Sparse Latent Signals for Adaptive Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) improves factual question answering by leveraging external knowledge sources; however, retrieving documents for every query may introduce irrelevant context, which can in turn degrade the quality of the model's responses. Existing adaptive retrieval methods typically infer retrieval utility from uncertainty signals observed during autoregressive decoding, causing retrieval decisions to depend heavily on the generation process.

We propose SAE-RAG, an adaptive retrieval framework that leverages Sparse Autoencoders (SAEs) to extract retrieval-predictive latent signals during the query understanding stage, prior to final answer generation. SAE-RAG projects intermediate LLM representations into sparse latent features that correlate with retrieval-related semantic patterns and utilizes these signals across multiple stages of the retrieval pipeline. In particular, SAE-derived signals are incorporated into (1) Query-encoding-stage Retrieval Routing, (2) SAE-guided query construction, and (3) retrieval reranking. In addition, decomposition-aware retrieval and bridge-based knowledge integration facilitate multi-hop reasoning by selectively combining parametric knowledge with external evidence.

Experiments on four open-domain QA benchmarks demonstrate that SAE-RAG achieves improved question answering performance, particularly in multi-hop settings, while enabling more selective retrieval behavior than existing adaptive RAG baselines. Further analysis suggests that sparse latent features are associated with retrieval-related semantic patterns and provide useful signals for retrieval routing.

1 Introduction

Large language models (LLMs) have achieved strong performance across diverse NLP tasks, yet they continue to face challenges in knowledge-intensive question answering and factual generation

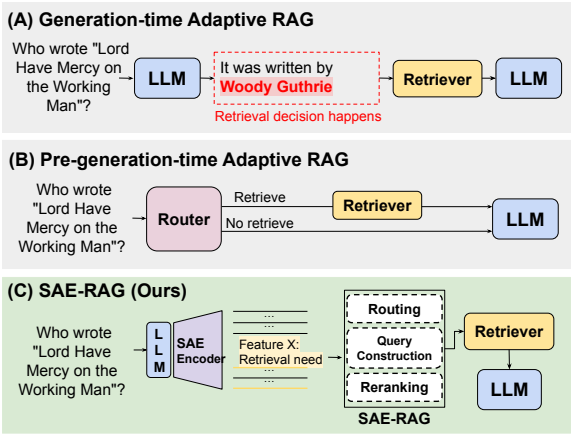

Figure 1: **Comparison between existing adaptive RAG approaches and SAE-RAG.** (A) Generation-time adaptive RAG performs retrieval during decoding based on uncertainty signals. (B) Pre-generation routing approaches predict retrieval utility before generation. (C) SAE-RAG leverages sparse latent signals from intermediate LLM activations for retrieval routing, query construction, and reranking.

(Lewis et al., 2020; Alansari and Luqman, 2026). This limitation largely arises from the difficulty of reliably encoding vast and continuously evolving factual knowledge within model parameters, particularly long-tail knowledge and information introduced after pretraining (Mallen et al., 2023).

Retrieval-Augmented Generation (RAG) mitigates these limitations by incorporating documents retrieved from external knowledge sources into the model input. However, performing retrieval for every query may introduce irrelevant context and even degrade generation quality when the required knowledge is already encoded in model parameters (Mallen et al., 2023; Shuster et al., 2021). Consequently, recent adaptive retrieval approaches have focused on selectively determining when external retrieval is required (Kadavath et al., 2022).

Many adaptive RAG methods rely on uncertainty signals arising during autoregressive decoding. For

example, FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024) dynamically trigger retrieval based on token probabilities, attention-based signals, and generation uncertainty observed during decoding. However, such approaches make retrieval decisions tightly coupled to the generation process, resulting in reactive retrieval behavior and limiting the ability to exploit retrieval-predictive signals during the query understanding stage.

Another limitation is that existing adaptive retrieval methods often rely on observable uncertainty proxies—such as question difficulty or generation-time confidence—rather than directly leveraging retrieval-related signals encoded in intermediate LLM representations. For instance, Adaptive-RAG (Jeong et al., 2024) trains a lightweight routing classifier based on estimated question difficulty, but such approaches provide limited insight into the internal representations associated with retrieval utility. As a result, identifying retrieval-related signals during query encoding remains challenging.

To address these limitations, we propose Sparse Autoencoder-guided Adaptive RAG (**SAE-RAG**), an adaptive retrieval framework that leverages Sparse Autoencoders (SAEs) to extract sparse latent representations from intermediate LLM activations during query encoding and reasoning planning. SAE-RAG transforms intermediate LLM representations into sparse latent features and utilizes them as retrieval-guidive latent signals during query encoding. Furthermore, these SAE-derived latent signals are utilized across multiple stages of the retrieval pipeline, including retrieval routing, query construction, and reranking. Specifically, SAE-derived signals are incorporated into (1) Query-encoding-stage Retrieval Routing, (2) SAE-guided query construction, and (3) retrieval reranking. This enables SAE-RAG to perform retrieval routing before retrieval-augmented final answer generation.

Experimental results on four open-domain QA benchmarks demonstrate that SAE-RAG achieves improved question-answering performance over existing adaptive RAG baselines, particularly in multi-hop QA settings. Further analysis shows that sparse latent features are associated with retrieval-related semantic patterns and are associated with retrieval-related semantic patterns and provide useful signals for retrieval routing in adaptive retrieval settings.

Our main contributions are summarized as follows:

- **Unified SAE-guided Adaptive Retrieval Framework**: We propose SAE-RAG, an adaptive RAG framework that leverages SAE-derived latent signals across multiple stages of the retrieval pipeline, including retrieval routing, query construction, and reranking.
- **Query-encoding-stage Retrieval Routing with Sparse Latent Features**: We demonstrate that sparse latent features extracted during query encoding can support retrieval routing prior to answer generation.
- **Analysis of Retrieval-related Sparse Features**: Through experiments and qualitative analyses, we show that SAE features are associated with retrieval-related semantic patterns and provide useful signals for adaptive retrieval in multi-hop QA settings.

