Designing a RAG-Based Q&A System for News Retrieval

Organizers

The project is organized by:
Prof. em. Dr. Gerd Kortemeyer, Michigan State University; Rectorate and AI-Center, ETH Zurich
Dr. Susie Xi Rao, Senior Researcher, ETH Zurich
Dr. Guang Lu, Lecturer for Data Science, Lucerne University of Applied Sciences and Arts
in collaboration with:
Dr. Diego Antognini, Senior Research Scientist, Google DeepMind
as a joint task of industry and academia for the HSLU Applied Information and Data Science Master's Program in the course Advanced Generative AI.

Objective

This project requires students to build a Retrieval-Augmented Generation (RAG) system that answers user queries based on the ETH News Dataset stored in .html format. The project is structured into three major phases:

- Data Preparation (10 points) – Structuring the dataset
- Building the RAG System (60 points, split into two steps):
    - Research Agents (30 points) – Implementing multiple pre-retrieval and retrieval strategies
    - Aggregation and Response Synthesis (30 points) – Merging retrieved results and applying advanced post-retrieval (e.g. re-ranking) techniques
- Evaluation (5 points) – Assessing answer quality through both automated and human evaluation

Deadlines and Deliverables

Students must submit:

- Codebase: A well-documented implementation (preferably in Python)
- Evaluation Report: Analysis of retrieval and generation performance within e.g. the Python notebook

The students form groups of size three and submit their solutions by *15 June 2025* at the latest, passing on the codebase and the evaluation report to the submission link (https://docs.google.com/spreadsheets/d/19pbPb6FJRhnf4dqsBA2yzxqFstjzanj6ZNnAFtwv8dQ/edit?gid=0#gid=0) provided by Dr. Diego Antognini. The contribution of each individual group member must be clearly stated in the submission.

Guidance and Requirements

Step 1 Data Preparation – Structuring the dataset

Objective:

Students will prepare a structured dataset of German and English news articles by extracting and cleaning text from HTML files, enriching it with language-aware metadata, and storing it in a format suitable for future retrieval in a RAG system.

Tasks and Requirements:

- Loading, Parsing, and Cleaning HTML Files (5 Points)
    - Use BeautifulSoup (https://beautiful-soup-4.readthedocs.io/en/latest/) to extract the main text from .html files while removing unnecessary elements such as JavaScript, CSS, and HTML tags
    - Use Docling (https://github.com/docling-project/docling) for advanced document parsing, especially if handling potentially complex layouts, tables, or non-standard structures
    - Implement a hybrid approach (BeautifulSoup + Docling) to compare the effectiveness of both tools and optimize extraction quality

- Multilingual Text Preprocessing and Cleaning (5 Points)
    - Perform necessary text preprocessing (e.g., removing extra spaces and redundant line breaks, normalizing Unicode characters, standardizing date formats from different sources), and handle German-specific text processing (e.g., compound words, umlaut normalization if needed)
    - Store the cleaned text and its metadata in a structured format suitable for retrieval (e.g., JSON, CSV, or a database) with fields such as language, title, date, source, main content, named entities, topics, keywords, summary
    - Create additional rich metadata that can support future semantic search and context filtering to enhance document retrieval later, and store metadata in a structured database (e.g., SQLite, Pandas DataFrame, or JSON format) for efficient access

Deliverables:

- A brief report within the Python notebook explaining the parsing choices and comparing the pros and cons between BeautifulSoup and Docling
- A clean and structured multilingual dataset of news articles with enriched metadata, ready for indexing and retrieval in a future RAG-based system

Step 2.1 Building the RAG System: Research Agents – Implementing multiple pre-retrieval and retrieval strategies

Objective:

Students will implement and compare different pre-retrieval and retrieval strategies to extract relevant candidate documents. These retrieved documents will go through the post-retrieval process in Step 2.2.

Retrieval techniques:

- Traditional keyword-based retrieval (BM25, baseline)
    - https://python.langchain.com/docs/integrations/retrievers/bm25/
    - https://docs.llamaindex.ai/en/stable/examples/retrievers/bm25_retriever/
    - https://huggingface.co/blog/xhluca/bm25s
- Semantic search-based retrieval (Dense retrieval using embeddings)
    - https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/
    - https://docs.llamaindex.ai/en/stable/api_reference/retrievers/vector/
    - https://github.com/LuciferUchiha/Cleantech-RAG
- GraphRAG-based retrieval (Following Microsoft's Local-to-Global GraphRAG approach)
    - https://microsoft.github.io/graphrag/

