decent performance, their method hinges on (i) question decomposition abilities of the model and (ii) token confidence, which, as explained (Mahaut et al., 2024), is not a reliable indicator of answer correctness.

5. **RT-RAG** (Reasoning Tree Guided RAG) (Shi et al., 2026): Similar to PruneRAG, RT-RAG also performs tree-style reasoning. It generates multiple candidate trees, selects the "optimal" one based on tree statistics such as depth and number of nodes, and, finally, gathers evidence based on the optimal tree to provide an answer.

6. **PAGER** (Li et al., 2026): This method first creates a "cognitive outline" (called "page") of what pieces of information a model needs to answer a question. The page has "slots" which are placeholders for potentially required data. These slots are progressively filled with iterative retrieval. Finally, once the entire page has been populated, the model provides its answer.

7. **Search-o1** (Li et al., 2025a): Inspired by `Agentic RAG` (Singh et al., 2025), i.e., letting an LLM-agent dictate what to retrieve instead of providing it beforehand with top-k documents. Search-o1 differs from the others in two ways: (i) performing *web* searches to retrieve background data, instead of using a pre-defined knowledge index, and (ii) including a module to refine the retrieved web pages to only provide the agent with the most relevant context for the question. We use the implementation provided by Jiao et al. (2026) as they modify it to use a local index rather than web searches.

D Prompts

Here we provide the prompts used in our framework in Figures 6, 7, 8.

E Code

Our codebase will be available upon publication.

Prompt For System-I Answering

```
You are a precise answering engine. Your task
is to provide the direct answer to a question
without any explanation.
### Rules:

  1. Provide ONLY the specific answer.

  2. Do not include introductory phrases (e.g.,
     "The answer is..."), explanations, or
     context.

  3. The answer must be wrapped in <answer>
     tags inside an <output> block.

  4. Output ONLY the <output> block.

### Example:
<input>
Question: What is the capital of France?
</input>

<output>
<answer>Paris</answer>
</output>

### Task: Process the following input and
provide the answer within an <output> block.
```

Figure 6: System-I answer prompt.

F Evaluation Details

All frameworks are evaluated using the HuggingFace implementation (`https://github.com/huggingface/evaluate`) of the metrics of the original work by Rajpurkar et al. (2018).

G Comparing RAG Methods

In Figures 9, 10, and 11, we contrast vanilla RAG and GraphAnchor with our method to show where each method struggles and where we excel.

Firstly, standard RAG pulls the top-5 most relevant documents for the query. Apart from the first document, which describes where *Justin Bannan* played, the others are irrelevant. As such, the model does not find what it is looking for, resorting to its parametric knowledge, as nowhere is it mentioned that *Colorado* played in the *Big 12 Conference*.

GraphAnchor, in this example, is surprisingly worse than standard RAG. First, it retrieves the same documents as standard RAG. It correctly identifies that *Justin Bannan* played for *Colorado*. However, it likely believes that *Colorado* has the *Carroll Fighting Saints* team, based on the generated graph, and thus states that he played in the *Frontier Conference*. This example highlights the limitations

Prompt For Triple Generation

```
You are a knowledge graph extractor. Your task
is to generate a detailed logical sequence of
subject-predicate-object triples that derive
a given Answer from a given Question.
### Rules:

  1. Format: Each triple must be enclosed
     in <triple> tags using the structure:
     <triple>Subject | predicate_link |
     Object</triple>.

  2. Logical Depth: Do not skip steps. If
     the question involves a specific role or
     relationship (e.g., "X's lead singer" or
     "Y's director"), you MUST identify that
     specific individual as a separate node
     before linking them to the final answer.

  3. Chain of Reasoning: The sequence must
     form a step-by-step path where the Object
     of one triple leads to the Subject of the
     next.

  4. Predicate Style: Use concise, lowercase,
     snake_case for predicates.

  5. Strict Output: Provide ONLY the <output>
     block. Do not include introductory text
     or explanations.

### Examples:
<input>
Question: Where was the lead singer of the
band Queen born?
Answer: Stone Town, Zanzibar
</input>
<output>
<triple>Queen | has_lead_singer | Freddie
Mercury</triple>
<triple>Freddie Mercury | born_in | Stone
Town, Zanzibar</triple>
</output>

<input>
Question: What is the birthplace of the person
who designed the Eiffel Tower?
Answer: Dijon, France
</input>
<output>
<triple>Eiffel Tower | designed_by | Gustave
Eiffel</triple>
<triple>Gustave Eiffel | born_in | Dijon,
France</triple>
</output>

### Task: Process the following input and
provide the triples within an <output> block.
```

Figure 7: Triple generation prompt.

Prompt For System-II Answering

```
You are a question answering assistant. You
are given a question, an initial guess,
supporting evidence for that guess (as
knowledge graph triples) and retrieved
context related to the question.

Think about everything step-by-step, by
considering all of the information and
determining where the flaws are.

Provide a clear, structured explanation of
your logic, and conclude by stating the final
answer clearly.

Always wrap your final answer inside
<final_answer> [answer] </final_answer> tags.
```

Figure 8: System-II answer prompt.

of GraphRAG approaches, where failure to understand relationships can lead to incorrect answers.

