KEEPING THE READER'S PLACE: WHY REVEALING EVIDENCE ONE STEP AT A TIME HELPS MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A masked diffusion language model rereads its prompt on every forward pass, so what the prompt shows can change while the answer is being written. Revealing one record at a time along a chain of pointers turns a hard multi-hop prompt into a sequence of easy local reads, and it is tempting to credit the gain to a model that has learned where to look. We take the gain apart on tasks whose dependency structure is explicit, holding the model, the items, and the forward passes fixed while varying what is visible when, what the reader was trained on, and whether the order of reads is decided during generation or before it. A LLaDA-8B model fine-tuned to read one record at a time solves chains longer than any it saw in training. The same model fine-tuned on full prompts and handed exactly the same records all at once falls well short, and its errors have a consistent signature: when a value recurs in the chain, it resumes from the wrong occurrence, far more often than chance predicts. Revealing one record at a time removes this bookkeeping from the reader. The order that does so need not come from the model: a text parser computes it from the prompt before generation begins. Two further controls locate what remains. The strength of a static baseline depends on what it was trained on, and when neighboring answers are independent, as on chains of IMDB reviews, classifying each record on its own does at least as well as reading the chain on average. Revealing evidence one step at a time helps a diffusion reader because it keeps the reader's place, and the schedule that does so can be fixed in advance.

1 INTRODUCTION

Figure 1a shows a prompt with sixteen records in shuffled order. Each record names a successor and carries a digit, and the answer is the sequence of digits along the chain that starts at node 8: `4 8 3 3`. A reader given the whole prompt has to find node 8, read its digit, find node 4, and so on, keeping track of where it is while it writes. A masked diffusion language model (Nie et al., 2025) offers another option. It produces its answer over many forward passes, and nothing requires the prompt to be the same on each pass. A controller can show node 8 alone while the first digit is being written, node 4 alone for the second, and so on. Each pass then asks for one local step, and the long-range bookkeeping moves out of the model.

This works very well, and that is what makes it hard to interpret. When a reader scheduled this way beats one that sees the whole prompt, several things have changed at once. The reader sees fewer records. It sees them in a particular order. It has usually been trained on inputs of the same shape. And the order may have been decided during generation, from what the model has written so far—which is what "adaptive retrieval" would mean. A headline accuracy cannot tell these apart, and they imply very different things about what the model has learned.

Following the retriever–reader split of open-domain question answering (Chen et al., 2017), we call the component that decides which records are visible the *selector*, and the model that reads them and writes the answer the *reader*. We separate the explanations above with a small set of controls on tasks whose dependency structure is explicit: digit pointer chains, chains of IMDB reviews linked by synthetic pointers, and alias chains hidden among decoys. Because the structure is explicit, a text parser can follow it, and we can ask what changes when the parser, rather than the model, acts as the

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.

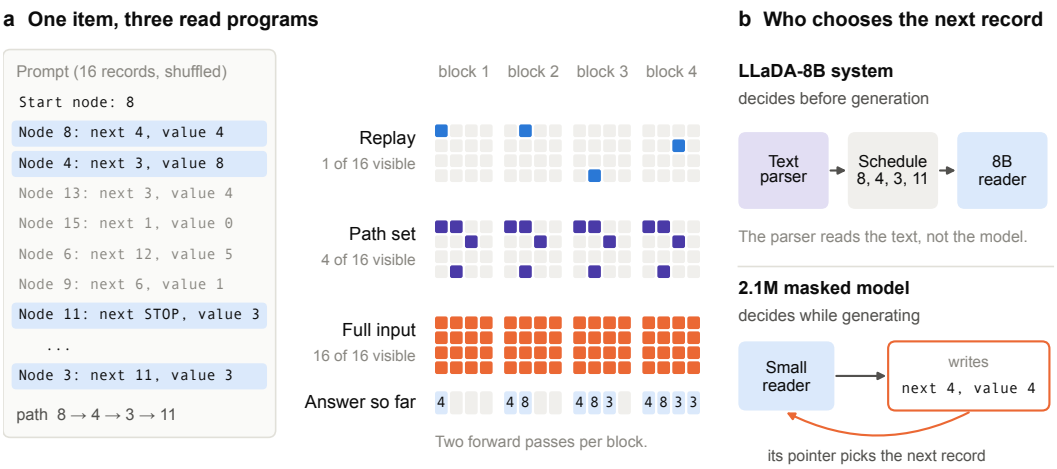


Figure 1: **A read program changes what the reader sees at each pass. (a)** The first digit test item (item seed 5,000,000), with the four path records tinted. Each grid shows the 16 records in prompt order; filled cells are visible during that block. Replay shows one path record per block, path set shows all four throughout, and full input shows all sixteen. All three programs use the same two forward passes per block and commit one digit per block. **(b)** Who picks the records. In the 8B system a text parser computes the whole schedule from the prompt before generation starts. The separate 2.1M model writes a pointer as part of its answer, and that pointer selects its next record.

selector. Within a comparison we hold the pretrained model and fine-tuning recipe, the items, the answer positions, and the number of forward passes fixed, and vary only the *read program* (which records are visible at each pass) or the input the reader was fine-tuned on.

We find that the order matters, and that on these tasks it does not have to come from the model.

- **A one-record reader extrapolates in depth.** LLaDA-8B trained on one record per block at depths 4 and 8 solves every depth-12 chain we test, in all three seeds (Section 4.1). The parser supplies the records; the model supplies a step that it can repeat more times than it was trained to.
- **Static readers fail by losing their place.** The strongest static reader we trained reaches 62.5% at depth 12 given only the path records. Most of its first errors are not random: it continues from another occurrence of the digit it just wrote (Section 4.2). Showing one record per block removes the need to track position, which is where most of these errors arise.
- **The static baseline depends on training.** How well a static reader does depends on how its test input relates to its training input, so comparing the single-record reader with a full-context reader on the full input overstates the benefit of scheduling (Section 4.3).
- **Sequencing adds nothing when labels are independent.** On review chains, classifying each review alone beats sequential reading on average for every trained reader (Section 4.4).
- **The order can be fixed in advance.** The parser reads the prompt, not the model's output, so the whole schedule can be computed before generation and yields the same run by construction; an implementation check confirms this (Section 4.5).

A separate 2.1M-parameter model shows that the step our 8B system leaves to the parser, choosing the next record from generated text, can be learned in miniature; we do not have an 8B model that does it. A MuSiQue diagnostic marks where our setup stops working (Section 4.6).