    - https://github.com/microsoft/graphrag
    - https://python.langchain.com/docs/integrations/retrievers/graph_rag/
    - https://docs.llamaindex.ai/en/stable/examples/cookbooks/GraphRAG_v1/
- Hybrid retrieval (Combining BM25 + Dense Search + GraphRAG)

Tasks and Requirements:

- Data Preprocessing and Benchmark Construction (5 Points)
    - Chunk the preprocessed news text data using fixed-size segmentation (e.g., 512 tokens) or semantic segmentation (detect topic shifts using embeddings)
    - Prompt e.g. GPT-4o to determine if a paragraph answers a given question in the benchmark Q&A dataset
    - Assign numerical scores to generate ground-truth relevance labels to evaluate retrieval quality: fully answers → 1.0, partially answers → 0.5, not relevant → 0.0
- Implement Multiple Retrieval Strategies (20 Points, 5 Points each task)
    - Baseline: Multilingual Keyword-Based Retrieval (BM25) – Index documents in their original language (EN or DE), detect query language, translate the query to the other language, and search both EN and DE documents using BM25 for cross-lingual retrieval
    - Semantic Search: Multilingual Dense Vector Retrieval – Convert text into multilingual vector embeddings (using models like mBERT, Sentence-BERT, OpenAI, Gemini, etc.), store embeddings in FAISS, Pinecone, or ChromaDB, and retrieve documents via cosine similarity, supporting both English and German queries
    - GraphRAG-Based Retrieval (Multilingual Approach) – Follow the methods described by Microsoft Research (https://arxiv.org/pdf/2404.16130), adapting them for multilingual retrieval by incorporating cross-lingual document indexing and query translation to enable retrieval from both English and German documents
    - Hybrid Retrieval (BM25 + Dense + GraphRAG) – Retrieve top-*k* results using BM25 and multilingual semantic search, merge and deduplicate results, and apply GraphRAG to identify additional relevant nodes from both English and German documents
- Evaluate Retrieval Performance (5 Points)
    - Evaluating retrieval methods using standard ranking metrics such as Precision@*k* (fraction of retrieved documents that are relevant), Recall@*k* (fraction of all relevant documents retrieved), and MRR (Mean Reciprocal Rank) (measures how soon a relevant document appears)
    - Note how the typical pre-retrieval approaches such as query routing, query rewriting, and query expansion influence the retrieval performance

Deliverables:

- Structured dataset of news paragraphs with benchmark relevance scores
- Hybrid retrieval pipeline integrating multilingual BM25, Semantic Search, and GraphRAG
- A report comparing the pre-retrieval and retrieval strategies, discussing the best-performing approach and why, how GraphRAG expands relevant context, and the performance of hybrid retrieval vs. standalone methods

Step 2.2 Aggregation and Response Synthesis – Merging retrieved results and applying advanced post-retrieval (e.g. re-ranking) techniques

Objective:

Students will enhance the results from Step 2.1 using post-retrieval, particularly the re-ranking models. The goal is to reorder the retrieved documents based on relevance to the query, ensuring that the most relevant information is passed to the Large Language Model (LLM). Students will apply state-of-the-art re-ranking techniques and evaluate the performance of their methods.

Tasks and Requirements:

- Implement Re-ranking Models (20 points)

    - Integrate re-ranking models into your RAG pipeline, apply models such as EcoRank (https://arxiv.org/abs/2402.10866), Set-Encoder (https://arxiv.org/abs/2404.06912), Multislot Reranker (https://arxiv.org/abs/2401.06293), List-Aware Reranking (https://dl.acm.org/doi/abs/10.1145/3589334.3645336), and DSLR (https://arxiv.org/abs/2407.03627), or open-source re-ranking tools like Rankify (https://arxiv.org/abs/2502.02464) to reorder the results of Step 2.1
    - Use pre-built re-ranking solutions (like OpenAI or Cohere) for comparison
    - Integrate with other useful post-retrieval approaches such as summary and fusion
- Evaluate Re-Ranking Performance (10 Points)
    - Compare re-ranking models and other post-retrieval approaches with the baseline retrieval performance from Step 2.1 using the metrics such as Precision@*k*, Recall@*k*, and MRR, and track the improvements (e.g., more relevant documents, better context alignment)
    - Assess computational efficiency of re-ranking models in comparison to baseline (important for scalability)
    - Conduct a qualitative analysis, check if the reranked documents have better context alignment with the query, and ensure the most relevant documents are now presented in the correct order for query response generation

Deliverables:

- Code notebook for integrated re-ranking models and other post-retrieval approaches in the RAG system
- Performance metrics (e.g. Precision@*k*, Recall@*k*, MRR) and visualizations comparing re-ranking models to the baseline
- Qualitative analysis on the relevance improvement and correct ordering of documents for query answering

Step 3 Evaluation – Assessing answer quality through both automated and human evaluation

Objective:

Students will evaluate the quality of the answers generated by their best RAG pipeline. They will apply the pipeline to benchmark questions, compare the generated answers with the provided ground truth answers, and evaluate the system's performance using both automated metrics and human evaluation.

