



MTH-424 PROJECT REPORT

RAG Pipeline Implementation and Evaluation on Anne of Avonlea

Selin Koçyiğit

Student ID: [REDACTED]

Department of Industrial Engineering

May 2026

Table of Contents

Abstract

1. Introduction

2. Methodology

2.1 Data Preprocessing and Text Cleaning

2.2 QA Dataset Loading

2.3 Chunking Strategies

2.4 System Prompt

2.5 RAG Pipeline Flow Diagram

3. Experimental Setup

3.1 Computational Environment (GPU Specs)

3.2 Programming Language and Libraries

3.3 How the Models Were Served

3.4 Parameters Used in the Experiments

4. Results

4.1 Quantitative Results

4.2 Comparison and Commentary

5. Discussion

5.1 Problematic Points Observed

5.2 Potential Improvements

6. Conclusion

Abstract

This report presents the implementation and evaluation of a Retrieval-Augmented Generation (RAG) pipeline built on the literary corpus Anne of Avonlea (Project Gutenberg #47) by Lucy Maud Montgomery. The system integrates the BAAI/bge-small-en-v1.5 embedding model, a Milvus Lite vector database and the Qwen/Qwen2.5-0.5B-Instruct large language model. Two document chunking strategies — Fixed-Size Chunking and Sentence-Based Chunking — were implemented and evaluated on 30 question-answer pairs from the deepmind/narrativeqa benchmark using ROUGE-L and BLEU metrics. Both strategies exceeded the minimum required thresholds ($\text{ROUGE-L} \geq 0.05$, $\text{BLEU} \geq 0.01$). Fixed-Size Chunking achieved ROUGE-L: 0.1836 and BLEU: 0.0370; Sentence-Based Chunking achieved ROUGE-L: 0.2341 and BLEU: 0.0260. Sentence-Based Chunking performed better on ROUGE-L, while Fixed-Size performed better on BLEU.

1. Introduction

Large Language Models (LLMs) have transformed natural language processing. However they are constrained by their training data cutoff and are prone to hallucinations when answering questions about domain specific or document specific knowledge not present in their parameters.

Retrieval Augmented Generation (RAG) addresses these limitations by combining parametric memory (LLM weights) with non-parametric memory (external document store). At inference time, the system retrieves relevant passages from a vector database and injects them into the prompt, grounding the answer in factual, retrievable evidence. This significantly reduces hallucinations and improves factual accuracy.

In this project, a complete RAG pipeline was implemented using the tools prescribed by the course: BAAI/bge-small-en-v1.5 as the embedding model, Milvus Lite as the vector database and Qwen2.5-0.5B-Instruct as the generative model. The pipeline was evaluated on the NarrativeQA benchmark to determine whether it surpasses the minimum thresholds of $\text{ROUGE-L} \geq 0.05$ and $\text{BLEU} \geq 0.01$ and to compare the effect of two distinct chunking strategies on retrieval and generation quality.

2. Methodology

2.1 Data Preprocessing and Text Cleaning

The raw text of Anne of Avonlea was downloaded directly from Project Gutenberg via HTTP request (URL: <http://www.gutenberg.org/ebooks/47.txt.utf-8>). The raw file contained substantial non-narrative boilerplate: legal disclaimers, licensing terms and metadata headers that would pollute the vector space if left in.

The following cleaning steps were applied using Python's re library:

- Header removal: all text before the '*** START OF THE PROJECT GUTENBERG EBOOK ***' marker was discarded.
- Footer removal: all text after the '*** END OF THE PROJECT GUTENBERG EBOOK ***' marker was discarded.
- Line ending normalization: Windows style `\r\n` sequences converted to Unix `\n`.
- Whitespace normalization: three or more consecutive blank lines collapsed to one blank line.

Findings from the cleaning process:

Metric	Value
Raw text length	517,539 characters
Clean text length	487,961 characters
Boilerplate removed	29,578 characters (5.7%)

2.2 QA Dataset Loading

The deepmind/narrativeqa test split (10,557 total rows) was loaded via the Hugging Face datasets library. To extract only the QA pairs for Anne of Avonlea, an exact URL match filter was applied:

```
url == "http://www.gutenberg.org/ebooks/47.txt.utf-8"
```

This exact match was essential — a substring match on 'ebooks/47' also matched unrelated entries such as /ebooks/4761 (The Cossacks by Tolstoy) , which would corrupt the evaluation. The exact match yielded 30 QA pairs for Anne of Avonlea.

2.3 Chunking Strategies

Two distinct chunking strategies were implemented to compare their impact on retrieval quality and generation performance.

Strategy 1 : Fixed-Size Character Chunking

The cleaned book text is divided into fixed length character windows with a configurable sliding overlap between consecutive chunks.