2 Related Work

2.1 Adaptive Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) improves the factuality and knowledge coverage of large language models (LLMs) by combining parametric memory with external non-parametric memory (Lewis et al., 2020). However, retrieving external documents for every query may introduce noisy or conflicting evidence into the generation process (Shuster et al., 2021; Mallen et al., 2023). Consequently, recent studies have explored adaptive retrieval approaches that selectively determine when external retrieval is required.

Pre-generation routing approaches estimate retrieval utility using observable heuristics such as question difficulty or model confidence. For example, Jeong et al. (2024) employs a lightweight routing classifier that selectively determines whether retrieval should be performed based on estimated question difficulty. However, such approaches primarily rely on observable uncertainty proxies and provide limited insight into the internal representations associated with retrieval utility.

In contrast, generation-time retrieval approaches leverage uncertainty signals observed during autoregressive decoding. Jiang et al. (2023) proposed FLARE, which dynamically performs retrieval based on future-token prediction confidence during generation. Subsequently, Su et al. (2024) introduced DRAGIN, which determines retrieval timing using uncertainty, token influence, and attention-

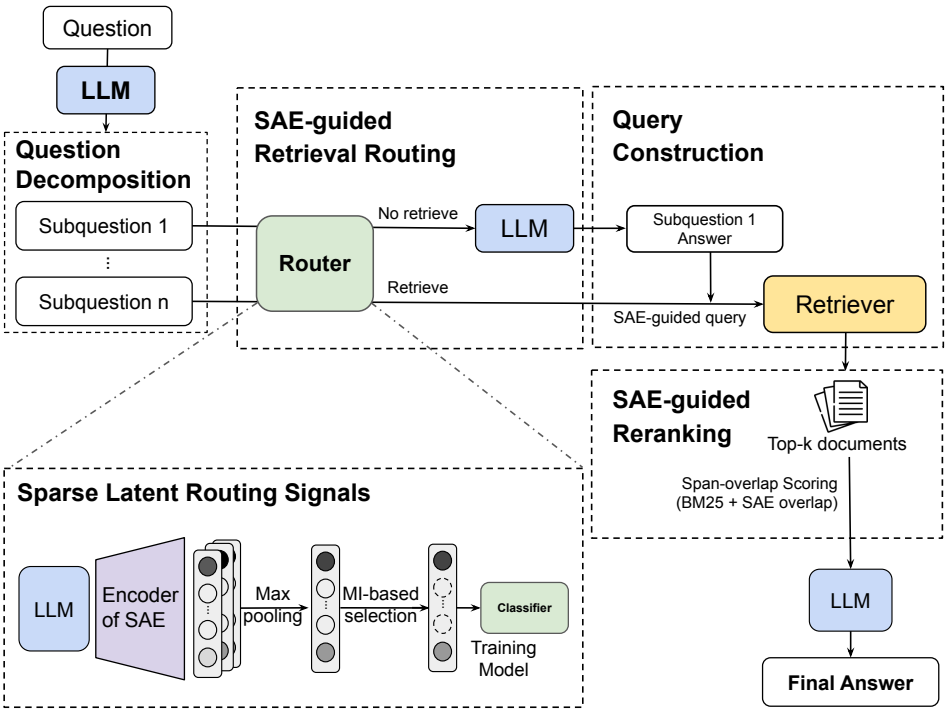

Figure 2: **Overview of SAE-RAG.** Given an input question, SAE-RAG first decomposes the question into sub-questions. For each sub-question, intermediate LLM representations are projected into sparse latent features using a pretrained SAE. The sparse latent signals are then used for: (1) retrieval routing to determine whether external retrieval is required, (2) Query construction for retrieval-oriented query generation, and (3) retrieval reranking based on salient sparse features. Retrieval-unnecessary sub-questions are answered using parametric knowledge only, while retrieved evidence and internally generated bridge knowledge are integrated for final answer generation.

based signals during decoding. Recently, Yao et al. (2025) proposed SEAKR, which extracts self-aware uncertainty from internal LLM hidden representations to dynamically determine retrieval timing and rerank retrieved knowledge, but still formulates retrieval as a generation-time uncertainty estimation problem based on consistency across multiple sampled generations. Although these methods improve retrieval adaptivity, retrieval decisions remain tightly coupled to autoregressive generation, limiting the ability to exploit retrieval-related signals prior to decoding. In contrast, SAE-RAG directly extracts sparse latent retrieval signals during the query encoding stage, enabling proactive retrieval routing prior to autoregressive decoding.

Unlike previous adaptive retrieval methods that primarily rely on observable uncertainty proxies or generation-time uncertainty estimation, our work focuses on directly extracting sparse latent retrieval signals from intermediate LLM activations during query encoding. To this end, we employ Sparse Autoencoders (SAEs) to enable retrieval routing before answer generation. Furthermore, SAE-RAG leverages sparse latent signals across multiple stages of the retrieval pipeline, including retrieval routing, query construction, and reranking.

2.2 Sparse AutoEncoder (SAE) in Retrieval-Augmented Generation (RAG)

Sparse Autoencoders (SAEs) are sparse representation learning methods that reconstruct internal LLM activations using sparse latent features, thereby decomposing entangled representations into sparse latent dimensions (Cunningham et al., 2023). Early SAE research primarily focused on interpretability and representation analysis, while recent studies have extended their applications to representation steering and activation-level analysis (Shu et al., 2025).

Recently, several studies have explored applying SAEs to Retrieval-Augmented Generation (RAG). Xin et al. (2025) utilized SAE features to analyze retrieval-related behaviors such as external context neglect and abstention behavior. Kang et al. (2025) proposed an interpretable retriever framework that decomposes document embeddings into sparse feature representations while maintaining retrieval quality. In addition, Xiong et al. (2025)

introduced RAGLens, which employs SAEs to detect hallucination and unfaithfulness signals during RAG generation.

However, existing SAE-based RAG studies primarily focus on interpretability, post-generation analysis, or representation-level behavior analysis. In contrast, our work leverages SAE-derived sparse latent features as retrieval control signals for adaptive retrieval decisions prior to decoding. Moreover, unlike prior work that primarily uses SAE features for analysis, SAE-RAG leverages sparse latent signals across multiple stages of the retrieval pipeline, including retrieval routing, query construction, and reranking.