2 Read Programs

An input consists of an instruction x_0, which is always visible, and records $x_1, \ldots, x_N$. At pass t the reader sees its current response z_t and the records indexed by R_t, and the update may commit only

the response positions in O_t:

$$z_{t+1} = U\big(z_t,\ f_\theta(z_t, x_0, x_{R_t}),\ O_t\big).$$

A *read program* is the sequence $R_1, \ldots, R_T$. We divide the T passes into B blocks of consecutive passes, and block b commits the b-th answer value. Within a task and depth, every program uses the same B, T, and ownership sets O_t, so programs differ only in which records are visible when. The same forward count is not the same cost: a full-input pass processes about 2.7 times as many tokens as a replay pass at depth 8 (Appendix G).

Programs. *Full input* shows every record on every pass. *Path set* shows the records on the chain, all at once, on every pass. *Replay* computes the chain from the prompt before generation and shows its b-th record during block b. *Online parser* produces the same schedule during generation by following the pointer in the record it last showed; it reads the prompt text, never the model's output. *Independent* is used for reviews: it classifies each chain review in its own call and concatenates the labels, so no predicted label ever enters a later input. We fine-tune three readers whose training inputs mirror replay, path set, and full input; we call them the *single-record*, *path-set*, and *full-context* readers. We call a reader evaluated on the path set or the full input a *static reader*, since every record it can use is visible on every pass. Each is evaluated under replay, path set, and full input, and review readers also under independent classification. The online parser is used only for the implementation check below.

A prompt-only schedule can be precomputed. Fix the reader, its random seed, the initial state, and the ownership sets, and assume deterministic computation. If two runs make the same sequence of reads, then by induction over passes they produce the same states and the same output. A controller whose decisions depend only on the prompt text makes the same reads whether it runs before generation or during it, so running it online can add nothing. The argument does not apply to controllers whose next read depends on what the model has written, which is the case our MuSiQue diagnostic probes. For our pointer tasks the equivalence holds by construction. Because the controls rely on it and an implementation error would break it silently, we checked it in the earlier experimental stage (Section 4.5); the final-stage evaluations run the precomputed schedule.

Coverage is not the issue here. A fixed set of k records chosen without looking at the content contains a given depth-D path with probability $\binom{k}{D}/\binom{N}{D}$, and never when $k < D$ (Appendix I). That bound explains failures of small fixed windows. It does not apply to our static programs: the path set and the full input both contain every record the answer needs, so their errors are about how the reader uses the evidence, not about missing it.

3 TASKS, READERS, AND SCORING

Digit chains. Sixteen shuffled records of the form `Node 8: next 4, value 4`; a depth-D chain visits D distinct nodes, and the answer is their D digits in order. We evaluate 128 fresh items per depth for $D \in \{4, 8\}$ and 64 at $D = 12$. Readers are trained only at depths 4 and 8.

Review chains. Each record is an IMDB review (Maas et al., 2011) truncated to 300 characters, followed by a synthetic line `Next review: 3`. The answer is the sentiment of each review along the chain, and the label never appears in the prompt (Table 1). Four unrelated reviews are added to every prompt. Documents come from a cached sample split 839/180/180 into train, development, and test, disjoint by document ID, full-text hash, and visible prefix. Each depth uses 128 chains built from the 180 test documents, so chains share documents.

Alias chains. Assignments such as `VAR v3 = v7` form a chain that ends in a number, hidden among 32 chunks of real review text, with either 0 or 32 decoy assignments. There are 200 items per cell and 800 in total. This is a RULER-style construction (Hsieh et al., 2024), not an official benchmark score. Alias chains are evaluated zero-shot with LLaDA-8B-Instruct, without any of our fine-tuning.

Readers. The 8B readers are LLaDA-8B-Instruct (Nie et al., 2025) fine-tuned with LoRA (Hu et al., 2022) (rank 16, scaling 32, learning rate 10^{-4}, 2,000 steps), each under training seeds 0, 1, and

Table 1: **A review chain, as the reader sees it.** The first review test item (item seed 5,000,000) in the order the chain visits it; each excerpt shows the opening words (at most 23) of the 300 characters the model sees. The single-record reader (training seed 0) answers correctly under replay. Given all four reviews at once, the same reader answers *Negative Negative Negative Negative*.

Record	Literal excerpt (shortened for display)	Next record
2	I'm a Petty Officer 1st Class (E-6) and have been in the USCG for 6 years and feel that this movie strongly represents …	1
1	Rita Hayworth is right there where she should be - as a "Cover Girl" in this 1944 Technicolor film also starring Gene Kelly, …	3
3	Better than average World War II-era "who-dun-it" featuring Warner Baxter as a former gangster who suffered amnesia and has been reborn as a …	4
4	The film made no sense to me whatsoever. …	STOP

Start: review 2. **Gold:** Positive, Positive, Positive, Negative. **Replay output:** Positive, Positive, Positive, Negative.

2. Digit chains use $B = D$ blocks and $T = 2D$ passes, and records that are not visible are deleted from the prompt rather than masked. Independent classification spends the same $2D$ passes on D one-label calls. The separate 2.1M model is a six-layer masked Transformer (width 192) trained from scratch for 6,000 steps; it writes both a pointer and a value for each hop and follows its own pointers at test time. It shows that the capability can be learned, and is not compute-matched to the 8B system.

Scoring. We report exact match (EM) on the whole answer. All training seeds are reported, including runs that failed, and no checkpoint was selected on test data. Means are over the three trained checkpoints; three seeds describe run-to-run variation, not independent samples, so we show every seed where it matters. Every final-stage 8B number is recomputed by one script from 23,764 saved prediction rows, each re-scored from its raw output; the script finds no scoring discrepancies and no duplicate items. Alias and small-model counts come from the saved aggregate records of an earlier stage (Appendix A).

4 Results

4.1 A one-record reader extrapolates in depth

Figure 2a compares the single-record reader under replay with the strongest static reader we have: the full-context reader given the path set. The single-record reader scores 100% EM at depths 4 and 8 in every seed, and 100% on all 64 depth-12 chains in every seed, although it never trained on a chain longer than 8. The step it learned—read the visible record, write its digit—can be repeated more often than it was during training.

The static reader reaches 99.2%, 88.3%, and 62.5% at depths 4, 8, and 12. At depth 12 the difference is 37.5 points (item bootstrap interval 29.7–45.3, conditional on the three checkpoints; Appendix E). This compares one-record reading and its matching training against the best static reader we have; it does not isolate the schedule from training, since the single-record reader given the same path set scores 0% at depth 12 (Appendix Table 4). We credit the result to the reader and parser together: the parser chooses every record, and none of these runs tests whether an 8B model could choose them.

The small model shows that the step our 8B system leaves to the parser can be learned in miniature. Following pointers it writes itself, it reaches 98.96% mean EM at depth 4 and 82.81% at depth 8. The depth-8 seeds score 100%, 74.2%, and 74.2%, so the behavior is learnable but not yet reliable across seeds.