Parameters used: chunk_size = 500 characters, overlap = 50 characters.

Aspect	Detail
Definition	Splits text at fixed character intervals regardless of sentence or word boundaries
Advantages	Simple, fast, uniform chunk sizes, predictable memory usage
Disadvantages	May split mid-sentence or mid-word; no semantic awareness; context fragmentation possible
Result	1,085 chunks produced, average length 499 characters

Strategy 2 : Sentence-Based Chunking

The text is tokenized into individual sentences using NLTK's punkt tokenizer then grouped into windows of N sentences with a one sentence overlap.

Parameters: sentences_per_chunk = 5, overlap = 1 sentence.

Aspect	Detail
Definition	Groups N consecutive sentences per chunk ensuring no sentence is cut mid-way
Advantages	Semantically coherent chunks; linguistically complete units; better factual QA retrieval
Disadvantages	Variable chunk sizes; slower due to sentence tokenization overhead
Result	1,698 chunks produced, average length 360 characters

Expected Outcome: Sentence-Based Chunking was expected to yield better ROUGE-L scores because it preserves complete semantic units. Fixed size splitting can cut sentences at arbitrary positions, potentially delivering incomplete or incoherent context to the LLM. Sentence aware boundaries ensure that each retrieved chunk contains fully formed ideas, improving the probability that the answer is found within the chunk and that the LLM can extract it correctly.

2.4 System Prompt

The following prompt template was used throughout all experiments. The '1-3 words only' constraint was added to encourage short, extractive style answers that better overlap with the typically brief NarrativeQA ground truths:

Context: {top_2_retrieved_chunks}

Based on the context, answer with 1-3 words only.

Question: {question}

Answer (1-3 words):

Only the top 2 chunks out of the 5 retrieved were included in the prompt to avoid exceeding the Qwen 0.5B model's 1024-token context limit.

2.5 RAG Pipeline Flow Diagram

The pipeline executes the following steps sequentially for each question:

Step	Component	Description
1. Query Embedding	BAAI / bge-small-en-v1.5	Question encoded to 384-dim vector with L2 normalization
2. Vector Retrieval	Milvus Lite (Cosine)	Top-K=5 most similar chunks retrieved from collection
3. Prompt Construction	Template	Top 2 chunks + question inserted into prompt template
4. LLM Generation	Qwen2.5-0.5B-Instruct	Greedy decoding, max_new_tokens=10, do_sample=False
5. Post-processing	Python string ops	Remove “Answer:” artifacts, keep first line only
6. Evaluation	rouge-score + sacrebleu	ROUGE-L and BLEU computed vs ground truth answer