3 Methods

3.1 Overview

Existing adaptive RAG methods determine retrieval utility from uncertainty signals during autoregressive decoding (Jiang et al., 2023; Su et al., 2024). Consequently, retrieval decisions depend on decoding-time signals, preventing proactive routing before generation. In this work, we propose SAE-RAG (Sparse Autoencoder-guided Adaptive RAG), an adaptive retrieval framework that predicts retrieval utility at the query encoding stage. We hypothesize that sparse latent activations encode retrieval-utility signals.

Prior SAE studies suggest that sparse latent representations capture more disentangled features than dense activations (Cunningham et al., 2023). Motivated by this observation, we investigate whether sparse latent features can support retrieval routing. To this end, we employ a pretrained SAE to extract sparse latent features from the intermediate activations of Large Language Models (LLMs). SAE-RAG leverages sparse latent signals across multiple retrieval stages, specifically incorporating (1) query-encoding-stage retrieval routing, (2) SAE-guided query construction, and (3) retrieval reranking.

Retrieval is executed only for sub-questions predicted to benefit from external evidence, while the remaining sub-questions rely solely on parametric knowledge. Intermediate responses are reused as bridge knowledge for subsequent reasoning.

The overall framework is illustrated in Figure 2.

3.2 Sparse Latent Retrieval Signals

To support compositional reasoning and fine-grained retrieval routing, an input question q is decomposed into a set of sub-questions using task-specific prompting:

$$\mathcal{S}(q) = sq_1, \ldots, sq_n \tag{1}$$

A Sparse Autoencoder (SAE) projects dense hidden representations into a high-dimensional sparse latent space, encouraging sparse activation patterns for a given input. In this work, we employ a pretrained SAE on the intermediate activations of the target LLM, and SAE parameters remain frozen during router training.

Hidden State Extraction We extract hidden activations from transformer layer l during the forward pass. Instead of utilizing the entire prompt representation, we selectively use only the hidden states corresponding to the user query span. Assuming the token span corresponding to the user message within the chat template is denoted as $[t_s, t_e)$, the query hidden representation is defined as follows:$H_q = h_t \mid t_s \leq t < t_e$ where $h_t \in R^{d_{\text{model}}}$ represents the hidden representation at token position t. This reduces interference from template tokens and better isolates question-specific representations.

SAE Projection. Each query token representation is projected into a sparse latent space via a pretrained SAE encoder:

$$z_t = \text{ReLU}(W_{enc} h_t + b_{enc}) \tag{2}$$

We retain only top-activated latent dimensions provided by the pretrained SAE. In this work, we employ an EleutherAI SAE with an expansion ratio of 32. Using an expansion ratio of 32, the SAE latent dimensionality becomes $dsae = 32 \times dmodel$

Query-level Sparse Feature Aggregation. The token-level sparse latent representations are aggregated into a single query-level sparse feature vector. For a given set of query token positions T, we apply max pooling over the token-level activations:

$$z_q[k] = \max_{t \in T} z_t[k] \tag{3}$$

where $z_q \in R^{d_{sae}}$ denotes the aggregated query-level sparse latent representation. After processing all samples, we obtain the following sparse feature matrix:

3.3 Query-encoding-stage Retrieval Routing

Our goal is to predict retrieval utility before autoregressive generation. During inference, retrieval utility is estimated independently for each sub-question. However, ground-truth retrieval-utility labels for the decomposed sub-questions are unavailable. We therefore construct routing supervision using single-hop factual QA data.

For each question q, we perform repeated temperature-sampled generation to produce a set of sampled responses $\mathcal{R}(q) = r_1, \ldots, r_N$ where N denotes the number of sampled generations, and $\hat{r}$ represents the majority response among the samples.

We calculate the response consistency as follows:

$$c(q) = \frac{\text{count}(\hat{r}, \mathcal{R}(q))}{N} \tag{4}$$

High-consistency correct predictions are labeled retrieval-unnecessary, while low-consistency incorrect predictions are labeled retrieval-beneficial:

$$\ell = \begin{cases} 0 & \text{if } c(q) > \tau_h \wedge \text{correct} \\ 1 & \text{if } c(q) < \tau_l \wedge \text{incorrect} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5}$$

Samples near the routing boundary are excluded from training to reduce supervision ambiguity. Self-consistency generation is used only for offline pseudo-label construction and is not required during inference.

MI-based Sparse Feature Selection. To reduce redundancy in the high-dimensional SAE latent space, we perform mutual-information-based feature selection.

$$Z \in R^{N \times d_{sae}} \tag{6}$$

We compute the mutual information between each latent dimension and the binary routing label. MI is computed using the empirical joint distribution:

$$\text{MI}(z_k, \ell) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{7}$$

We then select the top-k sparse latent features:

$$\tilde{z}q = \mathcal{F}MI(z_q) \tag{8}$$

In our default configuration, we select the top 1000 SAE latent dimensions for retrieval routing. The router predicts retrieval utility from the selected sparse features:

$$p(\ell = 1 \mid \tilde{z}q) = f\theta(\tilde{z}_q) \tag{9}$$

Among several classifiers, XGBoost achieved the best routing performance and was selected as the final router. We hypothesize that routing signals learned from single-hop QA generalize to decomposed multi-hop sub-questions.

3.4 SAE-guided Query Construction

SAE-RAG leverages sparse latent signals for both retrieval routing and retrieval-oriented query construction.Existing query generation methods rely on LLM-generated natural language queries, which may underemphasize retrieval-critical spans. We use sparse latent activations to identify retrieval-relevant tokens for query construction.

We define $\mathcal{F}_{ret}$ as the subset of sparse latent features highly correlated with retrieval utility. For each token, we compute a token-level attribution score, where s_f denotes the importance weight of feature f and $\phi_{t,f}$ the normalized activation magnitude at token position t:

$$\alpha_t = \sum_{f \in \mathcal{F}ret} s_f \cdot \phi t, f \tag{10}$$

Tokens with a high attribution score α_t are regarded as salient tokens associated with retrieval relevance.Top-scoring tokens are merged into retrieval-oriented salient spans. For a sub-question sq_j, the extracted salient spans are defined as $N_j = n_1, n_2, \ldots, n_m$.