4.2 How static readers fail: they lose their place

Figure 2b shows where the static reader goes wrong. Scored digit by digit, the full-context reader on the path set is perfect at the first hop and declines steadily to 67% at hop 10, recovering to 81% at

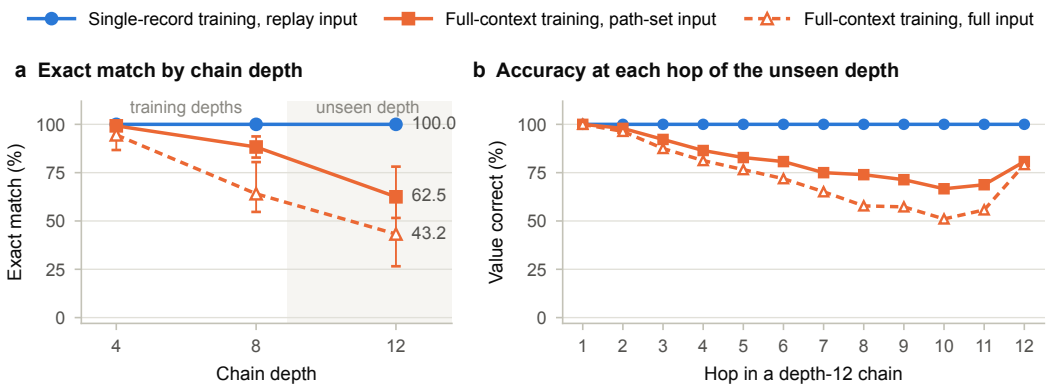


Figure 2: **The single-record reader holds at every depth; the static readers lose accuracy along the chain. (a)** EM for LLaDA-8B readers trained at depths 4 and 8 only. Points are means over three seeds; whiskers span the seeds. **(b)** The same depth-12 predictions scored digit by digit. Accuracy of the full-context reader falls with hop position through hop 10 under both path-set and full input, then partly recovers at the last hop. Items: 128 per depth at 4 and 8, 64 at 12, shared across readers.

the final hop. The errors accumulate along the chain, which is what one would expect if the reader sometimes loses track of where it is.

Figure 3 shows the first depth-12 item it gets wrong. The digit 5 appears at hops 3, 5, and 8. After writing the 5 at hop 8, the reader continues as if it had just written the one at hop 3, and its last four digits repeat hops 4–7. This is not a one-off. Take every wrong answer whose first error is after hop 1, and ask whether the wrong digit is the one that follows some *other* occurrence of the previous digit in the gold sequence (Appendix D gives the definitions). At depth 12 that is true for 62 of 72 errors (86%). A uniformly random wrong digit would match 17% of the time, and a digit copied from a random other path record 30%. The rates are 80% at depth 8 (chance 12% and 34%), and 72% for the same reader on the full input (16% and 25%). The reader behaves as though it locates its position by the last value it wrote, not by the node that value came from. Most of these jumps go forward to a later occurrence; the example happens to go back. Equal digits make position ambiguous, and the ambiguity grows with depth.

The same pattern holds within chains. Every depth-12 chain repeats some digit, but each also has hops that follow a digit occurring only once. While the reader is still on track, its next digit is wrong 6.6% of the time after a repeated digit and 0.2% after a unique one; 71 of its first errors follow a repeated digit and 1 follows a unique one. Stratified by hop position, the error rate after a repeated digit is 42 times higher (11 at depth 8, and 4 on the full input, where other errors also occur). This compares hops inside the same chains, but it is still observational rather than an intervention on the input.

A single-record reader is protected from this error: at any pass there is only one record to read, so there is nothing to confuse its position with. The schedule does the bookkeeping that the static reader gets wrong, which is a more specific account of the 37.5-point gap than "adaptive reading helps."

4.3 The static baseline depends on training

The comparison above uses the strongest static reader we have, and choosing it matters. Figure 4a crosses the three training inputs with the three evaluation programs. No reader does well on every program: the single-record reader handles only one record at a time, and the full-context reader handles everything except one record at a time. At depth 8 the single-record reader scores 100% under replay and 0% on the full input. The full-context reader scores 64.1% on the full input, 88.3% on the path set, and 0.3% under replay. Measured against the full-context reader on the full input, the single-record reader is 35.9 points ahead; measured against the same reader on the path set, it is 11.7 points ahead. The 24-point difference between those two comparisons comes from the baseline's input, not from scheduling.

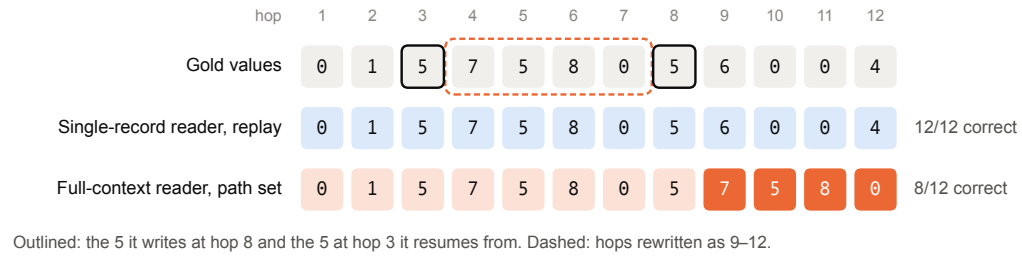


Figure 3: **A static-reader error, in detail.** The first depth-12 item (item seed 6,000,000) that the full-context reader (training seed 0) gets wrong on the path set. Its first eight digits are correct. The last four repeat hops 4–7, the digits that followed the 5 at hop 3. The single-record reader under replay answers the same item correctly.