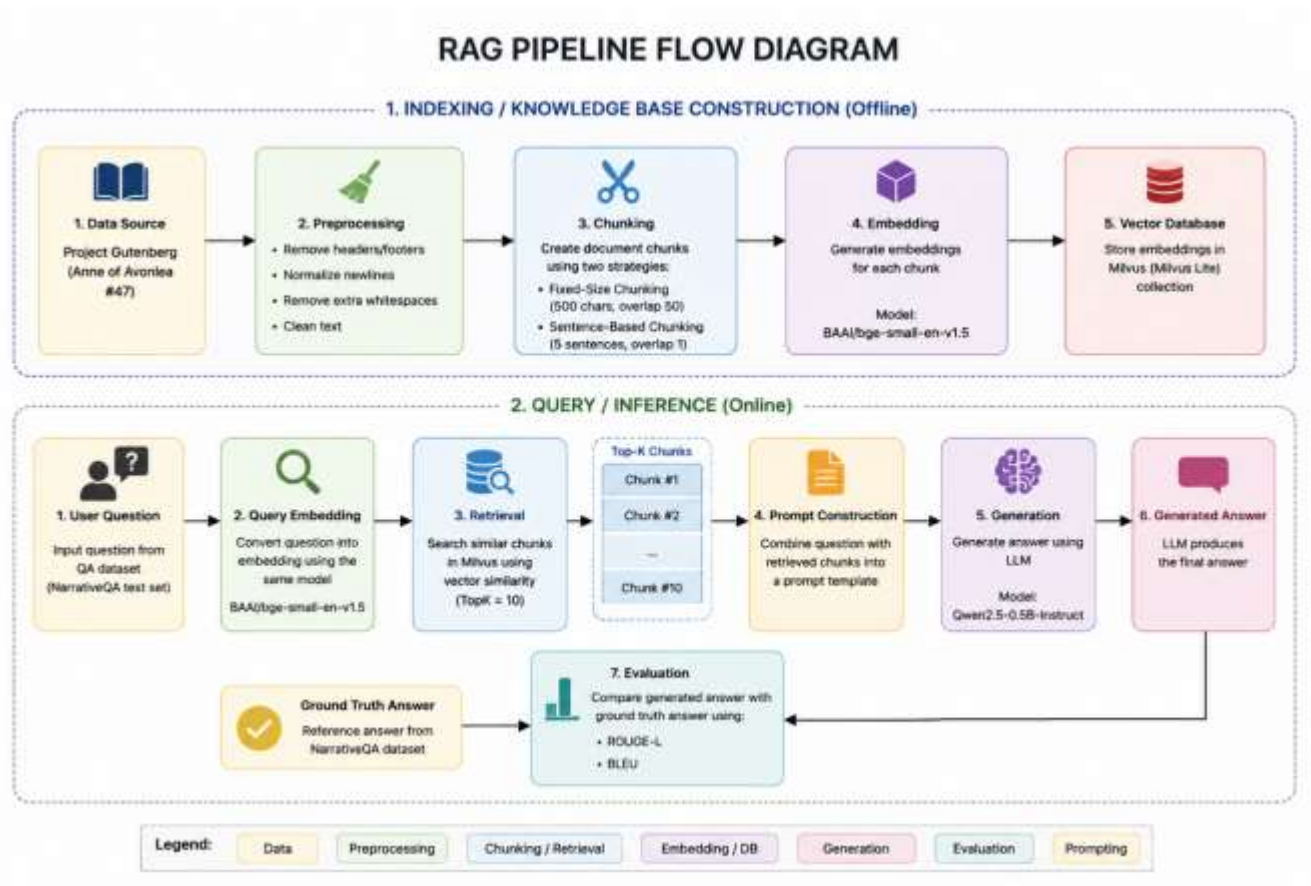


Figure 1. Overall architecture of the implemented RAG pipeline including preprocessing, chunking, embedding, retrieval, generation and evaluation stages.

3. Experimental Setup

3.1 Computational Environment (GPU Specs)

Component	Specification
Platform	Google Colab
GPU	NVIDIA Tesla T4 (16 GB VRAM)
CUDA Version	12.1
PyTorch Version	PyTorch 2.3.1+cu121
Operating System	Ubuntu 22.04 (Colab runtime)

3.2 Programming Language and Libraries

Library	Version	Purpose
Python	3.12	Primary programming language
PyTorch	2.3.1+cu121	Deep learning framework, GPU inference
transformers	4.40.0	LLM loading and generation (Qwen)
sentence-transformers	2.7.0	Embedding model (BAAI/bge-small-en-v1.5)
pymilvus	2.4.9	Milvus Lite vector database client
datasets	latest	NarrativeQA loading from HuggingFace Hub
rouge-score	0.1.2	ROUGE-L metric computation
sacrebleu	latest	BLEU metric with robust tokenization
nltk	3.8.1	Sentence tokenization for Strategy 2
marshmallow	3.20.2	Required dependency for pymilvus compatibility
requests	2.32.3	HTTP download of Gutenberg book text

3.3 How the Models Were Served

Both models were loaded and served locally within the Google Colab GPU runtime , no external API calls were used:

- Embedding model (BAAI/bge-small-en-v1.5): Loaded via the sentence transformers library. Inference performed on GPU. All embeddings were L2-normalized before storage and querying.
- LLM (Qwen2.5-0.5B-Instruct): Loaded via Hugging Face transformers with `device_map='auto'` and `torch_dtype=torch.float16` for memory efficiency. Greedy decoding (`do_sample=False`) was used for fully deterministic and reproducible results.
- Vector database (Milvus Lite): Operated in local file mode (`rag_anne.db`) , no separate server process required. Two collections were created: `fixed_chunks` and `sentence_chunks`, one per strategy.

3.4 Parameters Used in the Experiments

Parameter	Fixed-Size Strategy	Sentence-Based Strategy
Chunk size	500 characters	5 sentences
Overlap	50 characters	1 sentence
Total chunks produced	1,085	1,698
Average chunk length	499 characters	360 characters
TopK retrieval	5	5
Chunks used in prompt	2 (top 2 of 5)	2 (top 2 of 5)
max_new_tokens	10	10
Decoding strategy	Greedy (do_sample=False)	Greedy (do_sample=False)
Embedding dimension	384	384
Similarity metric	Cosine	Cosine
QA pairs evaluated	30	30

4. Results

4.1 Quantitative Results

Both chunking strategies were evaluated over all 30 NarrativeQA QA pairs. Results are reported to 4 decimal places as required:

