Test-Time Diffusion Deep Researcher for MMU-RAG

Dmitrii Magas
Independent Scientist
Team TTT-DR
dmitrii@eamag.me

Abstract

We present our implementation of Test-Time Diffusion Deep Researcher (TTD-DR) for the MMU-RAG text-to-text track. Our system conceptualizes research report generation as a diffusion process, where an initial draft is iteratively refined through retrieval-augmented denoising. The framework combines two key mechanisms: (1) report-level refinement via denoising with retrieval, and (2) component-wise optimization via self-evolution. By maintaining a dynamic draft that guides the research direction, our system achieves more coherent information integration while reducing information loss during multi-hop search. We demonstrate the effectiveness of this approach on the MMU-RAG benchmark, which requires intensive search and complex reasoning over web-scale corpora.

1 Introduction

The MMU-RAG competition challenges participants to build RAG systems that handle real-world user queries requiring extensive search and reasoning. Traditional RAG approaches often struggle with complex queries that demand multiple rounds of information gathering and synthesis. Inspired by human research patterns **?**, we implement the Test-Time Diffusion Deep Researcher (TTD-DR) **?**, which models research report generation as an iterative diffusion process.

Our key insight is that effective research involves maintaining an evolving understanding (draft) that guides subsequent searches, rather than collecting information linearly and synthesizing only at the end. This draft-centric approach enables more timely integration of discovered information and reduces the context loss common in long agentic trajectories.

2 System Architecture

2.1 Backbone Deep Research Agent

Our system operates in three main stages:

Stage 1: Research Plan Generation. Given a user query, we generate a structured research plan outlining key areas to investigate using self-evolution (see below). This plan serves as a high-level scaffold for the entire research process.

Stage 2: Iterative Search and Synthesis. This stage contains two sub-agents in a loop workflow:

- *Search Question Generation*: Formulates targeted search queries based on the research plan, user query, and previous search context
- *Answer Synthesis*: Retrieves documents via FineWeb Search API, applies intelligent chunking and reranking, then synthesizes answers using a RAG-style approach

Stage 3: Final Report Generation. Synthesizes all gathered information (plan and question-answer pairs) into a comprehensive final report.

2.2 Component-wise Self-Evolution

To enhance each stage's output quality, we apply a self-evolutionary algorithm inspired by **?**. For each component (plan, question, answer):

1. Generate multiple diverse initial variants
2. Apply environmental feedback via LLM-as-a-judge for helpfulness and comprehensiveness
3. Revise variants based on feedback scores and critiques
4. Merge revised variants into a single high-quality output

This process encourages exploration of diverse information and maintains high-quality context throughout the workflow.

2.3 Report-level Denoising with Retrieval

Our core innovation is treating report generation as a diffusion process. We initialize an early draft from the LLM's internal knowledge, then iteratively refine it:

1. Generate initial "noisy" draft based on user query
2. Feed current draft to Stage 2a to generate next search query
3. Retrieve and synthesize answer (Stage 2b)
4. Revise draft by integrating new information
5. Repeat until sufficient information is gathered

This continuous feedback loop ensures the evolving draft guides search direction while retrieved information progressively "denoises" the report. The synergy between self-evolution (providing quality context) and diffusion (maintaining coherence) is critical for performance.

3 Implementation Details

Infrastructure. We deploy our system as a Docker container with vLLM for efficient inference. The system exposes two endpoints:

- `/evaluate`: Static evaluation endpoint accepting `{query, iid}` and returning `{query_id, generated_response}`
- `/run`: Streaming SSE endpoint emitting updates with `intermediate_steps`, `final_report`, `citations`, and `complete` flags

Retrieval Pipeline. Our retrieval system implements a sophisticated three-stage approach:

1. *Initial Retrieval*: Query FineWeb Search API to retrieve top-K documents (K=50)
2. *Intelligent Chunking*: Apply custom sentence-level chunking with:
   - Maximum 500 tokens per chunk
   - 50-token overlap between chunks
   - Minimum 500 tokens to avoid fragmentary chunks
   - Metadata preservation (doc_id, URL, char_range)
3. *Neural Reranking*: Deploy Qwen3-Reranker-0.6B via vLLM in scoring mode to rerank chunks based on query relevance, returning top-K reranked chunks

This reranking step is crucial for filtering the most relevant information from retrieved documents before synthesis.

Generation Models. We use a dual-model strategy:

- *Local Inference*: Qwen3-4B-Instruct-2507 served via vLLM (max 16K context, 75% GPU utilization) for fast, local generation

- *Fallback*: OpenRouter API with configurable models (default: tongyi-deepresearch-30b-a3b:free) for systems without GPU

Optimization Strategy. We set maximum denoising iterations to 3 (reduced from 20 in original paper for efficiency). For self-evolution we set all parameters conservatively to not exceed latency budgets using local GPU inference:

- Generate N=1 variant for plans and reports
- Generate M=1 variant for search queries
- Apply K=1 evolution step across components

4 Results and Discussion

Our implementation focuses on the text-to-text track of MMU-RAG. Key technical innovations include:

Neural Reranking for Quality. Using Qwen3-Reranker-0.6B in vLLM's scoring mode enables efficient, GPU-accelerated reranking of retrieved chunks. This is particularly valuable given FineWeb's large result sets - we retrieve 50 documents, chunk them into potentially hundreds of segments, then rerank to select the top-K=20 most relevant chunks for synthesis.

Draft-Centric Information Flow. Unlike traditional RAG systems that collect all information before generation, our evolving draft:

- Guides targeted search query formulation at each iteration
- Incorporates findings progressively, reducing context window pressure
- Maintains coherence across multi-hop reasoning chains
- Enables early stopping when sufficient information is gathered

Practical Trade-offs. We reduced maximum iterations from 20 (original paper) to 3 for competition constraints, balancing thoroughness with latency. Similarly, self-evolution uses minimal variants (1-2) rather than the paper's 3-5, prioritizing deployment efficiency.

Strengths:

- Efficient GPU utilization via vLLM for both generation and reranking
- Modular chunking pipeline handles diverse document structures
- Graceful fallback to OpenRouter for non-GPU environments
- Clear separation of static (`/evaluate`) and streaming (`/run`) evaluation modes

Limitations:

- Conservative iteration limits may miss deeper insights for complex queries
- Single retrieval source (FineWeb); no ensemble or browsing tools
- Self-evolution limited to minimize latency overhead
- No learned components; fully reliant on prompting and in-context learning

5 Conclusion

We present a practical implementation of Test-Time Diffusion Deep Researcher for the MMU-RAG competition. Our system makes several key technical contributions: (1) efficient neural reranking via Qwen3-Reranker-0.6B in vLLM, (2) intelligent chunking that preserves document structure while optimizing for context windows, and (3) a draft-centric diffusion process that guides multi-hop search. By balancing the paper's ambitious algorithms with competition constraints, we demonstrate that diffusion-based research agents can be deployed efficiently. Future work includes exploring learned reranking models, integrating browsing tools, and adaptive iteration budgets based on query complexity.

Acknowledgments

We thank the MMU-RAG organizers for creating this benchmark, the TTD-DR authors for the foundational research, and the vLLM team for their efficient inference framework.

References

L. Flower and J.R. Hayes, *A cognitive process theory of writing*, College Composition and Communication, 32(4):365–387, 1981.

R. Han, Y. Chen, Z. CuiZhu, et al., *Deep Researcher with Test-Time Diffusion*, arXiv:2507.16075, 2025.

K.-H. Lee, I. Fischer, Y.-H. Wu, et al., *Evolving deeper LLM thinking*, arXiv:2501.09891, 2025.