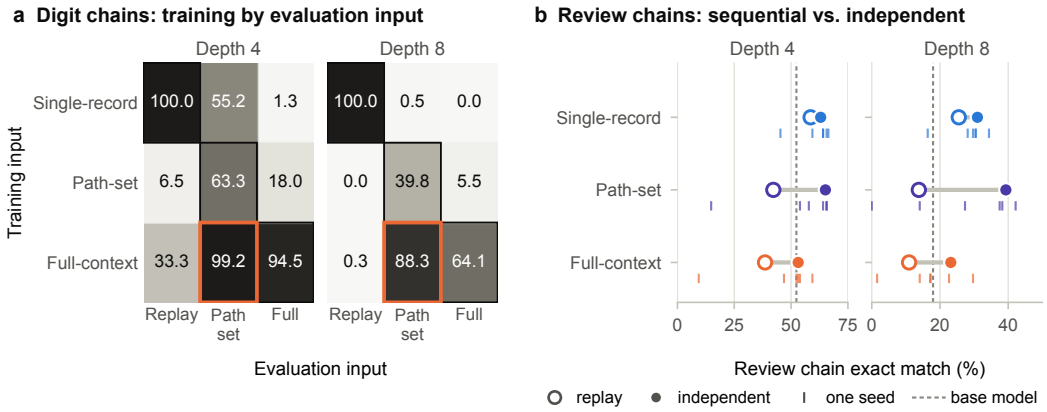


Figure 4: **Static baselines, done fairly.** **(a)** Digit-chain EM (%), mean of three seeds, for each training input (rows) and evaluation program (columns). Outlined cells are matched training and evaluation; the orange cell is the static baseline used in Figure 2. **(b)** Review-chain EM for the same checkpoints read in sequence (replay, open circles) or one review at a time (independent, filled circles). Ticks are single seeds; the dashed line is LLaDA-8B-Instruct without our fine-tuning, used as an independent classifier. Exact values with standard deviations are in Appendix C.

Training directly on path sets is unstable rather than simply weak. Its depth-8 EM across seeds is 0%, 29.7%, and 89.8%, and the seed-0 adapter writes only the digit 6, on every test item under every program. We report these runs and draw no conclusion about the ceiling of path-set training; they measure the variance of our recipe. The full-context reader on the path set is the more dependable static control, and it is the one we use.

4.4 On review chains, the chain adds nothing

Review chains ask a different question. The labels of neighboring reviews are independent, so an earlier label tells the reader nothing about the next one. If sequential reading helped here, it would have to be through something other than information, such as a steadier output format.

Figure 4b shows that it does not help. For every trained reader and both depths, classifying each review in its own call beats replay on average over seeds, though not in every seed. At depth 8 the single-record reader goes from 25.5% to 31.0% EM, and the path-set reader from 13.8% to 39.3%, against 18.0% for LLaDA-8B-Instruct without our fine-tuning, classifying the same reviews independently. The same path-set readers given all chain reviews at once score 0.8%. Fine-tuning did not make a better sentiment classifier. The base model leaves 7% of labels empty at depth 8, but on the reviews it does label it is 89% accurate, while the three trained readers score 87–91% on those same reviews; fine-tuning mainly taught it to emit a label every time. It did not produce a model that

reads a chain. Given several reviews at once, the single-record reader writes the same label for every review in 93% of depth-4 chains, as in Table 1, while only 18% of gold chains are constant.

Per-review accuracy accounts for nearly all of the chain result. If each of D labels is right with probability a, independently, chain EM should be about a^D. Across the 14 replay and independent cells (three trained readers and the base model, both depths), the observed EM is within 9.9 points of a^D, and within 2.8 points on average (Appendix D). The single-record reader's per-review accuracy at depth 8 is 85.3% under replay and 87.3% independently; its chain EM is low because $0.853^8 \approx 0.28$, not because the chain is hard to follow. Independent classification changes three things at once—it drops previous labels, changes the instruction, and splits the context—so we read it as a bound on what sequencing contributes, not as a clean ablation of error propagation.

4.5 What the selector does on its own

Because the parser reads the prompt and never the model's output, Section 2 guarantees that computing the schedule before generation gives the same run as computing it online. We checked the implementation against this guarantee in the earlier experimental stage: read traces and output strings agree in all 5,792 matched online/replay pairs, covering alias, digit, and review items across that stage's checkpoints and seeds (Appendix F). The check guards against implementation errors; it is not an independent empirical finding. The substantive point is the one the guarantee rests on: for pointers that can be read off the prompt, choosing the next record needs the text and not the model.

Alias chains show how much of the work selection can do on its own (Appendix Table 9). The parser resolves the chain without a forward pass, and the parser-only answer is correct on all 800 items; so is the model shown only the final assignment. Those scores measure the parser. The model matters once surface patterns stop picking out the chain: at depth 8 with 32 decoys, a filter that keeps every assignment line drops to 54% and the full input to 14%, while LLaDA-8B-Instruct given the path set, zero-shot, scores 99.5%.

4.6 Where it breaks: natural-language multi-hop questions

On MuSiQue-Ans (Trivedi et al., 2022) there are no pointers to parse. We run LLaDA-8B-Instruct zero-shot, without our fine-tuning, on 96 questions from the v1.0 development set (32 each at 2, 3, and 4 hops), with a protocol in which the model writes three short notes before answering and a BM25 retriever (Robertson & Zaragoza, 2009) conditioned on the question plus the notes chooses what it sees next. The feedback loop runs: in 93 of 96 questions the notes change the retrieved paragraphs (82 change which paragraphs are retrieved; the rest change only their order). It does not produce answers. Strict EM is 0/96 for dynamic retrieval, static retrieval, and the full input, and 10/96, all at two hops, when the annotated supporting paragraphs are provided.

Table 2 shows one question in detail. The model names Bombardier under every read program. Two answers stop at the subsidiary. The full-input answer ends with the line "Answer: Bombardier Inc.", which matches the gold answer, but it scores zero because strict EM compares the whole output; only the oracle output matches exactly. A permissive containment check counts 18–37 of 96 answers as containing the answer, but it also accepts partial and ambiguous matches, so we do not treat it as accuracy. The honest summary is that this zero-shot protocol and output format do not transfer to natural-language multi-hop questions, and that this diagnostic—separate note-generation calls rather than lookahead within one diffusion trajectory—does not test SARDI (Jünger et al., 2026) or state-conditioned retrieval in general.

5 Related work

Diffusion language models. LLaDA shows that a pretrained masked diffusion model can be competitive with autoregressive models of similar size (Nie et al., 2025), and block diffusion interpolates between the two by generating in blocks (Arriola et al., 2025). Our blocks are a different object: they schedule what the *input* shows, while the ownership of output positions stays fixed, and we use the same schedule to fine-tune readers and to evaluate them.

Table 2: **A MuSiQue question the model nearly answers.** "Which company owns the manufacturer of Learjet 60?" Gold: *Bombardier Inc.* Final answers are verbatim. EM is strict normalized match; "Contains" is the permissive check. Full counts over 96 questions are in Appendix H.