The final retrieval query is formed by combining the LLM-generated query and the SAE-guided span query:

$$R_j = r_j^{LLM} \cup r_j^{SAE} \tag{11}$$

Here, r_j^{LLM} denotes the LLM-generated query and r_j^{SAE} is the lexical query derived from SAE-guided spans. This hybrid query construction combines semantic reformulation with sparse-latent-based lexical emphasis.

3.5 Selective Retrieval and Knowledge Integration

Retrieval is executed exclusively for sub-questions predicted to be retrieval-necessary:

$$\mathcal{D}j = \begin{cases} \emptyset & \ell_j = 0 \bigcup r \in R_j \text{Retriever}(r) \\ \ell_j = 1 \end{cases} \tag{12}$$

Here, $\mathcal{D}j$ denotes the set of retrieved documents for the sub-question sq_j. Sub-questions predicted as retrieval-unnecessary are answered using only parametric knowledge. Intermediate responses are reused as bridge knowledge for later reasoning steps.

To support multi-hop reasoning, sub-questions are processed sequentially. Intermediate answers are appended to subsequent prompts and used to interpret retrieved context. The retrieved documents are reranked using the SAE-guided salient span overlap:

$$s(d_i) = sretriever(d_i) + \lambda \cdot \text{overlap}(N_j, d_i) \quad (13)$$

where $s_{retriever}(d_i)$ is the base retriever score and $\text{overlap}(N_j, d_i)$ the normalized unigram overlap between salient spans and document d_i. This reranking favors passages aligned with salient lexical cues.

We define $\mathcal{I}$ as the bridge knowledge accumulated during the intermediate reasoning process, and $\mathcal{D}$ as the retrieval evidence collected from all sub-questions. The final response is generated by integrating retrieval evidence, bridge knowledge, and the original question:

$$\hat{a} = \text{LLM}(\mathcal{I} \oplus \mathcal{D} \oplus q) \quad (14)$$

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate SAE-RAG on four open-domain QA benchmarks covering both single-hop and multi-hop reasoning: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017), HotpotQA (Yang et al., 2018), and 2WikiMultiHopQA (Ho et al., 2020). Following prior RAG work, we use BM25-indexed Wikipedia as the retrieval corpus. For controlled comparison, all methods are evaluated on the same randomly sampled 1,000-question subset from each development split.

Baselines. We compare SAE-RAG against both retrieval-based QA systems and adaptive retrieval methods, including **No RAG**, **Single RAG**, FLARE (Jiang et al., 2023), and DRAGIN (Su et al., 2024). Unlike prior adaptive retrieval methods that rely on uncertainty signals during generation, SAE-RAG performs retrieval routing using SAE-derived sparse latent features extracted during query encoding. All methods use the same backbone LLM and retrieval environment for fair comparison.

Implementation Details. We use Llama-3.1-8B-Instruct as the backbone LLM and EleutherAI/sae-llama-3-8b-32x as the pretrained SAE. Retrieval is performed using BM25 over Wikipedia. Additional implementation details, including decoding configurations, routing-label construction, training settings, and retrieval hyperparameters, are provided in Appendix A.1.

Evaluation Metrics. We evaluate answer quality using Exact Match (EM) and F1. We additionally report retrieval-related statistics, including retrieval count (RC), generation count (GC), token count (TC), and sentence count (SC), to analyze retrieval behavior and computational characteristics. Retrieval routing performance is evaluated using AUC and F1.

4.2 Main Results

Table 1 presents the comparison between SAE-RAG and existing adaptive retrieval methods. Overall, SAE-RAG achieves competitive EM and F1 performance across both single-hop and multi-hop QA benchmarks.

In particular, SAE-RAG outperforms adaptive retrieval baselines such as FLARE and DRAGIN on multi-hop benchmarks including HotpotQA and 2WikiMultiHopQA. These results suggest that sparse latent retrieval routing is especially effective in multi-hop reasoning settings, where selective integration of retrieval evidence and parametric knowledge is crucial.

SAE-RAG also demonstrates competitive performance on single-hop datasets such as NQ and TriviaQA. Although SAE-RAG does not always minimize retrieval frequency, selectively leveraging retrieval based on sparse latent routing signals is more effective for downstream QA performance than performing retrieval for every sub-question indiscriminately.

4.3 Ablation Studies and Analysis

4.3.1 Routing Performance Analysis

Figure 3 illustrates the layer-wise performance of retrieval routing classifiers based on SAE features. Overall, XGBoost achieved the highest AUC and F1 scores among all classifiers, with the best performance observed in the middle transformer layers. Full experimental results are provided in Appendix A.4. Consistent with prior SAE studies (Gallifant et al., 2025; Ferrando et al., 2025), these results suggest that middle-layer SAE features provide more separable representations for retrieval

RAG Method	Single-hop QA												Multi-hop QA											
	NQ						TriviaQA						2WikiMultihopQA						HotpotQA					
	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓
No RAG	25.10	33.37	**0.00**	**1.00**	**13.43**	**1.62**	62.80	68.98	**0.00**	**1.00**	**12.46**	**1.85**	12.70	20.92	**0.00**	**1.00**	**22.28**	**1.83**	15.60	25.28	**0.00**	**1.00**	**10.55**	**1.29**
Single RAG	24.50	31.02	1.00	**1.00**	18.11	1.67	59.10	66.08	1.00	**1.00**	14.57	1.78	14.30	21.32	1.00	**1.00**	38.22	2.85	**25.50**	**35.36**	1.00	**1.00**	18.23	1.84
FLARE	26.30	33.98	1.14	3.72	46.02	5.21	62.70	68.96	0.29	1.62	22.01	2.84	9.10	14.55	1.69	3.44	98.28	7.11	16.50	24.00	1.59	3.34	61.45	5.37
DRAGIN	26.40	34.49	1.29	3.52	66.07	6.42	63.00	69.11	0.46	1.76	22.87	2.92	13.00	17.05	2.60	5.99	143.66	11.96	18.20	26.35	1.64	4.22	80.11	7.53
SAE-RAG (Ours)	**31.20**	**41.48**	1.07	2.91	137.4	12.03	**68.90**	**75.67**	0.39	2.91	122.8	12.02	**23.60**	**29.52**	2.31	3.05	168.56	13.95	24.20	33.15	1.62	3.25	165.3	12.62

Table 1: **Performance comparison across single-hop and multi-hop QA datasets.** SAE-RAG consistently achieves the best EM and F1 scores across diverse QA settings while adaptively controlling retrieval behavior.