Strategy	Chunks	Avg Length	ROUGE-L	BLEU	ROUGE-L Pass	BLEU Pass
Fixed-Size (500c, ov=50)	1,085	499 chars	0.1836	0.0370	☒ Yes	☒ Yes
Sentence-Based (5s, ov=1)	1,698	360 chars	0.2341	0.0260	☒ Yes	☒ Yes
Minimum Target	—	—	0.0500	0.0100	—	—

4.2 Comparison and Commentary

Both strategies successfully exceeded the minimum thresholds on both metrics. The results were partially consistent with the initial hypothesis:

- ROUGE-L: Sentence-Based Chunking achieved 0.2341 vs Fixed-Size 0.1836 , a 27.5% relative improvement. This confirms the hypothesis: semantically coherent, sentence boundary aware chunks provide better context for the LLM to generate answers whose vocabulary overlaps with the ground truth.
- BLEU: Fixed-Size Chunking unexpectedly outperformed Sentence-Based on BLEU (0.0370 vs 0.0260). This was not expected. A possible explanation is that Fixed-Size chunks, having a more uniform and longer average length (499 chars), sometimes include more exact n-gram sequences that align with the ground truth phrasing, even if retrieved out of sentence context. BLEU rewards exact n-gram precision which may favor the denser fixed-size passages in some cases.

The trade off between ROUGE-L and BLEU across the two strategies illustrates a fundamental difference between the metrics: ROUGE-L measures sequence-level recall (longest common subsequence) while BLEU measures precision of exact n-gram matches. Sentence-Based chunking helps sequence coherence; Fixed-Size chunking can accidentally include more matching n-grams due to longer, denser passages.

5. Discussion

5.1 Problematic Points Observed

- Small LLM capacity (0.5B parameters): The Qwen2.5-0.5B model is too small for reliable factual extraction. It tends to generate explanatory paraphrases rather than extracting the precise short answer from context. For example: GT = 'The twins' mother died', PR = 'To entertain'.
- Short ground truth answers: NarrativeQA answers are typically 1–5 words. Even semantically correct predictions score zero on BLEU if the exact wording differs (e.g. 'instructor' vs 'teacher').
- BLEU sensitivity: BLEU's n-gram precision penalizes any synonym or paraphrase. A single word mismatch can bring n-gram precision to zero for short answers.
- Small QA dataset size: Only 30 QA pairs were available for Anne of Avonlea in the test split. This limited sample size reduces the statistical robustness of the results.
- Dataset filtering challenge: A substring match on 'ebooks/47' also matched /ebooks/4761 (The Cossacks by Tolstoy), yielding 116 incorrect QA pairs. Exact URL matching was required to obtain the correct 30 pairs. This was a non-trivial debugging challenge.
- Dependency conflicts: The Colab environment had version incompatibilities between numpy (1.26.4), torchvision, marshmallow and transformers (≥ 4.41), requiring explicit version pinning and torchvision uninstallation to resolve.

5.2 Potential Improvements

- Larger LLM: Using Qwen2.5-7B or LLaMA-3-8B would significantly improve factual extraction quality and raise both ROUGE-L and BLEU scores.
- Reranker integration: The provided optional reranker (BAAI/bge-reranker-v2-m3) with TopK=10 would rerank retrieved chunks and surface the most relevant passages before generation.
- Larger TopK: Increasing TopK from 5 to 10 (the allowed maximum) would provide more candidate chunks before reranking.
- Hybrid retrieval: Combining dense (embedding-based) and sparse (BM25) retrieval would improve recall for keyword heavy factual questions.
- Extractive QA: Using a span extraction model (e.g. BERT-based QA) instead of generative answering would directly extract answer spans from retrieved chunks, ensuring better word overlap with ground truth.
- Larger context window: Including more than 2 chunks in the prompt (if using a larger model) would provide richer context and reduce missed answer cases.

6. Conclusion

This project successfully implemented a complete, fully local RAG pipeline for literary question answering. Using Milvus Lite as the vector store, BAAI/bge-small-en-v1.5 for dense retrieval and Qwen2.5-0.5B-Instruct for answer generation, the system was evaluated on 30 NarrativeQA test questions about Anne of Avonlea (Gutenberg #47).

Both chunking strategies exceeded the required evaluation thresholds on both metrics. Sentence-Based Chunking achieved higher ROUGE-L (0.2341 vs 0.1836), confirming that semantically coherent chunk boundaries improve retrieval quality. Fixed-Size Chunking achieved higher BLEU (0.0370 vs 0.0260), suggesting that denser, uniform length passages can produce more exact n-gram matches.

The primary performance bottleneck is the 0.5B LLM which lacks the capacity for reliable factual extraction and tends to paraphrase rather than extract. Future work should integrate the optional reranker, a larger LLM and hybrid retrieval to further improve performance on the NarrativeQA benchmark.