Advanced RAG Task:
Hierarchical Retrieval from Literature on Constrained Resources

1 Introduction

The objective of this task is building a Retrieval-Augmented Generation (RAG) system. This RAG system is evaluated using google/gemma-3-1b-it LLM. The evaluation results are compared with pure model just using google/gemma-3-1b-it LLM. The comparison is reported. The RAG system is based on a book from Project Gutenberg which is the knowledge source. It is indexed using hierarchical chunking into an on-disk database.

For this task, candidate books are identified from the NarrativeQA dataset. "The Tale of Two Bad Mice" is chosen. The story is indexed using character-based hierarchical chunking into Qdrant on-disk vector database. Both the development and the evaluation run on a Google Colab free tier instance with a T4 GPU.

2 Approach & Methodology

First of all, the candidate books are examined through the Gutenberg Project and the NarrativeQA dataset simultaneously. "The Tale of Two Bad Mice" is selected since it is a children's book. Due to its nature, it is short and simple. It is a perfect choice regarding the resource constraints. Its document id from the NarrativeQA dataset is "08c3429e9717bc663bc5f41cd3a2c701c222ed2f". The story URL from Gutenberg Project is "http://www.gutenberg.org/ebooks/45264.txt.utf-8". The plain text version is downloaded from the story URL. Metadata from the Gutenberg project, headers and footers are cleaned. In addition, illustration lines are removed. Then, extra whitespaces are cleaned. The NarrativeQA dataset is filtered in order to only keep related question and answers with the book. The final test set contains 30 QA pairs.

For hierarchical chunking different strategies are applied such as character based, word based and token based using an embedding model. The best results are obtained using character-based. The advantages of this method are simple implementation, compatibility with long texts and easy to control exact chunk size. When the same embedding model used in the next steps is used for tokenization, it is impossible to use even mid-size chunks due to its maximum allowed input length which is 256 tokens. The parent size, the child size and overlap are set as 1000, 200 and 20 respectively as the total story length is almost 5k. 5 parents and 28 children are obtained in the end of this process.

The chosen sentence-transformer model is all-MiniLM-L6-v2. The main reason of this selection is its size. It is small and fast. It gives high quality results compared to its size. Therefore, it is appropriate for the Colab free tier.

Qdrant is selected as the vector database for various reasons. It works fully on-disk. It is compatible with Colab. It has simple Python API. It performs well at small-to-medium scale.

Retrieval stretegy is as follows. Since it is a small story, k is set to 3. It gives a balanced selection between precision and noise. All parents are retrieved for all top-k children in order to give broader narrative context which is essential for short story with faster context flow across chunks.

Finally, the prompt is designed to restrict the model to give answers in 1-3 words. Since it is a short children's story, the answers to the questions are really simple. Without giving restriction

to the model answers become long sentences and even paragraphs. Similar prompt is given for the baseline and the RAG system. The RAG one obviously includes the retrieved context.

3 Implementation Details

Key libraries are sentence-transformers, qdrant-client, transformers, evaluate, rouge_score and sacrebleu.

The biggest challenge is to implement chunking part since session crashes occur due to limited resources. It is important to implement fast and optimized strategy. Therefore, chunks are created without heavy regex. In addition, embeddings are generated one-by-one instead of batches.

4 Results & Discussion

The evaluation results of RAG compared with the no RAG baseline is shown in Table 1. There is a significant difference between the results. RAG improves the results. Since it is short story and the ground-truth answers are short and clear, the importance of the knowledge-base is even more visible. Baseline responses are generally more generic and common-sense answers as shown in the examples. RAG answers are based on the story. ROUGE-L scores are lower than BLEU-4 scores as it penalizes paraphrased outputs due to its order matching nature.

Here are some examples.

- Question: Where do Tom Thumb and Hunca Munca live?
  Reference: ['Under the skirting board.', 'Under the skirting board.']
  Baseline: Hungary.
  RAG: Doll's-house.
- Question: What does the nurse set up?
  Reference: ['A mouse trap.', 'A mouse-trap']
  Baseline: IV fluids
  RAG: Mouse trap.
- Question: Who owns the doll house?
  Reference: ['The little girl', 'Lucinda and Jane.']
  Baseline: Estate.
  RAG: Lucinda and Jane.

Peak RAM usage is 2475.0 MB. Qdrant DB size is 0.13 MB. Inference time is 0.24 seconds per question for baseline and 0.35 seconds per question for RAG.

Metric	**Baseline (No RAG)**	**RAG (Hierarchical)**
BLEU-4 (vs. Ground Truth)	1.4639	14.1561
ROUGE-L (vs. Ground Truth)	0.0407	0.3042
Notes / Qualitative Resource Impact	0.24 seconds per question inference	0.35 seconds per question inference, extra memory & time for indexing

Table 1: Results of RAG compared with no RAG baseline.

5 Conclusion

RAG improved the results significantly. If resources allowed, bigger embedding model can be applied. In that case, instead of character-based chunking, token-based chunking can be applied. On top that, re-ranking algorithm can be implemented in order to refine retrieval results. Finally, larger LLM model can be used to improve the results even more.