DIGAA: DECOUPLING INFORMATION GATHERING AND ACCURATE ANSWERING IN RETRIEVAL-AUGMENTED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

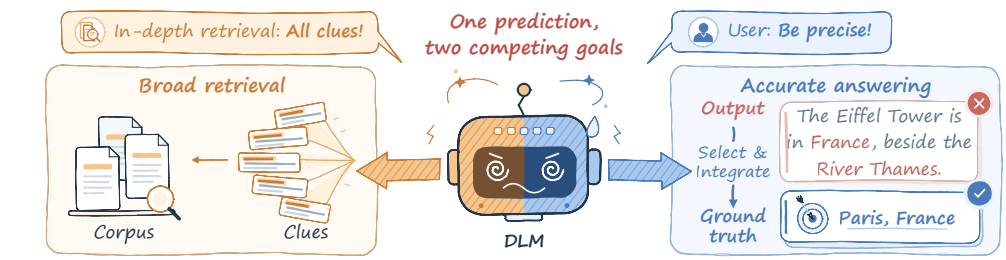
Diffusion language models (DLMs) have recently attracted attention in retrieval-augmented generation (RAG) owing to their ability to predict tokens at all masked positions in parallel at each denoising step. These parallel predictions can reveal bridge entities early (Jünger et al., 2026) and provide diverse information, thereby enabling in-depth and broad retrieval. However, this introduces noise along with additional information, making it harder for the model to generate accurate answers. The conflict here is that in-depth and broad retrieval of external knowledge introduces wide-ranging information, while answer generation requires accuracy. Therefore, we introduce Decoupled Information Gathering and Accurate Answering (DIGAA), which explicitly separates the roles of information gathering and answer generation to reduce this conflict. During information gathering, the model uses bridge entities generated during intermediate reasoning to explore external knowledge and collect relevant facts. During answer generation, the model needs to select and integrate this information to answer directly and accurately. Experiments on Dream-7B-Instruct and LLaDA-8B-Instruct across six factual QA benchmarks show that, relative to the strongest baseline for each benchmark and metric, DIGAA improves Dream's average EM and F1 by 4.10 and 5.16 percentage points, respectively, and LLaDA's by 12.39 and 15.61 percentage points. These results indicate that decoupling wide-ranging information gathering from accurate answering is a promising design principle when DLMs are augmented with dynamic retrieval.

1 INTRODUCTION

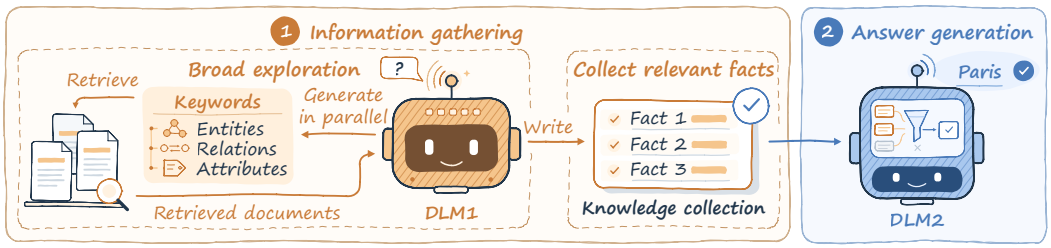
Traditional retrieval-augmented generation (RAG) methods for diffusion language models (DLMs) integrate retrieved external knowledge into a single generation process that produces the final answer (Yu et al., 2026; Jünger et al., 2026). Existing methods can be classified as static or dynamic depending on whether the retrieved documents are updated during generation (Jünger et al., 2026). Static retrieval uses a fixed document set retrieved using the question and does not retrieve new documents during generation (Kim & Ye, 2026). SARDI performs dynamic retrieval for DLMs by retrieving additional documents using the updated response after each decoding step, providing richer evidence to support generation and improve performance (Jünger et al., 2026).

In dynamic retrieval, intermediate predictions guide the search for additional evidence (Jünger et al., 2026), helping identify missing information when a single retrieval step is insufficient (Trivedi et al., 2023). At each denoising step, DLMs predict masked tokens in parallel (Nie et al., 2025; Jünger et al., 2026), potentially revealing keywords at later response positions before preceding text is complete and enabling earlier retrieval of relevant documents (Jünger et al., 2026). Unlike autoregressive models, which condition each new token on preceding tokens (Nie et al., 2025), DLMs do not condition on other tokens predicted within the same step. This can alleviate left-to-right error accumulation during query construction (Jünger et al., 2026). Beyond this advantage, Zhang et al. (2026) report higher semantic diversity across texts generated by DLMs than by autoregressive models. This supports exploring DLM-generated text as a source of diverse retrieval clues, since balancing relevance and diversity among retrieved documents can improve question answering (Khan et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Coupled retrieval and answering



(b) DIGAA: gather first, then answer

Figure 1: Decoupling information gathering and accurate answering in diffusion language models augmented with dynamic retrieval. (a) Coupling retrieval and answering requires one prediction to serve competing goals: broad information gathering versus answer accuracy. (b) DIGAA uses two dedicated stages, one for information gathering and the other for answer generation. During information gathering, keywords generated in parallel by the DLM guide broader and deeper retrieval, and relevant facts are collected into a knowledge collection. During answer generation, the DLM selects and integrates information from this collection to answer the question directly and accurately, and its output no longer needs to guide retrieval.

More diverse retrieval clues can broaden the retrieved context but may also introduce irrelevant information, making accurate answering more difficult. When the DLM's predictions both guide retrieval and serve as the final answer, the model must pursue two competing goals: generating diverse clues for further exploration and selecting and integrating relevant information to produce an accurate answer. To reduce the conflict between these goals, we introduce Decoupled Information Gathering and Accurate Answering (DIGAA), which assigns information gathering and answer generation to separate stages, as shown in Figure 1.

1. Information gathering. We seek a broader range of information to build a richer context for reasoning. At each denoising step, we introduce an additional prediction step into the conventional iterative self-augmenting retrieval process (Jünger et al., 2026). It is used to construct a good knowledge collection that can support the following answer generation.

2. Answer generation. The model then generates the final answer from the question and the knowledge collection. The DLM no longer needs to output diverse keywords to guide retrieval, and its main objective is to answer the question directly and accurately. This stage is designed to remove intermediate information used only for retrieval and derive conclusions by integrating facts in the knowledge collection. We evaluate DIGAA with Dream-7B-Instruct (Ye et al., 2025) and LLaDA-8B-Instruct (Nie et al., 2025) on six factual QA benchmarks. The results show that DIGAA outperforms all baselines in EM and token F1 on every model-benchmark pair.

Our contributions are:

1. We argue that decoupling information gathering from accurate answering can benefit DLMs augmented with dynamic retrieval by reducing the conflict between generating more clues that support further retrieval and producing direct and accurate answers. Adding a separate answer-generation stage improves average EM and F1 for all nine baselines.
2. We introduce DIGAA, a training-free framework that augments DLMs with dynamic retrieval and explicitly separates information gathering from answer generation. We evaluate DIGAA

with Dream-7B-Instruct (Ye et al., 2025) and LLaDA-8B-Instruct (Nie et al., 2025) on six factual QA benchmarks. Relative to the strongest baseline for each benchmark and metric, the average EM and F1 gains are 4.10 and 5.16 percentage points on Dream, and 12.39 and 15.61 percentage points on LLaDA, respectively. We further compare DIGAA with the baselines in terms of quality and latency, showing that DIGAA leads in the trade-off between answer quality and inference latency.

3. We analyze the gains from the answer-generation stage by comparing first-stage and final outputs across six benchmarks. The case analysis shows that answer generation can not only remove irrelevant information and locate answers, but also infer answers not explicitly mentioned in the knowledge collection by integrating the collected facts.

2 Related Work

Static retrieval. In early RAG systems based on autoregressive models, including RAG and Fusion-in-Decoder (FiD), passages are retrieved using the question and the document set remains fixed during answer generation (Lewis et al., 2020; Izacard & Grave, 2021). A related line of work on autoregressive models improves how the model uses this context when it conflicts with parametric knowledge. CAD contrasts predictions with and without context (Shi et al., 2024), while AdaCAD and COIECD adjust decoding according to the degree or presence of conflict (Wang et al., 2025; Yuan et al., 2024). Most existing RAG methods for DLMs also follow this single-shot static retrieval setting (Jünger et al., 2026). SPREAD prioritizes token positions by their semantic relevance to the question, reducing semantic drift (Yu et al., 2026). ARAM uses a signal-to-noise estimate to adapt how strongly retrieved context guides each token prediction during denoising (Kim & Ye, 2026).