RAG Method	EM ↑	F1 ↑
Full Retrieval	29.60	39.34
Dense Hidden-state Router	**33.50**	39.90
Sparse Latent Router (Ours)	31.20	**41.48**

Table 2: **Comparison of retrieval routing representations. The sparse latent router leverages SAE-based sparse features.**

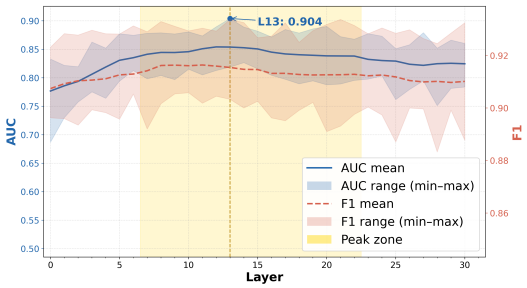

Figure 3: **Layer-wise retrieval routing performance distribution across classifier architectures using SAE latent features.** The x-axis denotes the transformer layer used for SAE feature extraction. The solid blue and dashed red lines represent the mean Test AUC and mean Test F1 scores, respectively, across four routing classifiers. The shaded regions indicate the min-max performance ranges across classifiers at each layer. The highlighted middle-layer region (Peak zone) corresponds to layers exhibiting consistently strong routing performance across classifiers. The best overall result was achieved by the Layer 13 XGBoost classifier (Test AUC = 0.904).

routing than early or late transformer layers.We adopt the Layer 13 XGBoost classifier, which achieved the best overall performance, as the final retrieval router.

4.3.2 Router Architecture Comparison

We compare SAE-based sparse latent routing with dense hidden-state routing under the same retrieval pipeline. As shown in Table 2, the *Full Retrieval* setting, which performs retrieval for every sub-question, achieved the lowest overall performance. While the Dense Hidden-state Router obtained the highest EM score, the proposed Sparse Latent Router achieved the highest F1 score. These results suggest that sparse latent features better isolate retrieval-relevant patterns than dense hidden representations.

4.3.3 Component Contribution Analysis

Table 3 shows the contribution of each SAE-RAG component. The Always Retrieve setting reduced EM on both NQ and 2WikiMHQA, suggesting that indiscriminate retrieval may introduce noisy or irrelevant context. Removing decomposition improved performance on NQ while reducing TC and SC, suggesting that decomposition may introduce unnecessary overhead for single-hop QA. In contrast, decomposition remained beneficial for multi-hop reasoning.

The results of w/o Bridge and w/o Rerank suggest that bridge knowledge integration and reranking primarily improve retrieval relevance and evidence ordering.

4.3.4 Threshold Analysis

Table 5 shows the trade-off between retrieval frequency and QA performance. The setting that performs retrieval for all questions ($threshold = 0.0$) exhibits the highest retrieval frequency but relatively lower EM and F1 scores. This suggests that indiscriminate retrieval does not necessarily improve QA performance.

As the routing threshold increases, retrieval becomes more selective, and EM/F1 scores gradually improve. The best performance is observed at $threshold = 0.9324$, which achieves the highest EM (31.00) and F1 (40.83). These findings support the effectiveness of selective retrieval based on sparse latent routing signals.

Interestingly, as the threshold increases, RC decreases while GC, TC, and SC consistently increase. This suggests that although retrieval is performed less frequently, the selected retrieval contexts at higher thresholds may encourage longer

Condition	NQ						2WikiMHQA					
	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓	EM ↑	F1 ↑	RC ↓	GC ↓	TC ↓	SC ↓
Full Model	31.20	**41.48**	1.07	2.91	137.4	12.03	23.60	**29.52**	2.31	3.05	168.56	13.95
Always Retrieve	29.60	39.34	2.34	2.16	124.15	10.75	19.10	23.69	3.33	2.28	139.60	12.24
w/o Decompose	**31.90**	41.39	**0.67**	**1.52**	**29.29**	**2.59**	22.00	27.11	**2.30**	**2.08**	**53.06**	**4.01**
w/o Bridge	30.10	40.46	1.15	2.82	148.92	12.02	**23.70**	29.17	2.31	3.05	168.24	13.97
w/o Rerank	30.50	40.43	1.15	2.83	148.82	12.03	21.40	27.01	2.31	3.07	168.85	13.99

Table 3: **Ablation study on NQ and 2WikiMHQA.** We analyze the contribution of decomposition, bridge retrieval, and reranking components to both QA performance and retrieval behavior.

Feature	Semantic Role	Sub-question	No Ret.	With Ret.	Gold
#48381	Competition / tournament entity	Who has the most *catches* in **NFL** history ?	Randy ✗ Moss	Jerry ✓ Rice	**Jerry Rice**
		When did **Michigan** last win a national championship in *basketball* ?	✗ 1992	✓ 1989	**1989**
		When did the **Eagles** win their last Super *Bowl* ?	✗ 2018	✓ 2017	**2017**
#104003	Geographic / nationality entity	When did the first *train* run in **England** ?	✗ 1825	✓ 1804	**1804**
		How many gold medals did **Australia** win in the 2000 *Olympics* ?	✗ 14	✓ 16	**16**
		What are the colors of the **Netherlands** flag ?	orange, ✗ white, blue	red, ✓ white, blue	**red, white, blue**

Table 4: Examples of retrieval-related SAE feature activations. Colors indicate activation intensity, and retrieval improves answer correctness.