Read program	Final answer (verbatim)	EM	Contains
Dynamic retrieval	Bombardier Aerospace	0	1
Static retrieval (top 4)	Bombardier Aerospace owns the manufacturer of Learjet 60.	0	1
Full	The manufacturer of Learjet 60, Bombardier Aerospace, is owned by Bombardier Inc. Answer: Bombardier Inc.	0	1
Oracle set	Bombardier Inc.	1	1

Retrieval interleaved with generation. IRCoT alternates retrieval with chain-of-thought steps (Trivedi et al., 2023), FLARE retrieves when a look-ahead sentence it generates contains low-confidence tokens (Jiang et al., 2023), and SARDI drives retrieval from a diffusion model's lookahead predictions (Jünger et al., 2026). These systems motivate our question; we do not reimplement or compete with them. Our 8B experiments use pointers that can be read from the prompt, which is exactly what makes a precomputed schedule possible. The contribution is a set of controls that tells the two regimes apart, not a claim of priority for iterative retrieval.

Controlled long-context evaluation. RULER and BABILong build long-context tasks with known dependency structure (Hsieh et al., 2024; Kuratov et al., 2024). We borrow the method, not the benchmarks: our generators are our own and we report no scores on either suite.

6 Discussion

What to credit, and to whom. In our experiments the benefit of read scheduling divides cleanly. When records are shown matters: one record per block spares the reader the job of tracking its position, and we can watch a static reader fail at exactly that job. The model does not need to decide the order: a parser that never consults the model chooses every record, and it can do so before generation. And a reader can fail completely on inputs far from what it was trained on: the single-record reader scores 0% on the full input. What our tasks leave open is whether a model that chooses its own next record would help where no parser exists. We suggest three habits for evaluating such systems: say what the selector is allowed to see, compare against a schedule computed in advance and not only against a fixed context, and check whether independent predictions are already enough. A reader that does badly on inputs unlike its training data is evidence about its training data.

Limitations. The dependency structure in our tasks is explicit and synthetic; natural text may not admit a comparable parser, and the MuSiQue diagnostic suggests it does not admit ours. The main results use one 8B model, with no autoregressive baseline and no second model family, so we cannot say the behavior is specific to diffusion. Equal optimizer steps are not equal training compute, and our timings on shared GPUs do not support a speed claim. Review chains reuse 180 test documents, which rules out chain-level confidence intervals, and truncation to 300 characters leaves some labels undetermined by the visible text (Appendix J). We do not know how much IMDB text the base model saw in pretraining. The small model masks unread records while the 8B system deletes them, so the two systems are not directly comparable.

7 Conclusion

Showing a diffusion reader one record at a time, and training it for that, yields a reader that extrapolates from depth 8 to depth 12 without error. Against the strongest static reader we trained, fine-tuned on full prompts and given the same records all at once, the advantage is 37.5 points at depth 12; this compares schedules and training together. Most of the static reader's first errors are one recognizable mistake: it loses its place after a repeated digit. The order that prevents this mistake comes from a parser and can be computed before generation. Revealing evidence one record at a time is useful because it keeps the reader's place; until a control shows otherwise, the credit belongs to that order, to whatever chooses it, and to the reader's training, not to the model steering its own reads.

REPRODUCIBILITY STATEMENT

The supplement contains the analysis code, generated tables and figures, anonymized prediction records, and experiment configurations. Appendix A gives data processing, training, scoring, and provenance. Trained weights are not included. One script rebuilds every final-stage table, figure, and in-text statistic from the frozen prediction records; alias and small-model counts are taken from saved aggregate records of an earlier stage.

ETHICS STATEMENT

This work uses only public research datasets: IMDB reviews (Maas et al., 2011), whose short excerpts appear in the paper as examples, and MuSiQue development questions (Trivedi et al., 2022). It involves no human subjects, no new data collection, and no personal information beyond what those public corpora already contain. The supplement redistributes model outputs and document identifiers, not the source corpora. We see no direct path from these findings to harm; if anything, the controls we propose make it harder to overstate what an evidence-scheduling system has learned.

AI USE STATEMENT

We used generative AI tools extensively in this work: to help frame the research question and refine hypotheses, to design and critique experiments and controls, to implement methods and training and evaluation code, to run and monitor experiments, to generate the synthetic task data (digit and alias chains) through code they helped write, to analyze and interpret results, and to write mathematical exposition. We did not use them to translate text or to conduct surveys or interviews, and no human-subject data are involved. We also used them to identify and summarize related work, to format references, to create the figures and tables, and to draft and edit the manuscript. Every figure is drawn by code from recorded model outputs. One script recomputes every final-stage number from saved prediction rows, re-scoring each row and checking row counts and duplicates. Bibliography entries were checked against arXiv, the ACL Anthology, and the publishing venues. We have reviewed all AI-assisted work, and we take responsibility for the final content of this paper, including every claim and artifact produced with the aid of generative AI.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2503.09573.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, 2017. URL https://aclanthology.org/P17-1171/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06654.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. URL https://aclanthology.org/2023.emnlp-main.495/.

Paul Jünger, Justin Lovelace, Linxi Zhao, Dongyoung Go, and Kilian Q. Weinberger. Self-augmenting retrieval for diffusion language models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.06474.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, 2024. URL https://arxiv.org/abs/2406.10149.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011. URL https://aclanthology.org/P11-1015/.

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. URL https://arxiv.org/abs/2502.09992.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

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. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023. URL https://aclanthology.org/2023.acl-long.557/.

A PROTOCOL DETAILS AND REPRODUCIBILITY

Experiment stages. Results come from two stages. The earlier controlled stage supplies the alias experiment, the online/replay equality checks, and the small-model results; it uses three seeds for trained readers, and alias evaluation is zero-shot. The final stage froze its training and evaluation configurations before any held-out review test evaluation, and added three path-set digit readers, nine review readers (three training inputs $\times$ three seeds), fresh digit items, and the depth-12 extension. A template-boundary error in the new independent-classification prompt was found and fixed during development smoke tests, before test evaluation. No test run or seed was discarded, including the path-set seeds that failed to train.

Data construction. Digit chains are simple paths on 16 records with independently sampled digits and shuffled presentation order. Review paths connect sampled document excerpts and include four unrelated excerpts at both train and test time. The cached IMDB corpus contains 839 training, 180 development, and 180 test documents; cleaned-text hashes, visible 300-character prefixes, and document IDs are disjoint across splits. This rules out exact overlap but not near-duplicate phrasing or pretraining contamination. Evaluation chain seeds start at 5,000,000, depth-12 digit seeds at 6,000,000, and budget development seeds at 4,000,000. Smoke seeds start at 9,000,000 and are excluded from all reported results.

8B training. Each step samples a depth, an item, and a target block, with earlier output slots teacher-forced. The three training inputs select different visible records but supervise the same current block. Digits use one digit token and a separator per hop; reviews use the tokenizer's single tokens for Positive and Negative. LoRA uses rank 16, scaling 32, dropout 0.05, AdamW with gradient clipping at norm 1.0, and learning rate 10^{-4}, targeting the attention query/key/value/output and feedforward projections. The constant learning-rate recipe was not tuned per training input, which is one reason the path-set runs vary so widely. All reported checkpoints use 2,000 steps, and the 500- and 1,000-step adapters come from the same uninterrupted trajectory. A shared step budget is not a shared token or FLOP budget.

