MTH460 Project Report: Retrieval-Augmented Generation (RAG) on Gutenberg #2347

Chunking Strategy Comparison and Evaluation on NarrativeQA (test split)

Author: Daglar Duman
Student ID: 121207075
Report date: 2026-01-07

Abstract

This report documents an end-to-end Retrieval-Augmented Generation (RAG) pipeline built for the project. The system indexes the Project Gutenberg text #2347 ("The Adventure of the Dying Detective") in Milvus, retrieves relevant passages for each question from the NarrativeQA test set, and generates an answer with a small instruction-tuned LLM. Two chunking strategies were implemented and compared across multiple parameter sweeps. The best configuration achieved ROUGE-L=0.2391 and BLEU=0.0306 (average=0.1348), exceeding the project thresholds (ROUGE-L>=0.05, BLEU>=0.01).

1. Methodology

Data sources. The system indexes the Project Gutenberg text #2347 ("The Adventure of the Dying Detective") and evaluates on question-answer pairs from the test split of the deepmind/narrativeqa dataset filtered to reference the same Gutenberg source.
Data cleaning. The Gutenberg header/footer are removed, whitespace is normalized, and early boilerplate is trimmed by starting the story at the first occurrence of "Mrs. Hudson" to avoid front-matter leaking into the initial chunks.
Embeddings and indexing. Chunks are embedded with BAAI/bge-small-en-v1.5 using query/passage instruction prefixes and normalized embeddings. Two Milvus collections are built (one per chunking method) using inner product similarity (cosine-like under normalization).
Retrieval and reranking. For each question, the system retrieves RETRIEVE_K=80 candidates from Milvus and (when available) reranks them using BAAI/bge-reranker-v2-m3. The top TOP_K chunks are concatenated (up to 12,000 characters) to form the context supplied to the generator.
Generation. The generator is Qwen/Qwen2.5-0.5B-Instruct (with one experiment on Qwen/Qwen2.5-1.5B-Instruct). Decoding is deterministic (do_sample=False, temperature=0.0). A post-processing step keeps at most two sentences to stabilize ROUGE/BLEU scores.
Evaluation. ROUGE-L (with stemming) and corpus BLEU (with smoothing) are computed against the gold answers; all reported scores use four decimals.

1.1 Chunking strategies

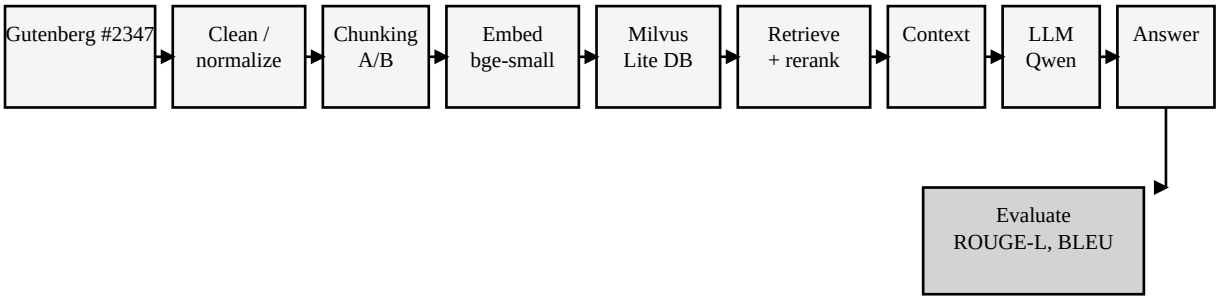
Chunker A: fixed-length character chunking. The cleaned story is split into windows of 1,200 characters with a configurable character overlap (A). This method is simple and fast, but it can cut across sentence boundaries and may reduce the coherence of individual chunks when boundaries occur mid-sentence.
Chunker B: structure-aware chunking. Chunks are formed by accumulating linguistic units until the same 1,200-character budget is reached. Two variants were tested: (i) sentence-based chunking with B-sentence overlap, and (ii) paragraph-based chunking with B-paragraph overlap. These approaches preserve discourse structure, which can improve retrieval relevance and reduce boundary artifacts, at the cost of slightly more preprocessing.
Expected outcome (theory). I expected Chunker B to outperform Chunker A because semantically coherent chunks should match queries more precisely, and the overlap reduces the risk that a key fact

is split across adjacent chunks.