Threshold	Unknown%	Retrieval%	EM	F1	RC	GC	TC	SC
0.0	100%	100%	29.60	39.34	2.34	**2.16**	**124.15**	**10.75**
0.3	89.5%	93.0%	30.60	40.08	2.89	2.29	128.77	11.00
0.5	80.5%	86.8%	30.60	40.23	2.66	2.39	133.77	11.24
0.9	50.7%	59.5%	30.80	40.82	1.71	2.76	146.67	11.92
0.9324 (optimal)	45.3%	53.8%	**31.20**	**41.48**	1.07	2.91	137.4	12.03
1.0	0%	0%	30.10	40.41	**0.00**	3.36	167.69	12.94

Table 5: Performance comparison across different retrieval thresholds.

query Tyle	EM	F1	RC	GC	TC	SC
LLM-based	28.60	41.65	1.15	**2.82**	148.8	12.02
SAE-based	28.40	41.17	**1.10**	2.88	**136.1**	**11.97**
Hybrid	**28.90**	**41.67**	1.54	**2.82**	148.9	12.04

Table 6: Comparison of retrieval and QA performance under different query construction strategies.

reasoning chains and richer contextual integration.

4.3.5 Query Construction Analysis

Table 6 compares different query construction strategies. The Hybrid setting achieved the best overall EM/F1 performance, suggesting that combining LLM-generated and SAE-guided queries improves retrieval effectiveness.

4.3.6 Analysis of Sparse Latent Retrieval Features

Table 4 presents qualitative examples of highly activated SAE features used in the retrieval router.

Feature #48381 was strongly activated for sports-related terms, while Feature #104003 was associated with country- and geography-related expressions. Activations were often concentrated around entity-centric and domain-specific terms. We additionally observed cases where retrieval corrected incorrect closed-book predictions, supporting the effectiveness of sparse latent retrieval routing.

5 Conclusions

We proposed SAE-RAG, a framework that leverages sparse latent features for adaptive retrieval. Experimental results on open-domain QA benchmarks demonstrate competitive performance, particularly in multi-hop settings. Our analyses suggest that sparse latent representations provide useful routing signals for selective retrieval. These findings highlight the potential of sparse latent representations for adaptive retrieval systems.

Limitations

This work has several limitations.

First, our experiments are limited to LLaMA-family models due to the availability of publicly released pretrained Sparse Autoencoders (SAEs). Therefore, further investigation is required to determine whether the proposed SAE-based retrieval routing approach generalizes effectively to other open-source LLM architectures.

Second, the current retrieval module relies solely on BM25-based lexical retrieval. Since BM25 depends heavily on exact lexical matching, it may not sufficiently capture semantic similarity between queries and documents. Future work should investigate whether SAE-based retrieval routing remains effective in dense retrieval or hybrid retrieval settings.

Third, the retrieval routing classifier is trained using pseudo-labels generated from self-consistency heuristics rather than manually annotated retrieval utility labels. Consequently, the quality of routing supervision may depend on the reliability of the pseudo-label generation process.

In addition, while our analysis suggests that SAE latent features correlate with retrieval-related patterns, it does not directly establish a causal relationship between sparse latent activations and retrieval utility. Our current findings are primarily based on correlational evidence, and further investigation is required to determine whether SAE features reflect retrieval-specific mechanisms.

Finally, this work does not thoroughly analyze how SAE scale, sparsity level, and backbone LLM size affect adaptive retrieval performance. Systematic investigation across different model scales and SAE configurations remains an important direction for future work.

References

Aisha Alansari and Hamzah Luqman. 2026. Large language models hallucination: A comprehensive survey. *Computer Science Review*, 61:100970.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2025. Do i know this entity? knowledge awareness and hallucinations in language models. In *International Conference on Learning Representations*, volume 2025, pages 77965–78000.

Jack Gallifant, Shan Chen, Kuleen Sasse, Hugo Aerts, Thomas Hartvigsen, and Danielle Bitterman. 2025. Sparse autoencoder features for classifications and transferability. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29939–29963, Suzhou, China. Association for Computational Linguistics.

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.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7969–7992.

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Hao Kang, Tevin Wang, and Chenyan Xiong. 2025. Interpret and control dense retrieval with sparse latent features. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 700–709, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1690–1712, Suzhou, China. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013.

Chunlei Xin, Shuheng Zhou, Huijia Zhu, Weiqiang Wang, Xuanang Chen, Xinyan Guan, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2025. Sparse latents steer retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4547–4562, Vienna, Austria. Association for Computational Linguistics.

Guangzhi Xiong, Zhenghao He, Bohan Liu, Sanchit Sinha, and Aidong Zhang. 2025. Toward faithful retrieval-augmented generation with sparse autoencoders. *arXiv preprint arXiv:2512.08892*.

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.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. 2025. SeaKR: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27022–27043, Vienna, Austria. Association for Computational Linguistics.

A Appendix

A.1 Experimental Details

Datasets. To evaluate the effectiveness of SAE-RAG, we use four open-domain QA benchmarks covering both single-hop and multi-hop reasoning: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017), HotpotQA (Yang et al., 2018), and 2WikiMultiHopQA (Ho et al., 2020). For each dataset, we randomly sample 1,000 questions from the development split using a fixed random seed and evaluate all methods on the same subset for controlled comparison. Following prior RAG studies, we use BM25-indexed Wikipedia as the retrieval corpus. **Baselines.** To evaluate the effectiveness of SAE-RAG, we compare against retrieval-based QA systems and adaptive retrieval methods. First, we include a closed-book baseline, **No RAG**, which does not use retrieval and relies solely on the backbone LLM's parametric knowledge. We also compare against **Single RAG**, which performs retrieval for every question, in order to analyze the effect of retrieval augmentation itself.

As adaptive retrieval baselines, we include FLARE and DRAGIN, selective retrieval methods that dynamically determine retrieval utility based on uncertainty or confidence signals. While these methods also aim to perform selective retrieval, our approach differs in that it uses SAE-based sparse latent signals extracted during query encoding rather than observable uncertainty signals arising during generation.