Dynamic retrieval. When a single retrieval based only on the question is insufficient, it is usually necessary to repeatedly retrieve documents based on the model's temporary predictions. For autoregressive models, IRCoT alternates retrieval with chain-of-thought reasoning, letting intermediate reasoning guide later searches (Trivedi et al., 2023). Self-RAG learns to retrieve on demand and uses reflection tokens to assess retrieved passages and generated responses (Asai et al., 2024). FLARE predicts an upcoming sentence. When the sentence contains low-confidence tokens, FLARE uses it to retrieve new documents before regenerating the sentence with the updated context (Jiang et al., 2023). However, these tentative text spans used for retrieval are generated from left to right, so a single early error can compound, leading to hallucinated queries that retrieve irrelevant documents (Jünger et al., 2026). In contrast, DLMs can predict all masked positions in parallel with a single forward pass at each denoising step, and these predictions can reveal bridge entities early, enabling earlier retrieval of documents needed for subsequent reasoning (Jünger et al., 2026). SARDI performs one forward pass per denoising step using documents retrieved at the preceding step, committing high-confidence tokens to the answer and using the same predictions to retrieve documents for the next step (Jünger et al., 2026). Despite separate thresholds, high-confidence tokens from the same prediction still serve both purposes. As a result, the answer may already be fixed while generation and retrieval continue (Huang et al., 2026). Furthermore, repeatedly conditioning these outputs on documents retrieved from earlier temporary predictions may reinforce the same clues and narrow subsequent exploration. In contrast, DIGAA not only decouples information gathering from answer generation, ensuring by design that answer generation begins only after retrieval is complete, but also further refines the dynamic retrieval process by performing two forward passes at each denoising step. It first generates keywords using documents retrieved based on the question, then uses these keywords for further retrieval to maintain relevance to the question and reduce self-reinforcement. It then collects relevant facts from the newly retrieved documents, allowing retrieval to progress as information accumulates.

3 Method

As shown in Figure 2, DIGAA consists of two stages: information gathering and answer generation. The first stage uses temporary predictions to guide in-depth and broad retrieval and collects relevant facts in a knowledge collection. The second stage selects and integrates information from this collection to answer the question directly and accurately.

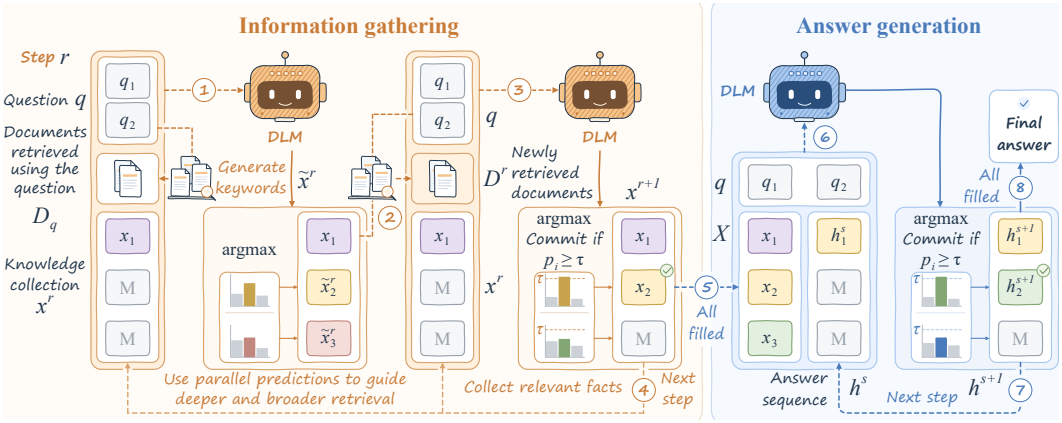


Figure 2: Overview of DIGAA. During information gathering, the DLM takes question q, the knowledge collection x^r, and fixed documents D_q retrieved using q as inputs in step r. The DLM generates keywords in parallel within $\widetilde{x}^r$ to retrieve documents D^r. A second parallel prediction uses D^r to update x^r by committing positions whose predicted tokens have confidence scores at or above the threshold τ. During answer generation, the DLM selects and integrates information from the completed collection X to answer q directly and accurately. At each denoising step s, the DLM predicts all masked positions in the answer sequence h^s in parallel based on q and X, and commits positions whose predicted tokens have confidence scores of at least τ. In both stages, if no prediction meets the threshold, the DLM fills the position with the highest-confidence prediction. Each stage repeats until no masks remain (proceeds along branch ⑤ or ⑧).

3.1 Information Gathering

The goal of information gathering is to explore external knowledge and collect relevant facts for subsequent answer generation. Given a question q and an external corpus $\mathcal{C}$, let $\mathcal{R}(s, k)$ denote a retriever that returns the top k documents from $\mathcal{C}$ for query s. We retrieve an initial document set $D_q = \mathcal{R}(q, k)$ using the question and keep this set fixed throughout this stage.

We construct the knowledge collection as a token sequence $X = (x_1, \ldots, x_L)$ of length L, initialized with mask tokens. Let x^r denote its state at denoising step r, with x^0 fully masked, and let M^r denote the set of masked positions. At each step, the DLM uses the question, the current sequence, and the fixed documents to predict all masked positions in parallel (Nie et al., 2025; Ye et al., 2025). For each masked position i, we select the most probable token as the predicted token $\widetilde{x}_i^r$:

$$\widetilde{x}_i^r = \arg\max_{v \in \mathcal{V}} p_i(v \mid x^r, q, D_q), \qquad i \in M^r, \tag{1}$$

where $\mathcal{V}$ is the vocabulary and $p_i(v \mid x^r, q, D_q)$ is the probability that the DLM assigns to token v at position i given the current sequence, question, and fixed documents.

These parallel token predictions can provide intermediate entities absent from the original question (Jünger et al., 2026), guiding more in-depth retrieval. They can also support broader retrieval because tokens predicted within the same step are not conditioned on one another. Compared with autoregressive models, which generate tokens conditionally from left to right, DLMs impose fewer constraints on token predictions, providing more diverse information to guide broader retrieval.

We temporarily fill each masked position i in x^r with $\widetilde{x}_i^r$ to obtain a temporary completion $\widetilde{x}^r$ used only to guide retrieval. We combine $\widetilde{x}^r$ with the question to retrieve a new document set:

$$D^r = \mathcal{R}(q \,\|\, \widetilde{x}^r, k), \tag{2}$$

where $\|$ denotes text concatenation.

The preceding predictions guide exploration of external knowledge. We then collect relevant facts by predicting the masked positions in x^r based on the newly retrieved document set D^r. For each masked position i, we select the most probable token as the predicted token $y_{r,i}$:

$$y_{r,i} = \arg\max_{v \in \mathcal{V}} p_i(v \mid x^r, q, D^r), \qquad i \in M^r. \tag{3}$$

The second prediction uses the same sequence x^r as the first, while the document context changes from D_q to D^r. For the first prediction, which guides retrieval, we use D_q rather than the document set D^{r-1} retrieved during the previous denoising step. Because we use model outputs to guide retrieval, repeatedly conditioning these outputs on documents retrieved from earlier temporary predictions may reinforce the same clues and narrow subsequent exploration. Using D_q is intended to keep exploration tied to the original question, while x^r retains the facts already collected, keeping retrieval relevant to the question while allowing exploration to advance as information accumulates.

To exploit the parallel decoding capabilities of DLMs, we adopt a probability threshold that allows multiple predicted tokens whose probabilities meet the threshold to be retained simultaneously, adapting the number of positions filled at each denoising step to the predicted token probabilities (Wu et al., 2026). Given a probability threshold τ, we select masked positions whose predicted tokens have probabilities at least τ:

$$U_\tau^r = \{i \in M^r : p_i(y_{r,i} \mid x^r, q, D^r) \geq \tau\}. \tag{4}$$

We fill each position $i \in U_\tau^r$ with its predicted token $y_{r,i}$, setting $x_i = y_{r,i}$. If no predicted token reaches the threshold, we fill the single position whose predicted token has the highest probability. The remaining masked positions stay masked, yielding x^{r+1}.

We repeat this process until no masks remain, obtaining the completed knowledge collection $X = (x_1, \ldots, x_L)$. Since positions can be filled in any order, the model can progressively incorporate retrieved information into the positions in the sequence where it is needed. This process can collect intermediate entities and relevant facts. The model can use this retained information across intermediate denoising steps to progressively deepen its exploration.

3.2 Answer Generation

At this stage, we stop retrieval and let the DLM generate the final answer based on the question and the knowledge collection X. Its output no longer needs to guide retrieval. Instead, the model selects and integrates the collected information to answer the question directly and accurately. This stage can remove intermediate information used only for retrieval and express conclusions that follow from the collected facts but are not explicitly stated in the collection.