Tasks and Requirements:

- Run the Best RAG Pipeline: Use the RAG pipeline that has been developed, applying it to the benchmark question set to generate answers
- Automated Metrics Calculation:
    - Calculate semantic Exact Match (matching results based on semantic similarity rather than exact words, using embeddings or semantic models to align underlying meaning), semantic F1 Score (tokenize predicted and reference answers, then calculate precision, recall, and F1 based on semantic matches using embeddings, not exact tokens), and BLEU or ROUGE (these are common text comparison metrics that measure how much the generated answer overlaps with the ground truth in terms of *N*-grams or sequence patterns) for each generated answer and compare with the ground truth answer
    - Record these metrics in a structured manner for comparison
- Human Evaluation:
    - Based on the benchmark questions and the ground truth answers, manually evaluate the quality of answers, considering Relevance (how well the answer matches the user's query), Correctness (how accurate the answer is based on the ground truth), and Clarity (how clear and understandable the generated answer is)
    - Use a rating scale (1-5) for each dimension (Relevance, Correctness, and Clarity)
    - Provide a brief written analysis based on the human evaluation results
- Report and Presentation:
    - Present both automated and human evaluation results in a concise format (tables, charts, etc.) in the code notebook

- Summarize findings, such as how well your pipeline performed in comparison to the ground truth answers and any potential areas for improvement

Deliverables:

- Automated Metrics: Calculate the following metrics for the answers generated by the RAG pipeline: (1) Semantic Exact Match, (2) Semantic F1 Score, (3) BLEU or ROUGE
- Human Assessment of Answer Quality: Review the answers generated by your RAG system manually to assess their Relevance, Correctness, and Clarity. This evaluation is subjective and will involve a small group of reviewers (could be your peers or teammates)

References

- RAG-Driven Generative AI: Build custom retrieval augmented generation pipelines with LlamaIndex, Deep Lake, and Pinecone (https://github.com/Denis2054/RAG-Driven-Generative-AI)
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997
- Zhou, Y., Liu, Y., Li, X., Jin, J., Qian, H., Liu, Z., ... & Yu, P. S. (2024). Trustworthiness in retrieval-augmented generation systems: A survey. arXiv preprint arXiv:2409.10102
- Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. arXiv preprint arXiv:2501.09136
- https://www.youtube.com/watch?v=mE7IDf2SmJg (Stanford CS25: V3 I Retrieval Augmented Language Models)
- https://research.trychroma.com/evaluating-chunking (Evaluating Chunking Strategies for Retrieval)
- https://relevanceai.com/prompt-engineering/implement-implicit-rag-for-better-ai-responses (Implement Implicit RAG for Better AI Responses)
- https://www.datacamp.com/tutorial/contextual-retrieval-anthropic (Anthropic's Contextual Retrieval: A Guide With Implementation)
- Lee, J., Dai, Z., Ren, X., Chen, B., Cer, D., Cole, J. R., ... & Naim, I. (2024). Gecko: Versatile text embeddings distilled from large language models. arXiv preprint arXiv:2403.20327
- Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024, November). Retrieval augmented generation or long-context LLMs? A comprehensive study and hybrid approach. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track (pp. 881-893)
- Jin, B., Yoon, J., Han, J., & Arik, S. O. (2024). Long-context LLMs meet RAG: Overcoming challenges for long inputs in RAG. arXiv preprint arXiv:2410.05983
- Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024, May). Raptor: Recursive abstractive processing for tree-organized retrieval. In The Twelfth International Conference on Learning Representations
- Lewis, P., Wu, Y., Liu, L., Minervini, P., Küttler, H., Piktus, A., ... & Riedel, S. (2021). PAQ: 65 million probably-asked questions and what you can do with them. Transactions of the Association for Computational Linguistics, 9, 1098-1115
- Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., ... & Poli, I. (2024). Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. arXiv preprint arXiv:2412.13663