For fair comparison, all baselines are implemented using the same backbone LLM and retrieval environment. For reproducibility, baseline implementations are adapted from publicly available codebases whenever available.[1]

Implementation Details. We use Llama-3.1-8B-Instruct[2] as the backbone LLM for all generation stages. Unless otherwise specified, greedy decoding is applied with `max_new_tokens=256` for decomposition and `max_new_tokens=100` for final

[1] https://github.com/oneal2000/DRAGIN

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

answer generation. We additionally use fixed 3-shot demonstrations for final answer generation across all datasets.

For SAE-based retrieval routing, we employ the pretrained EleutherAI/sae-llama-3-8b-32x sparse autoencoder[3]. SAE parameters remain frozen during adaptive retrieval training. Using self-consistency labels ($n = 5$, temperature $= 0.7$, $\tau_h = 0.80$, $\tau_l = 0.20$), we extract SAE activation features from 5,000 training samples and train the routing classifier. The final routing threshold is selected to maximize F1 performance on the validation set.

Retrieval is performed using standard BM25 parameters ($k_1 = 1.2$, $b = 0.75$), returning top-$k = 5$ passages per query without minimum-score filtering.

Evaluation Metrics. We evaluate answer quality using Exact Match (EM) and F1. In addition, we report several retrieval-related statistics to analyze retrieval behavior and computational characteristics: RC (Retrieve Count; the number of BM25 retrieval calls), GC (Generate Count; the number of LLM generation calls), TC (Token Count; the total number of input tokens), and SC (Sentence Count; the total number of input sentences).

RC reflects how frequently external retrieval is performed, while GC measures the number of intermediate reasoning and generation steps executed by the system. TC and SC indicate the amount of contextual information processed by the LLM during reasoning and answer generation. These metrics provide complementary perspectives for analyzing selective retrieval behavior in addition to answer accuracy.

Retrieval routing classifiers are evaluated using AUC and F1, where retrieval-necessary samples are treated as the positive class.

A.2 Prompt for Question Decomposition

Table 7 summarizes the dataset-specific decomposition prompts used in our experiments. The prompts are designed to reflect the reasoning characteristics of each benchmark. For single-hop datasets such as NQ and TriviaQA, the prompts discourage unnecessary decomposition and favor direct retrieval whenever possible. In contrast, for multi-hop datasets such as HotpotQA and 2WikiMultiHopQA, the prompts explicitly encourage bridge-style and comparison-based decomposition to support compositional reasoning.

We additionally distinguish between LLM-query and SAE-RAG prompting strategies. LLM-query prompts generate both sub-questions and BM25 retrieval queries, whereas SAE-RAG prompts focus only on decomposition-aware sub-question generation without explicit retrieval-query construction. This separation allows SAE-RAG to decouple retrieval routing from retrieval query generation and better isolate retrieval utility prediction in the sparse latent routing stage.

A.3 Additional Prompt Details

Table 8 summarizes the answer generation prompts used in SAE-RAG. The prompts are designed to enforce concise factual generation while reducing unnecessary verbosity during retrieval-augmented reasoning. For retrieval-unnecessary sub-questions, the model generates answers using only internal parametric knowledge. For retrieval-necessary sub-questions and final answer generation, retrieved evidence is explicitly incorporated into the prompt.

In addition, SAE-RAG uses bridge-aware prompting to reuse internally generated intermediate answers when interpreting retrieved evidence for compositional reasoning. For comparison-style questions, the prompt additionally instructs the model to explicitly compare retrieved values before generating the final answer.

A.4 SAE Layer Analysis

Table 9 reports per-layer routing performance across four classifier families (EBM, Logistic Regression, MLP, and XGBoost) using SAE features extracted from different transformer layers of Llama-3.1-8B-Instruct. We evaluate each classifier on the retrieval-necessity (Known/Unknown) detection task using Train AUC, Test AUC, and Test F1.

Overall, middle transformer layers consistently achieve stronger retrieval routing performance than shallow or final layers. In particular, layers 10–14 show the most stable and highest Test AUC across classifiers, suggesting that intermediate representations provide more separable routing features than shallow or final layers. Among all configurations, XGBoost achieves the strongest overall routing performance, reaching the highest Test AUC of 0.904 at layer 13 while maintaining consistently strong Test F1 scores across layers.

We additionally observe that MLP classifiers achieve nearly perfect training performance across

[3] https://huggingface.co/EleutherAI/sae-llama-3-8b-32x

Dataset	Method	Decomposition Strategy	Prompt Characteristics
NQ	LLM Query	Decompose only for bridge-entity reasoning	Includes explicit BM25 query generation rules, bridge entity constraints, and retrieval-oriented keyword construction. Default behavior is single-query retrieval.
NQ	SAE-RAG	Decompose only for bridge-entity reasoning	Uses self-contained sub-question generation without BM25 query generation. Focuses on decomposition-aware routing decisions.
TriviaQA	LLM Query	Strong preference for no decomposition	Emphasizes single-hop factual lookup and discourages decomposition unless an unnamed bridge entity is strictly required. Includes BM25 keyword generation.
TriviaQA	SAE-RAG	Strong preference for no decomposition	Uses decomposition-only prompting for bridge reasoning while avoiding retrieval-query generation instructions.
HotpotQA	LLM Query	Two-step bridge reasoning	Encourages decomposition for bridge and attribute-chain reasoning. Includes step-wise BM25 retrieval query generation for intermediate entities.
HotpotQA	SAE-RAG	Two-step bridge reasoning	Uses explicit multi-hop decomposition with self-contained sub-questions, without retrieval-query generation instructions.
2WikiMultiHopQA	LLM Query	Multi-hop and comparison reasoning	Supports bridge, comparison, and multi-hop decomposition patterns. Includes explicit BM25 query construction for chained reasoning and comparison lookup.
2WikiMultiHopQA	SAE-RAG	Multi-hop and comparison reasoning	Uses decomposition-oriented prompting for bridge and comparison reasoning while omitting retrieval-query generation instructions.

Table 7: Summary of dataset-specific decomposition prompts used in SAE-RAG and LLM-query baselines. LLM-query prompts additionally generate BM25 retrieval queries, whereas SAE-RAG prompts focus only on decomposition-aware sub-question generation.

almost all layers, indicating substantial overfitting despite relatively weaker generalization performance on the test set. In contrast, XGBoost provides a better balance between expressive capacity and generalization, yielding the most stable routing performance across layers. Based on these results, we select layer 13 and XGBoost as the default routing configuration in SAE-RAG.