Small-model training. The 2.1M model is a task-trained masked Transformer rather than a pretrained diffusion model, so it tests whether generated-pointer control is learnable at all, not whether it is competitive. Training uses 400 items per depth at depths 1/2/4/8, learning rate 3×10^{-4}, batch size 32, and a one-cycle schedule over 6,000 steps. Its three-slot hop representation holds a pointer, a value, and a separator. Training samples $B = D$ with probability 0.7 and otherwise a block count between 1 and D, so the number of supervised reachable hops varies with the read program and the programs are not matched on supervised tokens. Earlier pointers are teacher-forced during training and generated during self-pointer evaluation. The reported view fixes $B = D$ and $T = 3D$ with 256 held-out items per depth; other block counts are retained in the evidence record and are not pooled.

Scoring and accounting. Digit EM compares the first D parsed integers against the answer sequence; small-model EM compares the full space-separated value sequence; review EM compares the extracted label sequence, with independent outputs parsed separately and invalid labels left in place; alias scoring compares the first extracted number against the terminal value. These output-format rules differ by task and are implemented explicitly rather than shared. Per-example records retain the output, target, read program, depth, seed, forward count, processed tokens, and read trace, plus each raw one-document output for independent classification. The controller indexes visible input text and is not learned. Its work is not free either: parsing and reader construction are included in recorded evaluation time.

Reproduction entry points. `train_llada_pointer.py` trains each 8B reader; `run_submission_controls.py` evaluates the fixed controls; `train_tiny_pointer.py` and `run_pointer_ladder.py` implement the 2.1M-parameter model; `run_qa_feedback_v4.py` implements the MuSiQue diagnostic. The supplementary README gives commands parameterized by user-supplied model and data locations rather than cluster-specific paths. `build_paper_20260924.py` regenerates every figure and table from the frozen evidence, auditing 23,764 rows for duplicate cell IDs and EM re-scoring discrepancies; both counts are zero. Of these, 23,296 are digit and review prediction rows, 384 are QA rows, and the remaining 84 come

from smoke runs. Rows are repeated measurements on shared items, so their count is not a sample size.

Superseded results. Two earlier results are excluded rather than reported. An older run of our rule-based reference solver (a non-neural program that answers from the visible records) appeared to show perfect accuracy when all path records were visible at once. The current solver reads visible records in sorted-ID order and does not generally resolve multi-record paths, the rerun does not reproduce the original numbers, and so those counts are withdrawn. Separately, an early QA controller used decimal token-ID strings with a response window owned only by the last block, which left no meaningful intermediate query at earlier retrieval steps. That pilot is too degenerate to serve as evidence for or against state-conditioned retrieval.

B COMPLETE SEED RESULTS

Table 3: Final-stage test EM (%) for every seed and every evaluated train/evaluate direction. Digit and review tasks each use 128 items per depth per seed; the same items are shared across checkpoints, so the six columns are repeated measurements rather than distinct questions.

Task	Train	Evaluate	D4: seeds 0 / 1 / 2			D8: seeds 0 / 1 / 2		
Digit	Single-record	Replay	100.0	100.0	100.0	100.0	100.0	100.0
Digit	Single-record	Path set	53.1	67.2	45.3	0.0	1.6	0.0
Digit	Single-record	Full	0.8	3.1	0.0	0.0	0.0	0.0
Digit	Full-context	Replay	57.8	3.9	38.3	0.8	0.0	0.0
Digit	Full-context	Path set	97.7	100.0	100.0	82.8	88.3	93.8
Digit	Full-context	Full	99.2	86.7	97.7	57.0	54.7	80.5
Digit	Path-set	Replay	0.0	9.4	10.2	0.0	0.0	0.0
Digit	Path-set	Path set	0.0	89.8	100.0	0.0	29.7	89.8
Digit	Path-set	Full	0.0	2.3	51.6	0.0	0.0	16.4
Review	Single-record	Replay	45.3	66.4	64.1	16.4	29.7	30.5
Review	Single-record	Path set	18.0	18.0	19.5	0.0	0.0	0.0
Review	Single-record	Full	15.6	12.5	13.3	0.0	0.0	0.0
Review	Single-record	Independent	59.4	64.1	65.6	30.5	28.1	34.4
Review	Full-context	Replay	9.4	52.3	53.9	1.6	17.2	14.1
Review	Full-context	Path set	7.8	53.1	31.2	0.0	6.2	0.0
Review	Full-context	Full	7.8	58.6	33.6	0.0	3.9	0.0
Review	Full-context	Independent	46.9	59.4	53.1	17.2	29.7	22.7
Review	Path-set	Replay	53.9	57.8	14.8	14.1	27.3	0.0
Review	Path-set	Path set	25.8	28.1	14.8	0.8	1.6	0.0
Review	Path-set	Full	18.0	13.3	13.3	0.0	1.6	0.0
Review	Path-set	Independent	65.6	65.6	64.1	38.3	37.5	42.2

Table 4: Depth-12 digit EM (%), outside the training range for every reader. Each seed sees the same 64 items. The external parser supplies the path, so these numbers describe reader and system generalization rather than learned routing in 8B.

Train / evaluate	Seed 0	Seed 1	Seed 2	Mean
Single-record / Replay	100.0	100.0	100.0	100.0
Single-record / Path set	0.0	0.0	0.0	0.0
Single-record / Full	0.0	0.0	0.0	0.0
Full-context / Replay	0.0	0.0	0.0	0.0
Full-context / Path set	51.6	57.8	78.1	62.5
Full-context / Full	39.1	26.6	64.1	43.2
Path-set / Replay	0.0	0.0	0.0	0.0
Path-set / Path set	0.0	1.6	39.1	13.5
Path-set / Full	0.0	0.0	3.1	1.0

Table 5: Small-model self-pointer EM as exact counts at $B = D$, with every seed shown. The depth-8 spread is the reason we describe generated-pointer control as demonstrated but not seed-robust.

Training seed	D1	D2	D4	D8
0	256/256	256/256	256/256	256/256
1	256/256	256/256	256/256	190/256
2	256/256	256/256	248/256	190/256

C EXACT VALUES BEHIND FIGURE 4

Table 6: Digit-chain EM (%), mean $\pm$ sample standard deviation over three training seeds on 128 shared items per depth (Figure 4a). Rows are training inputs, columns evaluation programs. Per-seed values, including the failed path-set runs, are in Table 3.