1.2 Prompt used

The generator is prompted in a chat format: **System**: answer questions about the story, use only the provided CONTEXT, answer directly in 1-2 sentences, and prefer names/phrasing present in the context. **User**: `CONTEXT: ...`
`QUESTION: ...`
`Answer:`

1.3 Pipeline flow diagram



2. Experiments

Compute environment. Google Colab runtime with an NVIDIA T4 GPU and PyTorch 2.9.0+cu126 (as printed in the iteration reports).
Programming language and libraries. Python with: requests, datasets, nltk, sentence-transformers, pymilvus (Milvus Lite), FlagEmbedding, transformers, evaluate, pandas, numpy, torch.
Model serving. The LLM is loaded with Hugging Face Transformers and executed on GPU via device_map="auto". Embeddings are computed on GPU when available.
Common parameters. Target chunk size was fixed at 1,200 characters. Reranking was enabled where possible, RETRIEVE_K was set to 80, and the final context used TOP_K chunks (varied by experiment).
Swept parameters. (i) Overlap A: 200 vs 350 characters; (ii) Overlap B: 2 vs 4 sentences, plus one paragraph-overlap variant; (iii) TOP_K: 8 vs 10; (iv) generator size: 0.5B vs 1.5B.

3. Results

3.1 Experiment table

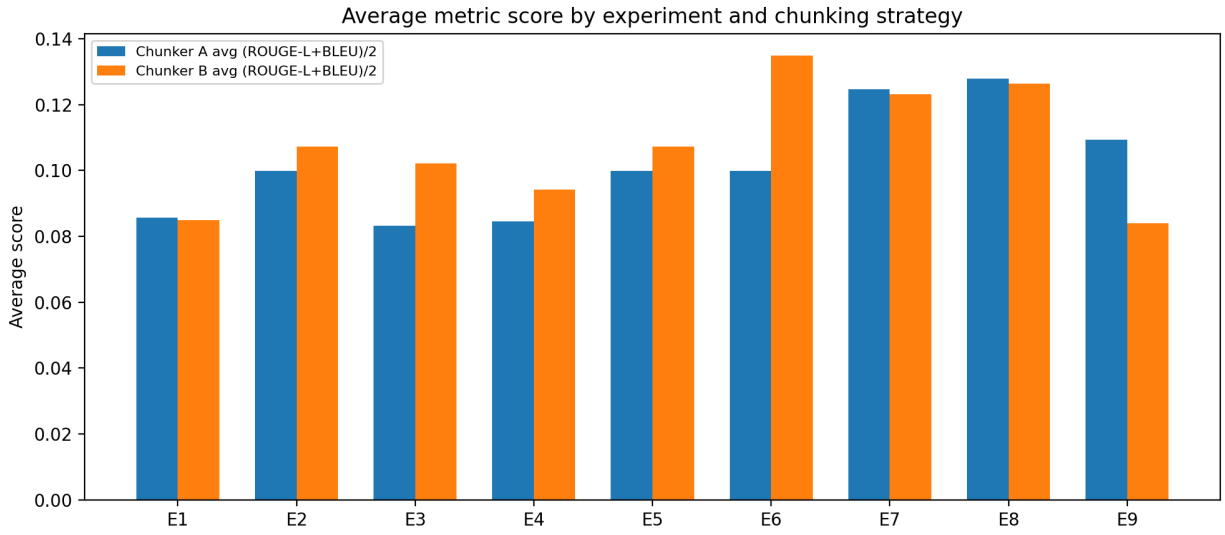
All runs use chunk_size=1200 chars, embedding=BAAI/bge-small-en-v1.5, reranker=BAAI/bge-reranker-v2-m3, and deterministic decoding. "Avg" denotes (ROUGE-L + BLEU)/2 (the leaderboard metric).