We initialize a new answer sequence with L mask tokens. Let h^s denote its state at denoising step s and N^s its set of masked positions. The model predicts these positions in parallel using the question and the completed knowledge collection. The predicted token at each masked answer position is

$$a_{s,i} = \arg\max_{v \in \mathcal{V}} p_i(v \mid h^s, q, X), \qquad i \in N^s. \tag{5}$$

We use the same probability threshold τ and filling rule as in information gathering. At each step, we fill all positions satisfying $p_i(a_{s,i} \mid h^s, q, X) \geq \tau$ with their predicted tokens in parallel. If none qualifies, we fill the position whose predicted token has the highest probability. The remaining masked positions stay masked, yielding h^{s+1}. We repeat this process until all masks are filled, and the resulting sequence serves as the final answer. Appendix A provides the complete algorithm.

4 Experiments

4.1 Experimental Setup

Models. We evaluate Dream-7B-Instruct (Ye et al., 2025) and LLaDA-8B-Instruct (Nie et al., 2025). All methods use the same instruction-tuned checkpoints without additional training.

Benchmarks. We use six factual QA benchmarks: Natural Questions (NQ) (Kwiatkowski et al., 2019), ZsRE (Levy et al., 2017), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), HotpotQA (Yang et al., 2018), and SynthWorlds-RM (Gu et al., 2025). These benchmarks cover settings including open-domain QA (NQ and HotpotQA), slot filling formulated as QA (ZsRE), and multi-hop QA (2WikiMultiHopQA, MuSiQue, HotpotQA, and SynthWorlds-RM), making them suitable for evaluating knowledge utilization in language models. For retrieval, we use each benchmark's officially provided corpus when available and otherwise follow existing evaluation protocols. Appendix B provides the specific settings for each benchmark and its corpus.

Retrieval and generation. We use the sparse retriever BM25 (Robertson & Zaragoza, 2009), retrieving $k = 5$ passages at each retrieval step. Appendix D reports additional results with $k = 3$ and 7. We also report results using the dense retriever Contriever-MSMARCO (Izacard et al., 2022) in Appendix E. Our prompts and decoding settings follow ARAM (Kim & Ye, 2026). Appendix C provides the implementation details.

Baselines. Under the same evaluation settings, we compare DIGAA with Official Direct (Ye et al., 2025; Nie et al., 2025), Standard RAG (Lewis et al., 2020), CAD-MDM (Shi et al., 2024), AdaCAD-MDM (Wang et al., 2025), COIECD-MDM (Yuan et al., 2024), SPREAD (Yu et al., 2026), A-CFG (Li et al., 2025), ARAM (Kim & Ye, 2026), and SARDI (Jünger et al., 2026). We report normalized exact match (EM) (Rajpurkar et al., 2016) and token-level F1 (Rajpurkar et al., 2016) for each method on each benchmark. The baseline implementations and fixed hyperparameters follow prior work (Kim & Ye, 2026; Jünger et al., 2026). Appendix C.2 provides the details.

4.2 Results and Analysis

Main results. Table 1 compares DIGAA ($\tau = 0.90$) with nine baselines across six factual QA benchmarks. DIGAA outperforms every baseline in EM and token F1 on all 12 model-benchmark pairs. Relative to the strongest baseline for each benchmark and metric, the average EM and F1 gains are 4.10 and 5.16 percentage points on Dream, and 12.39 and 15.61 percentage points on LLaDA, respectively. Appendix D reports supplementary results with retrieval depths of 3 and 7, and Appendix E reports results with the dense retriever Contriever-MSMARCO. In each setting, DIGAA again outperforms every baseline in both metrics on all model-benchmark pairs. The consistent performance advantages across different benchmark types, retrievers, models, metrics, and retrieval depths demonstrate the superior performance and broad robustness of DIGAA.

Table 1: Results of DIGAA ($\tau = 0.90$) and nine baselines across six factual QA benchmarks. Bold marks the best result for each model, benchmark, and metric. Underlining marks the best baseline. Retrieval-based methods use the sparse retriever BM25 to retrieve the top 5 documents. DIGAA outperforms all baselines in EM and token F1 on every model-benchmark pair.

Method	NQ		ZsRE		2Wiki		MuSiQue		HotpotQA		Synth-RM	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream	15.9	26.7	4.5	17.0	23.0	25.9	1.7	8.8	15.6	23.9	7.1	12.6
+ RAG	30.7	42.6	48.2	59.4	26.3	31.1	9.2	18.3	29.6	40.5	16.1	22.8
+ CAD	24.0	36.5	37.6	54.3	13.4	19.1	7.1	16.0	22.8	33.9	10.3	17.9
+ AdaCAD	29.7	41.9	45.4	58.3	25.6	30.5	9.1	18.5	29.0	39.6	14.7	21.1
+ COIECD	27.2	39.2	43.8	56.8	21.2	26.2	7.9	17.9	26.5	37.1	13.8	20.1
+ SPREAD	30.8	42.3	44.7	56.7	25.2	29.8	8.0	16.9	29.1	39.2	16.1	22.5
+ A-CFG	29.2	41.9	42.0	58.0	25.2	30.6	9.3	19.6	28.0	39.6	14.3	21.5
+ ARAM	30.8	42.9	48.0	58.8	25.0	29.6	7.8	16.6	28.6	39.4	14.8	21.2
+ SARDI	33.5	45.8	49.8	59.8	27.8	33.2	12.0	21.4	32.4	43.5	17.9	25.0
+ DIGAA	**35.8**	**48.7**	**53.2**	**65.9**	**36.0**	**41.4**	**13.8**	**24.8**	**37.0**	**49.0**	**22.2**	**29.7**
LLaDA	12.2	19.0	4.7	11.6	20.1	21.6	1.0	5.1	14.5	20.4	7.8	10.7
+ RAG	19.7	30.8	6.8	25.7	14.4	18.6	6.9	13.2	23.8	32.4	5.9	11.7
+ CAD	16.3	26.5	6.4	22.9	8.0	12.6	5.0	10.8	13.0	21.7	5.6	12.0
+ AdaCAD	19.1	30.0	7.5	25.0	12.7	16.9	6.4	13.1	22.1	30.6	6.5	13.0
+ COIECD	16.8	27.3	7.0	23.6	8.7	13.0	5.5	11.6	16.2	24.6	5.8	12.5
+ SPREAD	20.1	30.6	10.5	28.3	18.9	23.2	6.5	13.6	25.9	35.5	6.8	12.8
+ A-CFG	20.2	31.3	9.3	28.2	16.2	21.0	6.2	12.7	21.4	30.9	6.6	13.2
+ ARAM	23.7	34.5	21.2	36.3	18.7	22.9	7.1	14.3	27.5	37.0	7.7	13.5
+ SARDI	22.1	34.4	7.1	27.3	17.0	22.3	11.0	18.8	26.0	35.5	7.1	13.8
+ DIGAA	**30.0**	**43.7**	**45.3**	**60.8**	**36.8**	**47.3**	**16.5**	**28.3**	**36.7**	**48.9**	**20.5**	**28.3**

DIGAA leads in the trade-off between answer quality and inference latency. DIGAA can trade throughput for accuracy by adjusting the token-commitment confidence threshold τ. A higher τ leads to fewer tokens being committed at each denoising step, requiring more denoising rounds and reducing throughput while improving accuracy. A lower τ has the opposite effect. Figure 3 compares the quality and latency of DIGAA at different thresholds with those of the baselines, showing

that DIGAA leads in the trade-off between answer quality and inference latency. In particular, at $\tau = 0.50$, both the average performance and throughput of DIGAA across the two models exceed the corresponding two-model averages of every baseline. Appendix F further reports the mean numbers of forward evaluations and retrieval calls for each method.

