Evaluation of Retrieval-Augmented Generation (RAG) for Code Search

1. Baseline Performance (Regular RAG with No Modifications)

The initial setup involved splitting the code into chunks and embedding them using `text-embedding-small`. Using similarity retrieval, this approach achieved the following:

- **Best-performing setup:** Chunk size 700, chunk overlap 100.
- **Recall@10:** 63.7%.

Decent performance, but significant room for improvement.

---

2. Using LLM Summaries for Embeddings

Since queries are in natural language, code was summarized using an LLM before embedding. Although this made vector store creation much slower, it improved retrieval accuracy:

- **Chunk size 700, overlap 100 → Recall@10: 67.6%.**

Better than the baseline, but still far from optimal.

---

3. Experimenting with Query Expansion (Did Not Work Well)

Expanding queries using different prompt templates introduced several drawbacks:

- Inference time increased significantly.
- Accuracy dropped across all chunk sizes and overlaps.
- Best attempt (shortened expanded queries) reached only Recall@10: 64.3%, still worse than the original queries.

Conclusion: Query expansion is not worth it for this task.

---

4. Improving Summarization & Using a Bigger Embedding Model

To further enhance retrieval:

- A better summarization prompt with more metadata was used.
- Switched from `text-embedding-small` to `text-embedding-large` (slightly more expensive but potentially more effective).

Results:

- **Chunk size 1100, overlap 100 → Recall@10: 69.5%** (Best for this setting).
- **Chunk size 700, overlap 100 → Recall@10: 70.2%** (Best overall).

This was the most effective setup, achieving the highest retrieval accuracy at this stage.

---

5. Further Enhancements & Best Achieved Performance

After including file extensions in the dataset, accuracy improved. Additional experiments with adding code snippets alongside the summaries led to a major improvement:

- **Chunk size 1200, overlap 200 → Recall@10: 80.9%**.

  Enhancing metadata by including a list of functions and classes present in each file resulted in:

- **Recall@10: 84.6%**.

  Finally, fixing an error in the querying process boosted accuracy further:

- **Recall@10: 85.8%**.
- **Recall@40: 89.3%** (Suggesting a more expensive re-ranking system could further improve results).

Best RAG system so far, with a significant improvement over the baseline.

---

6. Experimenting with an LLM-Based Re-Ranker

An experiment with an LLM-based re-ranker produced mixed outcomes:

- Inference time increased significantly, making the system less practical for real-time applications.
- Accuracy dropped slightly, likely due to the re-ranker introducing noise or misaligning with the retrieval objectives.

Conclusion: The LLM-based re-ranker was not effective for this task and is not recommended.

---

7. Experiments with Other Languages

During evaluation, some files in the dataset were in Russian. Key observations include:

- Translated queries in Russian retrieved the exact Russian README file that was previously missed, but failed to retrieve the corresponding English README file.
- Overall accuracy with Russian queries was **83.3%**, slightly worse than with English prompts (**85.8%**).

Conclusion: Could be useful in specific cases but shouldn't make a difference overall.

8. Experiments with Contextual Embeddings

Contextual embeddings were tested to evaluate their impact on retrieval accuracy. While they introduced additional complexity, they provided mixed results:

- **Indexing time increased** significantly due to the added complexity of contextual embeddings.
- **Recall@10 dropped slightly** to **81.2%**, while **Recall@50 improved** to **94.7%**.
- Further experiments revealed that contextual embeddings were more useful for **generation tasks rather than retrieval**.
- The original approach of appending the code summary at the beginning of the file was better suited for the retrieval task since it produced both a chunk in natural language (NL) and code chunks, allowing the retriever to access both.

Conclusion: Contextual embeddings are beneficial for generation tasks but not optimal for direct retrieval at the file level.

---

9. Refining the Summary Prompt for Better Retrieval

Upon analyzing retrieved and missed files, it became clear that understanding **the overall purpose of a file** determined by both code and file path could help improve retrieval accuracy. The summary prompt was refined accordingly:

- **Recall@10 improved to 88.8%**—the highest achieved performance.
- **Recall@40 reached 94.5%**, suggesting further improvements could be achieved with a more sophisticated re-ranking system.

Conclusion: Refining the summary prompt by incorporating file path information and code-based metadata significantly improved retrieval accuracy.

Key Insights:

- The RAG system can handle multilingual content effectively.
- There is a trade-off in accuracy when translating queries.
- For multilingual datasets, a hybrid approach (e.g., querying in both languages and merging results) might be worth exploring.
- Contextual embeddings arent worth it for the task at hand

---

10. Summary of Findings

---

11. Final Conclusion & Next Steps

- **Chunk size:** 700 with 100 overlap performed well initially, but 1200/200 provided the best final performance.
- **LLM Summaries:** Summarizing code with an LLM improves retrieval but increases indexing time.
- **Embedding Model:** Switching to a larger embedding model further boosts recall.
- **Metadata:** Adding metadata and fixing query processing errors led to the highest accuracy gains.

Method	Best Chunk Size	Best Overlap	Recall@10 (%)
Regular RAG (no modifications)	700	100	63.7
With LLM-generated summaries	700	100	67.6
With Query Expansion	700	100	64.3
Improved summaries + larger embedding model	1100	100	69.5
Improved summaries + large embedding model	700	100	70.2
With additional extensions + code snippets	1200	200	80.9
With enhanced metadata (functions & classes)	1200	200	84.6
Optimized RAG (query fix applied)	1200	200	85.8
Final optimized RAG (file purpose)	**1200**	**200**	**88.8**
With LLM-based re-ranker	1200	200	78.0
With translated queries (Russian)	1200	200	83.3

Table 1: Summary of Experimental Findings

- **LLM-based Re-Ranker:** Increased inference time while reducing accuracy, making it unsuitable for this task.
- **Multilingual Handling:** The system can handle multilingual content, but translating queries may slightly reduce accuracy. A hybrid approach could be explored.
- **Contextual embeddings:** Contextual embeddings are not worth it for the task at hand.

NOTE: The scope of this RAG system was limited to a single code base: https://github.com/viarotel-org/escrcpy/blob/main/.nvmdrc. The tests and experiments conducted are not the best benchmarks for the techniques used; however, they still provide valuable insights. For future development and accuracy improvements, a larger dataset would definitely be necessary.