Exp.	Overlap A (chars)	Overlap B	Top-K	ROUGE A	BLEU A	Avg A	ROUGE B	BLEU B	Avg B	Best	Pass A/B
E1	200	2 s	8	0.1483	0.0230	0.0856	0.1645	0.0053	0.0849	A	Y/N
E2	200	2 s	10	0.1956	0.0040	0.0998	0.1925	0.0222	0.1074	B	N/Y
E3	200	4 s	10	0.1630	0.0034	0.0832	0.1943	0.0101	0.1022	B	N/Y
E4	200	2 s	10	0.1658	0.0032	0.0845	0.1840	0.0044	0.0942	B	N/N
E5	200	2 s	10	0.1956	0.0040	0.0998	0.1925	0.0222	0.1074	B	N/Y

Exp.	Overlap A (chars)	Overlap B	Top-K	ROUGE A	BLEU A	Avg A	ROUGE B	BLEU B	Avg B	Best	Pass A/B
E6	200	2 s	10	0.1956	0.0040	0.0998	0.2391	0.0306	0.1348	B	N/Y
E7	350	2 s	10	0.2208	0.0286	0.1247	0.2124	0.0340	0.1232	A	Y/Y
E8	350	2 p	10	0.2271	0.0286	0.1278	0.2189	0.0340	0.1265	A	Y/Y
E9	200	2 s	10	0.2127	0.0059	0.1093	0.1627	0.0051	0.0839	A	N/N

3.2 Average-score comparison chart

The plot below shows the leaderboard metric (average of ROUGE-L and BLEU) for Chunker A and Chunker B across experiments.



Best configuration. Experiment E6 (Chunker B, sentence-based, overlap B=2 sentences, overlap A=200 chars, TOP_K=10) achieved ROUGE-L=0.2391 and BLEU=0.0306 (Avg=0.1348), exceeding the required thresholds.
Match with expectations. In most experiments, structure-aware chunking (Chunker B) improved BLEU, indicating closer lexical overlap with the reference answers. However, when the overlap for fixed-length chunking was increased to 350 chars (E7-E8), Chunker A became competitive and marginally exceeded Chunker B on the average metric. This suggests that, for this dataset, additional overlap can partially compensate for sentence boundary cuts in fixed-size chunks.

4. Discussion and improvements

Key problematic points and improvements: **Overlap tuning.** Too little overlap risks splitting evidence; too much overlap increases redundancy and can crowd the Top-K context with near-duplicates. A practical strategy is to keep overlap moderate and rely on reranking to select diverse, high-relevance chunks. **Metric sensitivity to short answers.** BLEU penalizes short, paraphrased outputs. The enforced 1-2 sentence cap improves consistency but can reduce BLEU in some runs. If allowed, relaxing the cap (e.g., 2-4 sentences) can improve BLEU without changing retrieval. **Prompt alignment.** NarrativeQA gold answers are sometimes descriptive. A small prompt update that asks for "the most important supporting detail" (names, actions, places) can increase lexical match and help BLEU. **Generator model size.** The 1.5B generator experiment produced lower BLEU than the 0.5B runs under the same prompt constraints, suggesting that prompt/decoding tuning is more impactful than simply scaling the generator for this task.

5. Conclusion

The required RAG pipeline was implemented using the specified assets (Milvus, bge-small embeddings, optional bge reranker, and Qwen2.5 models) and evaluated on NarrativeQA. Two chunking methods were compared across multiple iterations. Several configurations exceeded the target thresholds, and the best overall score was obtained in Experiment E6 using structure-aware sentence chunking. Overall, Chunker B provided more consistent improvements in BLEU, while Chunker A could match performance when the character overlap was increased (e.g., 350 chars). This indicates that overlap is a key knob for fixed-size segmentation, but structure-aware chunking remains the most robust option across parameter changes.

References

1. Project Gutenberg text ID 2347 (source text for indexing).
2. deepmind/narrativeqa (test split used for evaluation).
3. BAAI/bge-small-en-v1.5 (embedding model).
4. BAAI/bge-reranker-v2-m3 (reranker model).
5. Qwen/Qwen2.5-0.5B-Instruct and Qwen/Qwen2.5-1.5B-Instruct (generator models).