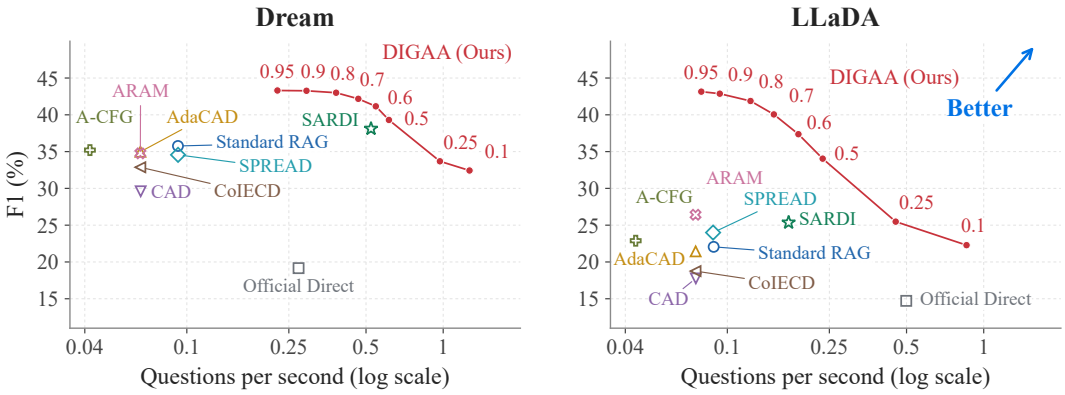


Figure 3: Trade-offs between quality and latency on Dream (left) and LLaDA (right). F1 is averaged over six benchmarks. Speed is the reciprocal of mean latency, measured as end-to-end wall-clock time on a single RTX 3090 GPU with batch size 1. Red labels indicate the confidence threshold τ used by DIGAA. The horizontal axis is logarithmic. Appendix F reports the trade-offs between quality and latency based on EM. DIGAA leads in the trade-off between answer quality and inference latency.

Answer-generation gains are method-agnostic. We apply the answer-generation stage of DIGAA to each baseline's generated output. As shown in Table 2, every baseline improves in average EM and F1, while DIGAA remains ahead of all baselines augmented with this stage. Moreover, a single answer-generation step is generally sufficient. As shown in Figure 4, repeating the procedure from the second step onward brings only marginal additional gains.

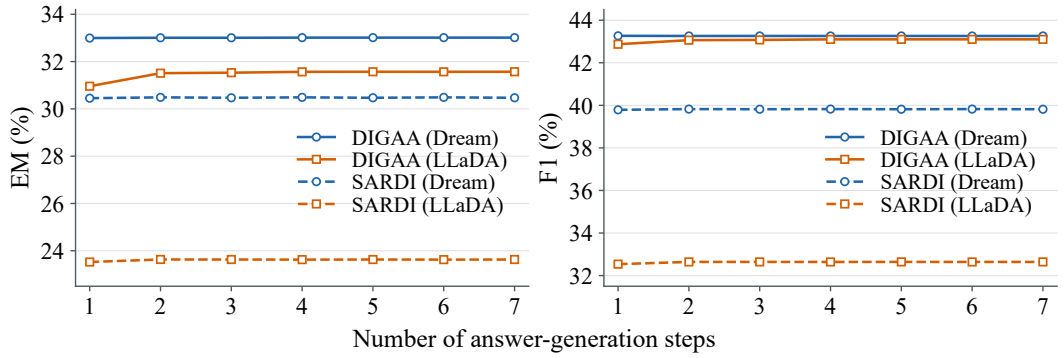


Figure 4: Repeating the answer-generation stage in DIGAA and SARDI. The left panel shows EM and the right panel shows F1, both averaged over six benchmarks. The first point corresponds to the complete DIGAA in Table 1 and to SARDI after one application of the answer-generation stage. Subsequent steps run the answer-generation stage using the preceding answer, without further retrieval. The results show that a single answer-generation step is generally sufficient.

Analysis of answer-generation gains. We compare the outputs generated by the first stage of DIGAA on six benchmarks with the corresponding outputs after the answer-generation stage. We use a large language model to inspect examples whose EM and F1 scores improve after the answer-generation stage. Table 3 illustrates several types of gains. For example, in the answer localization example, the answer-generation stage locates the answer in the first-stage output and removes irrelevant text to produce a concise answer; in the example of inference from collected facts, the answer-generation stage uses the stated years 1940 and 1972 to answer whether the two

Table 2: Average results across six benchmarks after applying the answer-generation stage of DIGAA to each baseline's generated output. Top$k$ denotes retrieving the top k documents with BM25. Smaller italic sans-serif values in parentheses show the original scores, and regular values below show the scores after applying the answer-generation stage. Green arrows indicate improvements of the latter over the former, while red arrows indicate decreases. Bold marks the best result for each model, retrieval depth, and metric. These results show that answer generation is broadly beneficial, and DIGAA remains ahead of all baselines augmented with the answer-generation stage.

Method	Dream						LLaDA					
	Top3		Top5		Top7		Top3		Top5		Top7	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Direct	(11.2) 12.1↑	(18.9) 19.7↑	(11.3) 12.3↑	(19.2) 20.0↑	(11.3) 12.1↑	(19.0) 19.8↑	(10.3) 10.5↑	(14.7) 14.9↑	(10.0) 10.2↑	(14.7) 14.9↑	(10.2) 10.4↑	(14.7) 14.8↑
RAG	(27.2) 28.7↑	(36.2) 37.9↑	(26.7) 28.4↑	(35.8) 37.6↑	(27.8) 29.7↑	(37.1) 39.2↑	(17.7) 22.7↑	(26.4) 30.6↑	(12.9) 20.6↑	(22.1) 28.9↑	(11.7) 20.6↑	(20.7) 28.9↑
CAD	(19.0) 25.4↑	(29.3) 35.0↑	(19.2) 25.2↑	(29.6) 34.9↑	(18.1) 25.2↑	(28.9) 35.3↑	(9.8) 18.8↑	(18.4) 26.2↑	(9.0) 18.2↑	(17.8) 26.0↑	(9.1) 18.6↑	(17.6) 26.3↑
ADACAD	(26.3) 28.2↑	(35.6) 37.6↑	(25.6) 27.7↑	(35.0) 37.1↑	(26.1) 28.6↑	(35.6) 38.2↑	(17.1) 22.4↑	(25.7) 30.5↑	(12.4) 20.1↑	(21.4) 28.4↑	(11.1) 19.7↑	(20.0) 28.0↑
COIECD	(23.7) 26.8↑	(33.0) 36.2↑	(23.4) 26.6↑	(32.9) 36.1↑	(23.4) 27.6↑	(33.2) 37.3↑	(12.1) 19.7↑	(20.7) 27.5↑	(10.0) 18.7↑	(18.8) 26.5↑	(9.9) 18.7↑	(18.4) 26.5↑
SPREAD	(26.0) 27.6↑	(34.9) 36.7↑	(25.6) 27.4↑	(34.6) 36.6↑	(26.5) 28.7↑	(35.7) 38.1↑	(19.9) 23.6↑	(28.6) 31.7↑	(14.8) 20.9↑	(24.0) 29.3↑	(13.3) 20.8↑	(22.4) 29.2↑
A-CFG	(24.9) 27.1↑	(35.7) 37.8↑	(24.7) 26.8↑	(35.2) 37.3↑	(24.8) 27.7↑	(35.8) 38.6↑	(16.9) 22.7↑	(26.3) 31.2↑	(13.3) 20.6↑	(22.9) 29.2↑	(12.5) 20.9↑	(22.1) 29.5↑
ARAM	(26.7) 28.1↑	(35.4) 37.0↑	(25.8) 27.5↑	(34.8) 36.5↑	(26.3) 28.4↑	(35.7) 37.9↑	(23.3) 25.3↑	(31.3) 32.9↑	(17.6) 21.6↑	(26.4) 30.0↑	(15.1) 20.7↑	(24.2) 29.4↑
SARDI	(29.6) 31.0↑	(38.8) 40.3↑	(28.9) 30.5↑	(38.1) 39.8↑	(29.6) 31.6↑	(39.1) 41.2↑	(19.8) 25.4↑	(29.4) 34.2↑	(15.1) 23.5↑	(25.4) 32.5↑	(14.2) 23.6↑	(24.4) 32.8↑
DIGAA	**33.3**	**42.9**	**33.0**	**43.3**	**34.6**	**44.9**	**30.2**	**40.7**	**31.0**	**42.9**	**33.7**	**46.2**

court cases occurred in the same decade. Appendix C.4 provides the review prompt template. Of these improvements, 61% stem from locating concise, direct answers in the first-stage outputs. The remaining 39% stem from recombining noncontiguous words, generating answers that are not explicitly stated but can be inferred from the collected facts, completing incomplete answers, and other changes. These case checks show that answer generation can not only remove irrelevant information and locate answers, but also complete incomplete answers and generate answers that are not explicitly stated but follow from the collected facts. This is consistent with our design goal of selecting and integrating information to answer directly and accurately.

Ablation of information gathering. As shown in Table 4, we compare DIGAA with RAG + AG and SARDI + AG, which apply the same answer-generation (AG) stage to the original outputs of RAG and SARDI, respectively. This comparison keeps the answer-generation procedure identical, with only the first stage differing. SARDI + AG outperforms RAG + AG in both metrics on every model-benchmark pair, showing that dynamic retrieval remains beneficial even when both methods use the same answer-generation stage. SARDI uses one forward pass per denoising step to collect facts from previously retrieved documents and generate intermediate entities for the next retrieval. In contrast, DIGAA uses the fixed initial documents and the evolving knowledge collection to construct retrieval queries, then makes a separate prediction using the newly retrieved documents to update the collection. DIGAA outperforms SARDI + AG in both metrics on every model-benchmark pair. Averaged equally across six benchmarks, EM/F1 gains are 2.57/3.47 percentage points on Dream and 7.45/10.35 on LLaDA. With answer generation held fixed, these gains show that DIGAA's redesigned information gathering can further improve answer quality.