Prompt Type	Usage	Prompt Template
Final Answer Generation	Answer generation using retrieved evidence	*You are a factual QA assistant. Extract the answer directly from the retrieved context. Give the shortest possible answer (1–5 words). No explanations. Always end with "So the answer is [answer]."*
Bridge-aware Final Generation	Final answer generation with internal bridge knowledge	*When Internal Knowledge provides a bridge answer, use it as a stepping stone to interpret the Retrieved Context. For comparison questions, explicitly compare retrieved values before answering.*
Internal Sub-question Answering	Answer generation for retrieval-unnecessary sub-questions	*Answer the question with a short factual phrase. Always end with "So the answer is [answer]."*
Retrieval-based Sub-question Answering	Temporary answer generation for retrieval-necessary sub-questions	*Use the provided context and previous answers to answer the question with a short factual phrase. Always end with "So the answer is [answer]."*
Bridge Entity Extraction	Entity extraction from retrieved context	*Extract the answer entity from the context. Give only the entity name (1–5 words). Always end with "So the answer is [answer]."*

Table 8: Summary of answer generation prompts used in SAE-RAG. Prompts are designed to encourage concise factual generation while supporting bridge-aware reasoning and retrieval-grounded answer generation.

Layer	EBM			LR			MLP			XGBoost		
	Tr.AUC	Te.AUC	Te.F1	Tr.AUC	Te.AUC	Te.F1	Tr.AUC	Te.AUC	Te.F1	Tr.AUC	Te.AUC	Te.F1
0	0.900	0.764	0.899	0.715	0.687	0.896	1.000	0.803	0.905	1.000	**0.833**	**0.923**
1	0.930	0.762	0.907	0.771	0.730	0.896	1.000	0.818	0.920	1.000	**0.822**	**0.928**
2	0.926	0.763	0.904	0.802	0.757	0.899	0.999	0.803	0.893	1.000	**0.819**	**0.928**
3	0.953	0.811	0.911	0.837	0.775	0.899	1.000	0.820	0.921	1.000	**0.863**	**0.930**
4	0.943	0.799	0.908	0.853	0.764	0.898	1.000	0.826	0.912	1.000	**0.852**	**0.928**
5	0.933	0.813	0.904	0.860	0.794	0.896	1.000	0.854	0.901	1.000	**0.877**	**0.928**
6	0.934	0.834	0.908	0.876	0.813	0.905	1.000	0.863	0.916	1.000	**0.875**	**0.931**
7	0.935	0.830	0.908	0.877	0.798	0.892	1.000	0.874	0.925	1.000	**0.878**	**0.929**
8	0.947	0.829	0.913	0.871	0.804	0.902	1.000	0.846	0.920	1.000	**0.879**	**0.933**
9	0.957	0.838	0.912	0.885	0.797	0.905	1.000	0.851	0.923	1.000	**0.877**	**0.931**
10	0.944	0.829	0.909	0.891	0.815	0.906	1.000	0.878	0.926	1.000	**0.881**	**0.929**
11	0.964	0.842	0.913	0.905	0.805	0.902	1.000	**0.874**	0.918	1.000	0.861	**0.930**
12	0.953	0.829	0.908	0.887	0.813	0.904	1.000	0.879	0.909	1.000	**0.890**	**0.929**
13	0.963	0.855	0.906	0.892	0.819	0.903	1.000	0.883	**0.934**	1.000	**0.904**	0.930
14	0.946	0.841	0.905	0.901	0.827	0.900	1.000	0.871	0.926	1.000	**0.889**	**0.933**
15	0.946	0.830	0.908	0.904	0.816	0.903	1.000	0.870	0.922	1.000	**0.882**	**0.927**
16	0.931	0.822	0.904	0.899	0.804	0.896	1.000	0.860	0.921	1.000	**0.872**	**0.927**
17	0.943	0.836	0.903	0.894	0.797	0.895	1.000	0.849	0.918	1.000	**0.885**	**0.932**
18	0.950	0.823	0.907	0.893	0.789	0.899	1.000	0.858	0.913	1.000	**0.879**	**0.925**
19	0.963	0.830	0.910	0.899	0.795	0.902	1.000	0.854	0.922	1.000	**0.886**	**0.929**
20	0.962	0.834	0.902	0.895	0.788	0.891	1.000	0.857	0.926	1.000	**0.890**	**0.932**
21	0.962	0.833	0.904	0.896	0.789	0.893	1.000	0.844	0.914	1.000	**0.872**	**0.934**
22	0.962	0.835	0.911	0.894	0.795	0.898	1.000	0.849	0.910	1.000	**0.868**	**0.931**
23	0.979	0.826	0.910	0.905	0.798	0.900	1.000	0.845	0.905	1.000	**0.877**	**0.928**
24	0.954	0.820	0.914	0.896	0.803	0.897	1.000	0.865	0.925	1.000	**0.873**	**0.930**
25	0.951	0.797	0.903	0.894	0.762	0.887	1.000	**0.851**	0.921	1.000	0.844	**0.928**
26	0.938	0.807	0.908	0.887	0.780	0.900	1.000	0.852	0.914	1.000	**0.858**	**0.930**
27	0.943	0.817	0.908	0.903	0.795	0.900	1.000	0.836	0.901	1.000	**0.879**	**0.926**
28	0.943	0.795	0.903	0.892	0.751	0.883	1.000	0.837	0.905	1.000	**0.852**	**0.925**
29	0.962	0.819	0.903	0.899	0.781	0.899	1.000	0.857	0.926	1.000	**0.866**	**0.929**
30	0.934	0.812	0.907	0.901	0.784	0.888	1.000	0.855	0.916	1.000	**0.860**	**0.932**

Table 9: Per-layer Train AUC, Test AUC, and Test F1 for all four classifiers (EBM, LR, MLP, XGBoost) on the Known/Unknown detection task (Llama-3.1-8B-Instruct hidden states). **Bold** indicates the best Test AUC and Test F1 per layer.