Finally, in our method, we observe a better coverage of information coming from two retrievals, based on the two generated triples. Using this, the SLM realizes that while *Colorado* initially played in the *Big 12 Conference*, they later joined *Pac-12*. This overrules the initial guess (*Big Ten Conference*), which turned out to be quite useful in generating the triples that led to the acquisition of the relevant documents.

H Hardware Requirements

As we are dealing with small language models, our hardware demands are relatively modest. Our cluster gives us access to NVIDIA A100 80 GB GPUs. However, we use only a *single GPU*, which loads both SLM and retriever models quite comfortably.

Q: Which conference is the team for which Justin Bannan played college football currently a member of?

Standard RAG

Documents:

1. Justin Lewis Bannan (born April 18, 1979) is a former American football defensive tackle. He was drafted by the Buffalo Bills in the fifth round of the 2002 NFL Draft. He played college football at Colorado.
2. Justin Harper (born February 24, 1985 in Catawba, North Carolina) is an American college football coach and former professional Canadian football wide receiver ...
3. Timothy Joseph "Tim" Landis (born July 13, 1964) is an American football coach who is currently quarterbacks coach and special teams coordinator at Lycoming College ...
4. Benjamin "Bennie" Oosterbaan (February 4, 1906 – October 25, 1990) ...
5. The Carroll Fighting Saint football program represents Carroll College of Helena, Montana in college football ...

. . .

To answer this question, let's break down the information provided:

1. *Justin Bannan played college football at Colorado.*
2. *We need to find out which conference the current team representing Colorado is a member of.*

From the given information, we can infer that the current team representing Colorado is part of the Big 12 Conference, as Colorado is a member of the Big 12 Conference.

Therefore, the answer is the Big 12 Conference.

<final_answer> Big 12 Conference </final_answer>

Figure 9: Demonstrating Standard RAG on an example question.

Q: Which conference is the team for which Justin Bannan played college football currently a member of?

GraphAnchor

Documents:

1. Justin Lewis Bannan (born April 18, 1979) is a former American football defensive tackle. He was drafted by the Buffalo Bills in the fifth round of the 2002 NFL Draft. He played college football at Colorado.
2. Justin Harper (born February 24, 1985 in Catawba, North Carolina) is an American college football coach and former professional Canadian football wide receiver ...
3. Timothy Joseph "Tim" Landis (born July 13, 1964) is an American football coach who is currently quarterbacks coach and special teams coordinator at Lycoming College ...
4. Benjamin "Bennie" Oosterbaan (February 4, 1906 – October 25, 1990) ...
5. The Carroll Fighting Saint football program represents Carroll College of Helena, Montana in college football ...

. . .

<think>
Based on the question, we need to identify the current conference of the team for which Justin Bannan played college football. From the given documents, we can find the following information:

- *Justin Bannan played college football at Colorado.*
- *The team for which Justin Bannan played college football is currently a member of the Frontier Conference.*

Therefore, we have sufficient information to answer the question.
</think>
<judgement> sufficient </judgement>
<graph>
Entities:

- *[Carroll Fighting Saints football] (Attributes: [current conference: Frontier Conference])*
- *[Justin Bannan] (Attributes: [played college football at: Colorado])*

Relationships:

1. *[Carroll Fighting Saints football] -> [current conference] -> [Frontier Conference]*
2. *[Justin Bannan] -> [played college football at] -> [Colorado]*

</graph>

<final_answer> Frontier Conference </final_answer>

Figure 10: Demonstrating GraphAnchor on an example question.

Q: Which conference is the team for which Justin Bannan played college football currently a member of?

Proposed Method

Initial Guess: Big Ten Conference

Initial Reasoning: (Justin_Bannan | played_college_football_for | Penn_State_Nittany_Lions), (Penn_State_Nittany_Lions | current_conference_member_of | Big_Ten_Conference)

Documents:

1. Justin Lewis Bannan (born April 18, 1979) is a former American football defensive tackle. He was drafted by the Buffalo Bills in the fifth round of the 2002 NFL Draft. He played college football at Colorado.
2. Benjamin "Bennie" Oosterbaan (February 4, 1906 – October 25, 1990) ...
3. Timothy Joseph "Tim" Landis ...
4. Justin Harper (born February 24, 1985 in Catawba, North Carolina) is ...
5. Judson Albert "Jud" Timm (August 28, 1906 – December 23, 1994) ...
6. The Trenton Lightning were an indoor ...
7. The 2013 Texas Longhorns football team ...
8. ... Colorado Buffaloes ... It was also the final season as members the Big 12 Conference in the North Division for Colorado, before joining the Pac-12 Conference for the 2011 season ...
9. Roy Randolph Roundtree ...
10. The 2015 Wisconsin Badgers football ...

. . .

Let's break down the information step-by-step:

1. *****Identify the Team**:** The question asks about the team for which Justin Bannan played college football. From the provided information, we know that Justin Bannan played college football for the Colorado Buffaloes.*
2. *****Determine the Current Conference**:** The question asks about the current conference of the team for which Justin Bannan played college football. From the provided information, we know that the Colorado Buffaloes were members of the Big 12 Conference until the 2011 season when they joined the Pac-12 Conference.*

Therefore, the current conference of the team for which Justin Bannan played college football is the Pac-12 Conference.

<final_answer> Pac-12 Conference </final_answer>

Figure 11: Demonstrating our method on an example question.