Table 3: Breakdown of answer-generation improvements in DIGAA, illustrated with shortened examples. The Type column indicates the type of benefit. The Without answer generation column shows examples of outputs from the first stage of DIGAA, and the With answer generation column shows the corresponding outputs after the answer-generation stage. The rightmost column reports the approximate share of improvements, with the last three types grouped together.

Type	Without answer generation	With answer generation	
Answer localization	**Question:** who plays at the prudential center in newark		∼61%
	The New Jersey Devils play at the Prudential Center in Newark.	New Jersey Devils	
Noncontiguous word recombination	**Question:** What does Smithfield Street Bridge cross over?		∼39%
	The bridge spans the Monongahela before the river joins the Allegheny.	Monongahela River	
Inference from collected facts	**Question:** Did Minersville School District v. Gobitis and Gravel v. United States occur in the same decade?		
	Minersville School District v. Gobitis occurred in 1940 and Gravel v. United States in 1972.	**No**	
Answer completion	**Question:** What is the full name of the viral respiratory disease in which Yi Guan's research helped to prevent the outbreak of?		
	SARS	**Severe acute respiratory syndrome**	

Table 4: Comparison of DIGAA with RAG and SARDI augmented with the answer-generation stage. The settings are the same as in Table 1, using BM25 to retrieve the top 5 documents and a confidence threshold of $\tau = 0.90$ for DIGAA. RAG + AG and SARDI + AG denote applying the answer-generation stage to the corresponding baseline's original output. Bold marks the best result among these three methods for each model, benchmark, and metric. DIGAA outperforms both alternatives in EM and token F1 on every model-benchmark pair, demonstrating the advantages of the information-gathering stage.

Model	Method	NQ		ZsRE		2Wiki		MuSiQue		HotpotQA		Synth-RM	
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream	RAG + AG	32.1	44.0	48.2	59.5	31.1	35.7	9.5	19.0	31.9	43.0	17.3	24.4
	SARDI + AG	34.5	46.8	49.9	60.0	32.6	37.7	12.1	21.9	34.5	45.9	19.0	26.4
	DIGAA	**35.8**	**48.7**	**53.2**	**65.9**	**36.0**	**41.4**	**13.8**	**24.8**	**37.0**	**49.0**	**22.2**	**29.7**
LLaDA	RAG + AG	25.3	35.4	29.5	43.5	24.0	27.7	7.4	14.3	26.7	35.8	10.8	16.4
	SARDI + AG	28.5	39.6	32.1	46.3	26.8	31.5	12.0	20.2	29.0	38.8	12.7	18.8
	DIGAA	**30.0**	**43.7**	**45.3**	**60.8**	**36.8**	**47.3**	**16.5**	**28.3**	**36.7**	**48.9**	**20.5**	**28.3**

5 Conclusion

In this work, we introduce DIGAA, a training-free framework that separates information gathering from answer generation in DLMs augmented with dynamic retrieval. When the same predictions must guide retrieval and serve as the final answer, the model is driven to generate diverse clues for further exploration and integrate relevant information to produce direct and accurate answers. DIGAA assigns the two goals to separate stages to reduce this conflict. DIGAA outperforms the baselines in EM and F1 on six factual QA benchmarks while maintaining a favorable quality-latency trade-off, demonstrating its effectiveness. Adding a separate answer-generation stage to the existing baselines also consistently improves the average EM and F1, further supporting the value of separating information gathering from answer generation. Our case analysis further shows that this stage can integrate collected facts to infer answers rather than merely remove irrelevant information and locate existing answers. Together, these findings support decoupling information gathering from answer generation as a promising design principle for DLMs augmented with dynamic retrieval.

Reproducibility Statement

The main paper and appendices specify the two model checkpoints, full evaluation sizes, external corpora, retrieval depth, and decoding settings. Appendix A provides complete pseudocode, Appendix B documents the datasets and corpora, Appendix C.1 provides the decoding settings, and Appendix C.3 provides the specific prompts. Anonymized code will be provided with the submission.

AI Use Statement

We used AI to check the grammar of the manuscript and to perform preliminary categorization of case-study examples. We manually spot-checked 10% of these results and found no obvious classification bias. The specific examples in Table 3 were manually selected from the samples. We take responsibility for all submitted content.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. URL https://arxiv.org/abs/1611.09268.

Ken Gu, Advait Bhat, Mike A. Merrill, Robert West, Xin Liu, Daniel McDuff, and Tim Althoff. Synthworlds: Controlled parallel worlds for disentangling reasoning and knowledge in language models. *arXiv preprint arXiv:2510.24427*, 2025.

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

Pengcheng Huang, Tianming Liu, Zhenghao Liu, Yukun Yan, Shuo Wang, Tong Xiao, Zulong Chen, and Maosong Sun. Empirical analysis of decoding biases in masked diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6853–6876, 2026. doi: 10.18653/v1/2026.acl-long.311. URL https://aclanthology.org/2026.acl-long.311/.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, 2021. doi: 10.18653/v1/2021.eacl-main.74. URL https://aclanthology.org/2021.eacl-main.74/.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=jKN1pXi7b0.

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

Paul Jünger, Justin Lovelace, Linxi Zhao, Dongyoung Go, and Kilian Q. Weinberger. Self-augmenting retrieval for diffusion language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2606.06474.

Saadat Hasan Khan, Spencer Hong, Jingyu Wu, Kevin Lybarger, Youbing Yin, Erin Babinsky, and Daben Liu. DF-RAG: Query-aware diversity for retrieval-augmented generation. In *Findings of*

the Association for Computational Linguistics: EACL 2026, pp. 2873–2894, 2026. doi: 10.18653/v1/2026.findings-eacl.150. URL https://aclanthology.org/2026.findings-eacl.150/.

Jaemin Kim and Jong Chul Ye. Adaptive guidance for retrieval-augmented masked diffusion models. *arXiv preprint arXiv:2603.17677*, 2026.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pp. 333–342, 2017.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Pengxiang Li, Shilin Yan, Joey Tsai, Renrui Zhang, Ruichuan An, Ziyu Guo, and Xiaowei Gao. Adaptive classifier-free guidance via dynamic low-confidence masking. *arXiv preprint arXiv:2505.20199*, 2025.

Xinze Li, Sen Mei, Zhenghao Liu, Yukun Yan, Shuo Wang, Shi Yu, Zheni Zeng, Hao Chen, Ge Yu, Zhiyuan Liu, Maosong Sun, and Chenyan Xiong. RAG-DDR: Optimizing retrieval-augmented generation using differentiable data rewards. *arXiv preprint arXiv:2410.13509*, 2024. URL https://arxiv.org/abs/2410.13509.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2502.09992.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktaschel, and Sebastian Riedel. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2523–2544, 2021. URL https://aclanthology.org/2021.naacl-main.200/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of NAACL-HLT*, pp. 783–791, 2024.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 10014–10037, 2023.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Adacad: Adaptively decoding to balance conflicts between contextual and parametric knowledge. In *Proceedings of NAACL-HLT*, pp. 11636–11652, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, 2026.

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

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Chuanyue Yu, Jiahui Wang, Yuhan Li, Heng Chang, Ge Lan, Qingyun Sun, Jia Li, Jianxin Li, and Ziwei Zhang. Unlocking the potentials of retrieval-augmented generation for diffusion language models. *arXiv preprint arXiv:2601.11342*, 2026.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3903–3922, 2024.

Zeyang Zhang, Chengwei Liang, Xingyan Chen, Meiqi Gu, Minrui Luo, Jingzhao Zhang, and Tianxing He. Differences in text generated by diffusion and autoregressive language models. *arXiv preprint arXiv:2605.12522*, 2026. URL https://arxiv.org/abs/2605.12522.