	Depth 4			Depth 8		
Training input	Replay	Path set	Full	Replay	Path set	Full
Single-record	100.0 ± 0.0	55.2 ± 11.1	1.3 ± 1.6	100.0 ± 0.0	0.5 ± 0.9	0.0 ± 0.0
Path-set	6.5 ± 5.7	63.3 ± 55.0	18.0 ± 29.1	0.0 ± 0.0	39.8 ± 45.8	5.5 ± 9.5
Full-context	33.3 ± 27.3	99.2 ± 1.4	94.5 ± 6.8	0.3 ± 0.5	88.3 ± 5.5	64.1 ± 14.3

Table 7: Review-chain EM (%), mean ± sample standard deviation over three training seeds (Figure 4b); replay and independent columns share checkpoints. The base model is LLaDA-8B-Instruct without our fine-tuning and has no seed variation. Each depth uses 128 chains drawn from 180 shared documents, so these are descriptive statistics, not independent-chain intervals.

	Depth 4		Depth 8	
Training input	Replay	Independent	Replay	Independent
Single-record	58.6 ± 11.6	63.0 ± 3.3	25.5 ± 7.9	31.0 ± 3.2
Path-set	42.2 ± 23.8	65.1 ± 0.9	13.8 ± 13.7	39.3 ± 2.5
Full-context	38.5 ± 25.3	53.1 ± 6.2	10.9 ± 8.3	23.2 ± 6.3
Base model	—	52.3	—	18.0

D How the error diagnostics are computed

Jumps to another occurrence. For each wrong digit-chain answer we find the first hop i at which the output differs from the gold sequence g. Answers wrong at hop 1 or truncated before hop i are skipped, because they have no preceding digit to condition on. Let $A = \{g_{j+1} : g_j = g_{i-1},\ j \neq i-1\} \setminus \{g_i\}$ be the digits that follow some other occurrence of the previous gold digit. The error is a *jump* if the output digit at hop i lies in A. The uniform chance rate is the mean of $|A|/9$, the probability that a uniformly random wrong digit falls in A. The stricter visible-record rate assumes the reader copies the value of a uniformly chosen other record it can see: it is the mean over errors of the fraction of visible records whose value differs from g_i and lies in A, where the visible records are the D path records under the path-set program and all 16 records on the full input. The reported rates pool the three training seeds of the full-context reader: at depth 12, 62 of 72 first errors on the path set are jumps. The definition counts both forward and backward jumps; forward jumps are the majority in every condition. At depth 12 on the path set, a jump-type first error occurs in 62 of the 192 item–seed pairs, 32.3 of the 37.5 points; this is a description of the errors, not an estimate of what removing them would recover.

Error rate after repeated and unique digits. For each answer we walk along the hops while every earlier digit is correct. Each hop $i \geq 2$ reached this way is at risk; it follows a *repeated* digit if g_{i-1} occurs more than once in the gold sequence and a *unique* digit otherwise, and it counts as an error if the output digit at hop i is wrong, after which the walk stops. The rate ratio pools the three training seeds and is the Mantel–Haenszel estimate over hop positions, so it compares repeated and unique digits at the same position in the chain. Under the single-record reader no hop is ever wrong, so the ratio is undefined there.

Per-review accuracy. Label accuracy is the fraction of chain positions whose extracted label matches the gold label, with invalid or missing labels counted as wrong. The base model emits no label for 7% of reviews at depth 8; on the reviews it does label it is 89% accurate, and the trained readers score 87–91% on those same reviews. The comparison with a^D uses the pooled accuracy a of each reader–program–depth cell and so assumes errors are independent across positions; the 14 cells are the three trained readers under replay and independent classification at depths 4 and 8, plus the base model's independent classification at both depths. The constant-label rate counts chains whose output uses only one of the two labels.

E Statistics and training-budget sensitivity

For the depth-12 comparison we align item seeds between single-record-trained replay and full-context-trained path-set evaluation, average each item's difference over the three fixed training checkpoints, and resample the 64 item indices with replacement 10,000 times (NumPy generator seed 20260924). The mean difference is 37.5 percentage points with a percentile 95% interval of 29.7–45.3 points; resampling with other generator seeds moves the upper bound by about half a point. The interval is conditional on these three checkpoints and on the item generator: it describes item-to-item variation, not variation over training runs. With three seeds, the per-seed values in Table 3 remain the more informative summary.

Review chains reuse documents from a 180-document pool, so a chain-level bootstrap treating chains as independent would overstate the evidence about new documents. We therefore report no interval or significance claim for the review results and treat them as descriptive statistics conditional on that pool. Document IDs are retained so that a cluster-aware analysis is possible later; we have not performed one. Table 8 shows how the review readers change with training budget on development chains.

Table 8: Review depth-8 development EM (%) across training budgets, using 32 chains per seed, checkpoint, and read program. Cells show mean ± sample standard deviation over three seeds. The 500-, 1,000-, and 2,000-step checkpoints come from one uninterrupted trajectory. All final tests use 2,000 steps; these development scores were not used to select checkpoints.

Training input	Steps	Replay	Path set	Full
Single-record	500	37.5 ± 14.3	0.0 ± 0.0	0.0 ± 0.0
Single-record	1000	38.5 ± 1.8	0.0 ± 0.0	1.0 ± 1.8
Single-record	2000	27.1 ± 10.0	0.0 ± 0.0	0.0 ± 0.0
Full-context	500	10.4 ± 7.9	1.0 ± 1.8	1.0 ± 1.8
Full-context	1000	15.6 ± 14.3	0.0 ± 0.0	0.0 ± 0.0
Full-context	2000	15.6 ± 12.5	0.0 ± 0.0	3.1 ± 3.1
Path-set	500	16.7 ± 7.9	2.1 ± 1.8	0.0 ± 0.0
Path-set	1000	18.8 ± 8.3	3.1 ± 3.1	3.1 ± 0.0
Path-set	2000	11.5 ± 10.0	2.1 ± 1.8	1.0 ± 1.8

F ALIAS SELECTION CONTROLS

Table 9: Alias exact counts by depth and number of decoy assignment lines, with 32 real-text filler chunks in every condition. These are custom generated tasks and not official RULER or BABILong results.