A COMPLETE ALGORITHM

Algorithm 1 DIGAA: Information Gathering and Accurate Answering

Input: Question q; retriever $\mathcal{R}$ over corpus $\mathcal{C}$; DLM p with vocabulary $\mathcal{V}$; retrieval depth k; sequence length L; probability threshold τ
Output: Final answer h^s

Stage 1: Information gathering

$D_q \leftarrow \mathcal{R}(q, k)$
Initialize $X = (x_1, \ldots, x_L)$ with mask tokens; $x^0 \leftarrow X$; $r \leftarrow 0$
while x^r contains a mask **do**
  $M^r \leftarrow \{i \in \{1, \ldots, L\} : x_i^r = \texttt{[MASK]}\}$
  $\widetilde{x}^r \leftarrow x^r$
  $\widetilde{x}_i^r \leftarrow \arg\max_{v \in \mathcal{V}} p_i(v \mid x^r, q, D_q)$ for all $i \in M^r$ in parallel
  $D^r \leftarrow \mathcal{R}(q \,\|\, \widetilde{x}^r, k)$
  $y_{r,i} \leftarrow \arg\max_{v \in \mathcal{V}} p_i(v \mid x^r, q, D^r)$ for all $i \in M^r$ in parallel
  $U_\tau^r \leftarrow \{i \in M^r : p_i(y_{r,i} \mid x^r, q, D^r) \geq \tau\}$
  if $U_\tau^r = \varnothing$ **then**
    $U_\tau^r \leftarrow \{\arg\max_{i \in M^r} p_i(y_{r,i} \mid x^r, q, D^r)\}$
  end if
  $x_i \leftarrow y_{r,i}$ for all $i \in U_\tau^r$ in parallel
  $x^{r+1} \leftarrow X$; $r \leftarrow r + 1$
end while

Stage 2: Answer generation

$h^0 \leftarrow (\texttt{[MASK]}, \ldots, \texttt{[MASK]})$ of length L; $s \leftarrow 0$
while h^s contains a mask **do**
  $N^s \leftarrow \{i \in \{1, \ldots, L\} : h_i^s = \texttt{[MASK]}\}$
  $a_{s,i} \leftarrow \arg\max_{v \in \mathcal{V}} p_i(v \mid h^s, q, X)$ for all $i \in N^s$ in parallel
  $N_\tau^s \leftarrow \{i \in N^s : p_i(a_{s,i} \mid h^s, q, X) \geq \tau\}$
  if $N_\tau^s = \varnothing$ **then**
    $N_\tau^s \leftarrow \{\arg\max_{i \in N^s} p_i(a_{s,i} \mid h^s, q, X)\}$
  end if
  $h^{s+1} \leftarrow h^s$
  $h_i^{s+1} \leftarrow a_{s,i}$ for all $i \in N_\tau^s$ in parallel
  $s \leftarrow s + 1$
end while
return h^s

B DATASETS AND EXTERNAL CORPORA

Table 5: Evaluation splits and retrieval corpora used in the main experiments.

Benchmark	Questions	External corpus (reference for the setting)
NQ (KILT dev)	2,837	MS MARCO 2.1 segmented documents (Li et al., 2024)
ZsRE (KILT dev)	3,724	KILT Wikipedia snapshot (Petroni et al., 2021)
2WikiMultiHopQA (dev)	12,576	Dev gold and distractor passage union (Trivedi et al., 2023)
MuSiQue-Ans v1.0 (dev)	2,417	Dev gold and distractor passage union (Trivedi et al., 2023)
HotpotQA (dev)	7,405	Official full-wiki index (Yang et al., 2018)
SynthWorlds (QA-RM)	1,200	Official QA-RM documents (6,290) (Gu et al., 2025)

Natural Questions. We use the KILT development version of NQ (Kwiatkowski et al., 2019; Petroni et al., 2021), obtained through ARAM's `nq_dev_psg.jsonl` release (Kim & Ye, 2026). Its IDs, questions, and accepted answers match `nq-dev-kilt.jsonl`. We evaluate open-domain short-answer generation, retrieving from MS MARCO 2.1 segmented documents following RAG-DDR (Bajaj et al., 2016; Li et al., 2024). The annotated Wikipedia page is not supplied, and the original NQ long-answer span-selection task is not evaluated.

ZsRE. We use KILT's open-domain slot-filling version of ZsRE (Levy et al., 2017; Petroni et al., 2021), with all examples in `structured_zeroshot-dev-kilt.jsonl`. The first question in `meta.template_questions` is the input, and the accepted entities or values are the reference answers. Retrieval uses the KILT Wikipedia snapshot. This evaluates slot filling through question answering, without supplying the original annotated sentence or predicting relation labels.

2WikiMultiHopQA and MuSiQue: corpus construction. For both benchmarks, we pool and deduplicate the supporting and distractor passages attached to the evaluated development questions. Each resulting corpus is shared across questions, replacing their predefined passage bundles. This follows IRCoT's pooling approach (Trivedi et al., 2023).

2WikiMultiHopQA. We use IRCoT's processed `2wikimultihopqa/dev.jsonl`, derived from the official `dev.json` release (Ho et al., 2020; Trivedi et al., 2023). All four question types are included: 5,236 compositional, 3,040 comparison, 2,751 bridge-comparison, and 1,549 inference questions. The pooled corpus contains 56,660 passages.

MuSiQue. We use MuSiQue-Ans v1.0 through IRCoT's `musique/dev.jsonl` (Trivedi et al., 2022; 2023). Its IDs and questions match `musique_ans_v1.0_dev.jsonl`; all questions are answerable, including 1,252 two-hop, 760 three-hop, and 405 four-hop questions. The pooled corpus contains 21,098 passages. Throughout the paper, MuSiQue denotes this retrieval-based multi-hop QA setting, rather than the answerability-detection task in MuSiQue-Full.

HotpotQA. We use IRCoT's `hotpotqa/dev.jsonl`, containing 5,918 bridge and 1,487 comparison questions (Yang et al., 2018; Trivedi et al., 2023).

SynthWorlds-RM. We use the official `datasets/mhqa/qa-rm.json` release (Gu et al., 2025), including all six reasoning motifs A-F with 200 questions each. QA-RM uses real-world entity names in a constructed corpus derived from Wikidata and includes two- to four-hop reasoning structures. The released `query` is the input, its accepted gold answers are the references, and retrieval covers all 6,290 documents.

C Implementation Details

C.1 Decoding Settings

Following ARAM (Kim & Ye, 2026), all methods use temperature zero, a length budget of 32 tokens per sequence, and at most 32 denoising steps. Both stages of DIGAA follow these settings.

C.2 Baseline Implementation Details

Official Direct. We use the backbone's released no-retrieval prompt and decoder (Ye et al., 2025; Nie et al., 2025).

Standard RAG. We condition generation on the initial retrieved passages and keep this document set fixed throughout decoding (Lewis et al., 2020; Kim & Ye, 2026).

CAD-MDM. We follow the settings of ARAM (Kim & Ye, 2026). At each unfinished position, we adjust the logits using $\ell_{\text{prior}} + 2(\ell_{\text{cond}} - \ell_{\text{prior}})$ (Shi et al., 2024; Kim & Ye, 2026).

AdaCAD-MDM. We follow the settings of ARAM (Kim & Ye, 2026). We compute token-level Jensen-Shannon divergence between context-conditioned and prior distributions and use it as the adaptive contrast weight independently at every unfinished position (Wang et al., 2025; Kim & Ye, 2026).

COIECD-MDM. We follow the settings of ARAM (Kim & Ye, 2026). We use $\alpha = 1$ and the published contextual information-entropy constraint with threshold 0.25, vectorized over DLM positions (Yuan et al., 2024; Kim & Ye, 2026).

SPREAD. SPREAD changes position selection from the last-layer hidden states and query representation without altering pre-softmax token logits (Yu et al., 2026).

A-CFG. We remask the lowest-confidence proportion $\rho = 0.7$ to construct the second input and apply a guidance multiplier of 1.5, matching the setting used in ARAM (Li et al., 2025; Kim & Ye, 2026).

ARAM. We compute the symmetric KL signal and conditional-entropy noise, set $\beta = 0.1$, cap the guidance scale at 1.0, and use 10^{-8} for numerical stability (Kim & Ye, 2026).

SARDI. We fix the query threshold at zero and the commitment threshold at 0.90. If no masked position reaches the latter, the most probable predicted token is committed. The complete provisional response is appended to the question for retrieval, and the retrieved passages replace the preceding set (Jünger et al., 2026).

C.3 Decoding Prompts

The prompts for the baselines and the information-gathering stage of DIGAA follow ARAM (Kim & Ye, 2026). The complete prompt templates are provided in Appendix B, Figures 5 and 6 of Kim & Ye (2026).

The answer-generation stage of DIGAA performs no further retrieval and uses the following prompt:

> Based on the knowledge collection, return the most accurate answer that directly answers the question. Do not add any prefix or explanation.
>
> **Question:** {question}
> **Knowledge collection:** {knowledge_collection}
> **Answer:**

C.4 SAMPLE-REVIEW PROMPT

Below we provide a prompt for the GPT-assisted sample review in Section 4.2.

> Review the supplied examples. For each example, use the question to determine what is being asked, and compare the knowledge collection produced by the information-gathering stage with the final answer produced by the answer-generation stage.
>
> First, describe the textual changes in each example before and after the answer-generation stage. Group the observed changes into categories and report the proportion of each category. Return a brief record for each example containing its ID, the observed change, supporting text, and the category derived from the analysis.
>
> **Examples:** {examples}
>
> Each example contains the following fields:
> **Example ID:** {example_id}
> **Question:** {question}
> **Knowledge collection:** {knowledge_collection}
> **Final answer:** {final_answer}
> **Reference answers:** {reference_answers}
> **Change:** {change_type}

D ABLATION OF RETRIEVAL DEPTH

Table 6: Results of DIGAA ($\tau = 0.90$) and nine baselines across six factual QA benchmarks. Bold marks the best result for each model, benchmark, and metric. Underlining marks the best baseline. Retrieval-based methods use the sparse retriever BM25 to retrieve the top 3 documents. DIGAA outperforms all baselines in EM and token F1 on every model-benchmark pair.

Method	NQ		ZsRE		2Wiki		MuSiQue		HotpotQA		Synth-RM	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream	14.9	24.8	4.4	16.4	22.9	26.2	2.4	9.6	15.1	23.8	7.7	12.7
+ RAG	28.4	39.5	49.7	60.4	27.0	31.7	12.5	22.6	28.9	39.5	16.6	23.7
+ CAD	23.0	34.7	37.5	54.3	10.0	15.7	9.9	20.0	21.7	32.3	11.7	19.0
+ ADACAD	27.2	38.3	46.4	59.2	26.8	31.7	13.0	22.7	28.2	38.8	16.3	23.2
+ COIECD	26.2	36.6	45.2	57.9	20.8	25.9	11.2	21.2	24.8	35.4	14.1	21.3
+ SPREAD	28.2	38.9	46.4	57.5	25.0	29.7	11.4	21.6	28.1	38.5	16.6	23.3
+ A-CFG	26.6	38.2	41.4	59.3	25.6	30.5	13.2	24.4	27.0	38.7	15.4	22.9
+ ARAM	29.1	40.1	49.3	60.1	26.3	30.8	11.3	20.8	28.0	38.6	16.0	22.2
+ SARDI	31.4	43.6	51.7	61.8	28.4	33.3	15.9	26.5	32.4	43.2	17.6	24.6
+ DIGAA	**33.5**	**45.6**	**53.7**	**64.8**	**35.3**	**40.4**	**19.0**	**29.3**	**36.2**	**47.7**	**21.8**	**29.5**
LLaDA	10.5	16.7	5.0	11.3	22.0	23.4	1.6	5.5	14.8	20.8	8.0	10.8
+ RAG	21.9	32.3	22.1	38.0	19.7	24.0	9.4	16.8	25.1	33.6	8.2	13.6
+ CAD	15.8	25.2	11.9	28.1	6.5	10.5	5.1	12.1	14.2	23.3	5.1	11.1
+ ADACAD	21.2	31.3	22.6	37.9	18.7	22.8	8.5	16.7	24.0	32.6	7.5	13.0
+ COIECD	17.2	26.9	14.0	29.2	9.2	14.0	6.4	14.2	20.0	28.3	5.8	11.7
+ SPREAD	22.3	32.0	27.8	42.5	22.4	27.4	10.8	19.1	25.7	34.9	10.4	15.8
+ A-CFG	20.7	31.0	21.5	39.4	18.6	23.8	9.1	17.1	23.0	32.2	8.4	14.2
+ ARAM	24.2	34.4	42.0	53.0	22.7	27.0	10.6	18.0	27.0	36.1	13.4	19.2
+ SARDI	23.8	34.8	23.2	40.1	22.0	26.8	14.4	23.0	26.3	36.2	9.0	15.5
+ DIGAA	**28.2**	**39.6**	**46.0**	**59.3**	**35.8**	**43.8**	**18.2**	**29.5**	**32.8**	**44.6**	**19.9**	**27.3**

Table 7: Results of DIGAA ($\tau = 0.90$) and nine baselines across six factual QA benchmarks. Bold marks the best result for each model, benchmark, and metric. Underlining marks the best baseline. Retrieval-based methods use the sparse retriever BM25 to retrieve the top 7 documents. DIGAA outperforms all baselines in EM and token F1 on every model-benchmark pair.

Method	NQ		ZsRE		2Wiki		MuSiQue		HotpotQA		Synth-RM	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream	14.9	24.8	4.2	16.1	23.3	26.6	2.7	9.9	14.8	23.9	7.6	12.6
+ RAG	30.1	42.7	48.0	59.0	26.0	30.5	14.8	24.8	30.3	41.0	17.3	24.3
+ CAD	22.3	34.8	34.0	52.5	11.3	16.6	10.1	19.8	21.5	32.9	9.5	16.9
+ ADACAD	28.0	40.9	45.4	58.1	24.5	29.1	13.8	23.8	28.9	39.4	15.9	22.2
+ COIECD	25.8	38.2	42.5	56.8	20.0	25.5	12.1	22.5	26.5	36.6	13.3	19.7
+ SPREAD	30.4	41.9	42.9	54.9	26.2	30.9	13.1	23.0	29.0	39.1	17.6	24.5
+ A-CFG	27.0	40.7	38.5	57.5	25.9	30.9	12.8	24.3	28.8	39.5	15.8	22.1
+ ARAM	31.6	43.4	43.8	56.4	25.2	29.8	12.2	21.8	28.8	39.9	16.4	23.1
+ SARDI	31.8	44.8	50.6	60.7	26.4	32.0	17.8	28.2	32.1	43.5	18.6	25.3
+ DIGAA	**35.0**	**47.9**	**54.8**	**67.2**	**35.3**	**40.7**	**20.9**	**31.8**	**38.7**	**50.6**	**22.8**	**31.1**
LLaDA	10.4	16.7	5.1	11.4	21.6	22.9	1.6	5.5	14.5	20.6	8.2	11.0
+ RAG	18.6	29.3	8.3	27.6	9.0	12.9	7.7	13.8	20.7	28.6	6.0	11.9
+ CAD	15.8	26.1	8.9	24.7	7.2	11.7	4.9	11.5	12.3	20.5	5.3	11.2
+ ADACAD	17.7	28.5	8.4	25.9	8.5	12.5	6.4	13.4	19.6	27.7	6.1	12.2
+ COIECD	16.7	26.9	9.6	25.7	6.9	11.2	5.7	12.2	14.6	21.8	6.1	12.6
+ SPREAD	20.3	30.6	9.2	27.7	13.7	17.8	7.6	14.9	23.7	31.9	5.4	11.3
+ A-CFG	18.8	29.7	9.8	29.2	12.6	17.5	8.1	14.8	20.3	29.1	5.5	12.1
+ ARAM	19.7	30.3	13.1	30.4	16.4	20.3	10.0	17.5	25.7	35.0	5.9	11.9
+ SARDI	21.3	33.4	8.9	28.7	12.1	17.2	13.3	21.2	22.3	31.6	7.2	14.1
+ DIGAA	**30.4**	**43.9**	**47.8**	**62.5**	**36.5**	**48.7**	**27.2**	**40.9**	**38.1**	**50.9**	**22.5**	**30.6**

E DENSE RETRIEVAL WITH CONTRIEVER-MSMARCO

Table 8: Results of DIGAA ($\tau = 0.90$) and nine baselines across six factual QA benchmarks. Bold marks the best result for each model, benchmark, and metric. Underlining marks the best baseline. Retrieval-based methods use the dense retriever Contriever-MSMARCO to retrieve the top 5 documents. DIGAA outperforms all baselines in EM and token F1 on every model-benchmark pair.