Read program	D4 / 0	D4 / 32	D8 / 0	D8 / 32
Online parser	200/200	200/200	200/200	200/200
Replay	200/200	200/200	200/200	200/200
Path set	200/200	200/200	199/200	199/200
Terminal record only	200/200	200/200	200/200	200/200
Parser only	200/200	200/200	200/200	200/200
Syntax filter	200/200	172/200	199/200	108/200
Full	80/200	53/200	43/200	28/200

Syntax filtering retains every assignment line, decoys included, which is why adding decoys costs it so much. The text parser follows named assignments and so already solves the symbolic chain; terminal-record-only passes just the last selected assignment to the reader, and parser-only returns the resolved value with no forward pass at all. Their perfect scores measure the parser, which is precisely why symbolic work of this kind must not be credited to diffusion reasoning. The 5,792 online/replay checks all come from the earlier experimental stage: 1,184 pairs from the alias, digit, and review controls (alias with the zero-shot model), plus 4,608 pairs from that stage's 2,000-step single-record and full-context digit and review checkpoints, three seeds each. Read traces and output strings agree in every one; 32 further pairs from calibration runs, not counted in the 5,792, also agree. The final-stage evaluations run the precomputed schedule only and rely on the construction of Section 2.

G COST ACCOUNTING

Equal forward counts hide a large spread in processing cost, since full-context and single-record inputs differ by more than a factor of two in processed tokens. The comparison also runs the other way: a parser may scan every record before handing the reader a short input, work that no forward-count or visible-token measure captures. We therefore report both columns and do not convert either into a FLOP or throughput claim.

Table 10: Depth-8 digit evaluation cost per item, averaged over three seeds and 128 items. Every read program uses 16 forward passes; processed tokens sum sequence lengths across those passes. Times include controller and prompt preparation on shared GPUs and are descriptive rather than controlled benchmarks.

Train / evaluate	Processed tokens	Seconds
Single-record / Replay	1,810	1.04
Single-record / Path set	3,230	1.17
Single-record / Full	4,867	1.45
Full-context / Replay	1,810	1.04
Full-context / Path set	3,230	1.16
Full-context / Full	4,867	1.44
Path-set / Replay	1,810	1.04
Path-set / Path set	3,230	1.17
Path-set / Full	4,867	1.45

H MuSiQue feedback diagnostic

This diagnostic asks whether the approach survives when pointers are not given in the text. It runs LLaDA-8B-Instruct zero-shot, without any of our LoRA adapters. We evaluate 32 MuSiQue-Ans v1.0 development questions at each of 2, 3, and 4 hops (96 total), skipping the first four eligible questions per depth, which were used during piloting. Using development data makes this a diagnostic rather than a benchmark result. All available paragraphs (at most 20) are retained and truncated to 600 characters. Each read program generates three 24-token factual notes followed by a 32-token answer, for 104 forward passes, appending previously decoded notes to the question at each call. BM25 (Robertson & Zaragoza, 2009) (our own implementation: k_1=1.5, b=0.75, IDF $\log(1 + \frac{N-n_t+0.5}{n_t+0.5})$, lowercased punctuation-stripped whitespace tokens, no stemming or stopword removal) operates on the visible truncated paragraphs. Dynamic retrieval queries the question plus the notes, static retrieval always uses the question's top four paragraphs, full input keeps every paragraph, and the oracle set uses annotated supporting paragraphs and is explicitly privileged. No other read program sees gold decompositions or intermediate answers.

Table 11: MuSiQue diagnostic counts over 96 questions. Containment is the pre-existing permissive scorer and is reported only to show that outputs are not empty; it is not a validated accuracy measure.

Read program	Strict EM	Containment diagnostic
Dynamic retrieval	0/96	19/96
Static retrieval (top 4)	0/96	18/96
Full	0/96	25/96
Oracle set	10/96	37/96

All 384 runs produce nonempty answers and 93 of 96 dynamic runs change the retrieved paragraphs (82 change which paragraphs, 11 only their order), so the feedback channel is functioning. It does not help: strict normalized EM is zero for dynamic, static, and full input, and the oracle reaches only 10/96, all on 2-hop items. Some failures are about output format: one full-input response ends with the line "Answer: Bombardier Inc.", which matches the gold answer, and still scores zero because strict EM compares the whole output. It is the only one of the 384 outputs with such a line. The containment check catches such cases but also accepts partial and ambiguous mentions, so we do not promote it to a correctness measure. An answer or alias appears in the rendered truncated pool for 78 of 96 items, which bounds retrieval loss without guaranteeing that the supporting evidence survived.

Because it uses separate note-generation calls rather than lookahead tokens within one diffusion trajectory, this diagnostic neither reproduces SARDI nor tests whether model-state retrieval can work in principle. What it does establish is that this zero-shot protocol and output format do not transfer to natural-language multi-hop questions, and that the limitation is unresolved here.

I COVERAGE CALCULATION AND CLAIM BOUNDARIES

For a uniformly sampled ordered simple path of length D on N records there are $(N)_D$ possible paths, and a content-independent set of k records contains $(k)_D$ of them, giving $(k)_D/(N)_D$. Conditioning on the revealed start, count the remaining $D-1$ positions among $N-1$ records. Containment is impossible whenever $k < D$, independently of any probability assumption. Guessing remains possible even without coverage: conditional on seeing m path values and granting the path order, all remaining independent uniform digits are guessed correctly with probability $10^{-(D-m)}$.

The calculation does not apply to our static programs, for two reasons. A parser that reads the input before generation breaks the independence premise outright and can precompute a perfect path. And a time-varying schedule is not a fixed set: the union of its reads can cover D records while never showing more than one at a time. We state the boundary explicitly because it is easy to overread: nothing here shows that every pre-generation plan must fail or that read timing confers a diffusion-specific information advantage.

J AUDITED QUALITATIVE EXAMPLES

These cases use the single-record 8B reader, training seed 0, at depth 4. We take the first test item by item seed and then the first item with an incorrect replay output, a rule fixed in advance so that neither example is selected for a favorable contrast. Excerpts show at most the first 23 words of each visible record; the actual input runs to 300 characters. Because a review's label can depend on text outside the visible prefix, a single mismatch does not by itself diagnose error propagation.

First test item (item seed 5000000). The path is $2 \to 1 \to 3 \to 4$. Gold labels: Positive Positive Positive Negative.

Read program	Recorded output
Replay	Positive Positive Positive Negative
Independent	Positive Positive Positive Negative
Path set	Negative Negative Negative Negative
Full	Negative Negative Negative Negative

First replay error (item seed 5000001). The path is $4 \to 1 \to 3 \to 2$. Gold labels: Negative Positive Positive Positive.

Read program	Recorded output
Replay	Negative Positive Negative Positive
Independent	Negative Positive Negative Positive
Path set	Negative Negative Negative Negative
Full	Negative Negative Negative Negative

Replay and independent classification both fail on record 3 (gold Positive, both Negative), which reads *"You have to be awfully patient to sit through a film with one-liners so flat and unfunny that you wonder what all the . . . "* The visible text is negative while the label comes from the full review, so truncation, not sequential dependence, is the likely explanation.