Method	NQ		ZsRE		2Wiki		MuSiQue		HotpotQA		Synth-RM	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream	14.8	24.7	4.6	16.5	23.6	26.9	2.4	9.6	15.1	23.8	7.5	12.5
+ RAG	34.9	47.8	55.5	65.8	26.4	30.7	18.0	29.5	32.5	43.1	16.5	23.2
+ CAD	29.4	43.2	42.4	60.6	11.9	17.6	14.8	26.2	23.5	34.5	12.4	19.9
+ ADACAD	34.0	47.2	50.4	63.7	25.8	30.4	17.5	28.5	31.0	41.4	16.3	23.2
+ COIECD	31.7	45.3	49.9	64.2	19.0	24.5	15.7	27.0	27.6	38.3	14.9	21.7
+ SPREAD	35.5	48.0	49.1	62.9	25.6	30.4	17.4	28.1	31.7	41.6	16.5	23.2
+ A-CFG	32.0	46.0	43.9	64.2	24.3	29.2	17.9	30.3	31.0	42.4	15.5	22.1
+ ARAM	34.9	47.6	55.8	65.6	25.3	30.4	16.5	27.4	31.3	41.7	16.5	22.9
+ SARDI	36.6	49.6	60.9	69.8	28.3	33.3	21.2	33.1	34.5	45.7	17.6	25.0
+ DIGAA	**37.5**	**51.5**	**64.4**	**74.3**	**36.1**	**43.0**	**25.6**	**37.5**	**38.8**	**50.6**	**22.2**	**29.9**
LLaDA	9.9	16.1	5.0	11.3	22.0	23.4	1.6	5.5	14.5	20.6	8.1	11.0
+ RAG	24.9	37.5	16.3	36.0	11.4	15.8	10.2	18.5	22.4	30.5	5.5	11.4
+ CAD	19.5	32.0	10.2	28.2	7.3	11.7	7.2	15.9	13.5	20.9	5.4	10.8
+ ADACAD	23.0	36.4	15.9	34.2	9.8	14.3	9.2	18.4	20.9	29.1	5.0	11.0
+ COIECD	20.8	32.9	11.9	29.3	8.2	12.5	7.9	16.2	15.2	22.9	5.7	11.9
+ SPREAD	23.6	36.3	19.1	37.9	14.5	19.7	11.7	20.7	26.2	34.8	7.2	12.6
+ A-CFG	25.8	38.5	17.4	37.8	14.0	18.9	10.5	18.8	20.7	29.5	6.1	12.3
+ ARAM	28.2	40.2	33.2	47.2	18.0	21.9	13.7	22.8	29.0	37.8	9.1	14.7
+ SARDI	23.9	37.8	16.3	37.1	13.4	18.6	15.1	24.1	25.1	33.5	5.7	12.5
+ DIGAA	**31.8**	**46.6**	**54.0**	**70.3**	**36.6**	**49.1**	**27.9**	**42.5**	**36.6**	**49.4**	**22.1**	**30.0**

F Computation and Timing

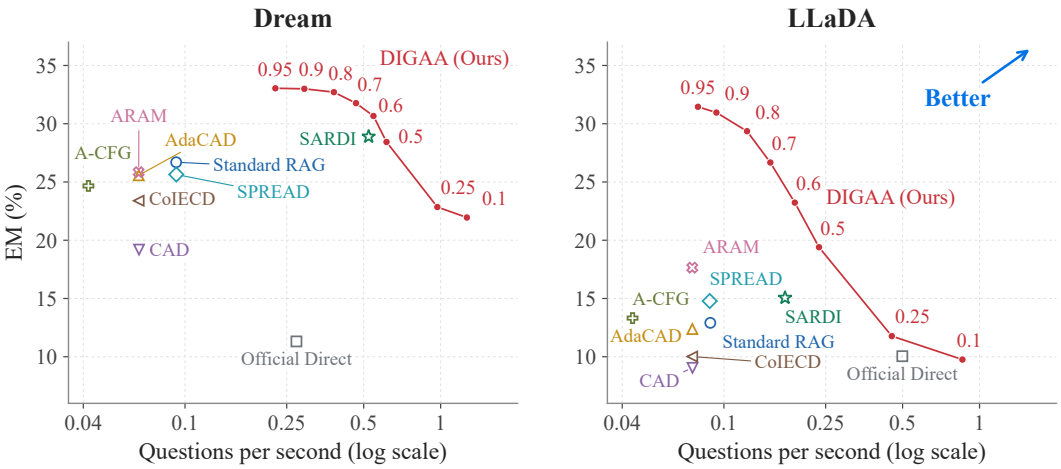


Figure 5: Trade-offs between quality and latency on Dream (left) and LLaDA (right). EM is averaged over six benchmarks. Speed is the reciprocal of mean latency, measured as end-to-end wall-clock time on a single RTX 3090 GPU with batch size 1. Red labels indicate the confidence threshold τ used by DIGAA. The horizontal axis is logarithmic. DIGAA leads in the trade-off between answer quality and inference latency.

Table 9: Average computation and speed over six factual QA benchmarks with BM25 top-5 retrieval and 32-token sequence budgets. NFE denotes the average number of forward passes per question, and Retrievals denotes the average number of document-retrieval calls per question. Speed is the reciprocal of mean end-to-end wall-clock latency. All methods process questions individually on a single RTX 3090 GPU without batching (batch size 1).

Method	**Dream**			**LLaDA**		
	NFE	Retrievals	Questions per second	NFE	Retrievals	Questions per second
Official Direct	32.0	0.0	0.273	32.0	0.0	0.499
Standard RAG	32.0	1.0	0.092	32.0	1.0	0.088
CAD	64.0	1.0	0.066	64.0	1.0	0.075
AdaCAD	64.0	1.0	0.066	64.0	1.0	0.075
COIECD	64.0	1.0	0.066	64.0	1.0	0.075
SPREAD	33.0	1.0	0.092	33.0	1.0	0.088
A-CFG	64.0	1.0	0.042	64.0	1.0	0.044
ARAM	64.0	1.0	0.066	64.0	1.0	0.075
SARDI	5.1	6.1	0.523	13.6	14.6	0.173
DIGAA ($\tau = 0.10$)	2.8	1.9	1.267	3.4	2.3	0.857
DIGAA ($\tau = 0.25$)	3.6	2.3	0.971	5.9	3.3	0.454
DIGAA ($\tau = 0.50$)	5.3	2.8	0.614	11.4	5.3	0.235
DIGAA ($\tau = 0.60$)	6.3	3.0	0.546	14.4	6.3	0.189
DIGAA ($\tau = 0.70$)	7.3	3.1	0.467	17.5	7.2	0.152
DIGAA ($\tau = 0.80$)	8.7	3.4	0.382	22.2	8.4	0.123
DIGAA ($\tau = 0.90$)	11.4	3.7	0.293	29.6	9.9	0.093
DIGAA ($\tau = 0.95$)	15.0	4.0	0.226	36.3	10.9	0.079

G COMMITMENT THRESHOLD SENSITIVITY

Table 10: Threshold sensitivity analysis with BM25 top-5 retrieval. The table shows eight threshold settings; bold marks the best displayed value for each model, benchmark, and metric, including ties.

τ	**NQ**		**ZsRE**		**2Wiki**		**MuSiQue**		**HotpotQA**		**Synth-RM**	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Dream												
0.10	26.6	37.6	32.9	47.6	24.0	33.5	7.2	15.5	27.2	38.6	13.8	21.9
0.25	27.9	39.7	33.8	48.7	24.4	34.0	8.1	17.1	28.3	40.0	14.6	22.7
0.50	32.3	45.1	44.1	58.3	30.2	38.2	12.0	22.0	33.0	45.1	19.1	27.0
0.60	34.1	46.6	49.1	62.3	32.6	39.5	12.9	23.5	34.8	46.9	20.3	28.1
0.70	34.7	47.4	51.1	64.3	34.2	40.5	14.1	24.7	36.0	48.1	20.4	28.1
0.80	**35.9**	48.4	52.4	65.4	35.2	40.8	**14.2**	**25.0**	36.8	48.8	21.8	29.5
0.90	35.8	48.7	53.2	**65.9**	36.0	41.4	13.8	24.8	37.0	49.0	22.2	**29.7**
0.95	35.8	**48.8**	**53.3**	**65.9**	**36.1**	**41.5**	13.7	24.9	**37.1**	**49.2**	**22.3**	29.6
LLaDA												
0.10	11.6	26.1	10.5	27.7	11.2	26.3	4.0	11.8	16.2	28.1	5.1	13.8
0.25	14.8	29.9	15.9	34.9	12.2	28.4	4.7	14.1	17.1	29.8	5.9	15.7
0.50	21.1	37.4	28.4	48.1	20.3	36.6	9.6	21.4	25.7	39.3	11.3	21.5
0.60	23.8	39.7	34.1	53.1	25.1	40.1	12.2	24.2	30.0	43.4	14.1	23.8
0.70	27.0	41.9	39.7	57.2	30.3	43.6	13.6	25.6	33.0	46.0	16.5	26.0
0.80	29.0	43.3	43.4	59.8	33.5	45.5	15.7	27.2	35.3	47.7	19.3	27.7
0.90	**30.0**	43.7	45.3	60.8	36.8	47.3	**16.5**	**28.3**	36.7	48.9	20.5	28.3
0.95	29.9	**43.8**	**45.7**	**61.0**	**38.8**	**48.2**	**16.5**	28.2	**37.2**	**49.1**	**20.6**	**28.6**