A Scalable Retrieval-Augmented Generation Pipeline for Domain-Specific Knowledge Applications

Emmanuel A. Olanrewaju

DOI: https://doi.org/10.51584/IJRIAS.2025.1010000014

Received: 26 September 2025; Accepted: 02 October 2025; Published: 28 October 2025

This work presents a Retrieval-Augmented Generation (RAG) pipeline that integrates document preprocessing, embedding-based retrieval, and large language model (LLM) generation into a unified framework. The pipeline begins with the ingestion of PDF documents, followed by text cleaning, sentence segmentation, and chunking to ensure compatibility with embedding model constraints. High-dimensional vector representations are generated using transformer-based embedding models and stored for downstream use. Semantic similarity search, implemented via dot product and cosine similarity, enables efficient retrieval of contextually relevant text. For scalability, the framework is designed to accommodate vector indexing methods such as Faiss. On the generation side, locally hosted LLM (Gemma-7B) is employed with optional quantization for reduced resource consump- tion. Retrieved context is integrated with user queries to enhance the accuracy and relevance of generated responses. This pipeline demonstrates a practical approach for building domain-specific, retrieval-augmented applications that balance efficiency, scalability, and adaptability to local com- pute environments.

INTRODUCTION

Digital transformation signifies the integration of dig- ital technology across various facets of a business, re-shaping both its operations and the delivery of value to customers [1]. At the forefront of this transforma- tion are Large Language Models (LLMs), advanced ma- chine learning models trained extensively on textual data to understand and generate human-like text [1]. LLMs, such as the Generative Pre-training Transformer (GPT) series [2, 3] and others, have demonstrated remarkable capabilities in natural language processing (NLP) tasks [4]. Despite these advances, LLMs often struggle with domain-specific queries, generating inaccurate or irrele-vant information—commonly referred to as "hallucina- tions"—especially when data is sparse [5]. Consequently, applying LLMs in practical scenarios is challenging, as their outputs cannot always be relied upon for accuracy.

Pre-trained models are capable of learning substantial amounts of in-depth knowledge directly from data [6], functioning as parameterized implicit knowledge bases without requiring external memory [7, 8]. While this capability is promising, such models have limitations: they cannot easily expand or revise their knowledge, can-not transparently explain their predictions, and remain prone to hallucinations [9]. Hybrid approaches that com-bine parametric memory with non-parametric, retrieval- based memory [10–12] address some of these limitations by allowing knowledge to be updated dynamically and providing transparency in the retrieved information.

Retrieval-augmented generation (RAG) has emerged as a powerful approach in NLP, effectively combining the strengths of retrieval and generative models [13]. RAG has been applied to reduce hallucinations, provide knowledge grounding, and enable personalization [14– 17]. Evaluating RAG systems is crucial for ensuring the effectiveness of integrating retrieval-based methods with generative models [18, 19]. Traditionally, such evalua-tion relies on end-to-end assessment, comparing gener- ated outputs with one or more ground truth references [20]. While essential, this approach presents limitations, particularly for assessing the performance of the retrieval component in RAG systems.

In this work, we present a Retrieval-Augmented Gen- eration (RAG) pipeline that integrates document pre-processing, embedding-based retrieval, and LLM gener- ation into a cohesive framework. The pipeline begins with the ingestion of PDF documents, followed by text cleaning, sentence segmentation, and chunking to ensure compatibility with embedding model constraints. High- dimensional vector representations are generated using transformer-based embedding models and stored for ef- ficient downstream retrieval. Semantic similarity search, implemented via dot product and cosine similarity, en- ables rapid identification of contextually relevant text, while vector indexing methods such as Faiss provide scal- ability for large document collections.

On the generation side, locally hosted LLM (Gemma- 7B) is employed, with optional quantization to reduce computational resource requirements. Retrieved con- text is combined with user queries to enhance the rele- vance and accuracy of generated responses. This pipeline provides a practical approach for constructing domain-specific, retrieval-augmented applications that are effi- cient, scalable, and adaptable to local compute environ- ments, establishing a foundation for robust knowledge- driven systems.

METHODOLOGY

This study implements a RAG pipeline by following a structured computational workflow developed in Python. The methodology consists of several sequential stages: data acquisition, preprocessing, embedding generation, retrieval system design, large language model integration, and evaluation using specialized metrics. Each stage is implemented as described below.

Data Acquisition and Parsing

The PDF for this work was downloaded from Press- books OER Hawaii Human Nutrition. Raw data was ingested from PDF documents using the fitz library (PyMuPDF) [21]. This parser was selected due to its ro- bustness in extracting structured and unstructured text compared to alternatives such as PyPDF [22]. The ex- tracted text was organized into structured tabular form using pandas [23] and datasets [24] to facilitate fur- ther downstream processing. Metadata such as docu- ment name, section identifiers, and chunk indices were preserved.

Text Preprocessing

To prepare the text for embedding and retrieval, sev- eral preprocessing steps were executed:

Tokenization and sentence boundary detection were performed using the spacy.lang.en tokenizer [25].

Text was segmented into coherent chunks to bal- ance retrieval efficiency and context length require- ments.

Cleaning procedures included lowercasing, removal of extraneous whitespace, and preservation of domain-specific entities.

This ensured consistency across documents while retain- ing semantic information critical for embedding.

Embedding Generation

Semantic embeddings were generated using the all-mpnet-base-v2 model implemented in the SentenceTransformer library [26]. A pre-trained transformer-based embedding model was employed to convert text chunks into dense vector representations. The sentence_transformers.util module was used for similarity computations, enabling effective nearest- neighbor retrieval. Embeddings were stored for rapid querying during RAG execution.

The choice of all-mpnet-base-v2 was mo- tivated by its balance between semantic rich- ness and computational efficiency.

Compared to alternative models such as MiniLM-L12-v2 or distilbert-base-nli-stsb-mean-tokens, it provides superior performance on sentence-level semantic similar- ity tasks. Despite challenges posed by domain-specific medical terminology, this model demonstrated robust generalization and strong clustering behavior, making it a suitable baseline for our pipeline.

Retrieval System Design

A vector-based retrieval system was constructed over the generated embeddings. Document similarity was computed using cosine similarity. This enabled the re- trieval of top-k semantically relevant chunks given a user query. The retrieval process was optimized with progress tracking using tqdm to monitor performance across large datasets.

Large Language Model Integration

A causal language model was integrated using the

transformers library [27]. Specifically:

Tokenization was handled by AutoTokenizer.

Model loading was performed with

AutoModelForCausalLM.

To improve memory efficiency, quantization was enabled using BitsAndBytesConfig, with conditional acceleration checks via is_flash_attn_2_available.

The retriever passed the top-ranked text chunks to the language model, enabling contextually grounded answer generation.

Evaluation Metrics

The RAG pipeline was systematically evaluated using the ragas library and its metrics [28]. The following quantitative metrics were computed:

Context Entity Recall: Measures the ability of retrieved passages to cover key entities relevant to the query.

Noise Robustness: Evaluates pipeline resilience to irrelevant or noisy inputs.

Additional evaluation functions from ragas.evaluate were applied to provide ag- gregate scores.

Timing benchmarks were recorded using time.perf_counter.

Workflow Summary

The entire methodology, from PDF ingestion through model response evaluation, was executed in a Jupyter Notebook environment. Each step was modularized into reproducible code cells to ensure transparency and reproducibility. The integration of preprocessing, retrieval, and generative modeling established a complete RAG pipeline suitable for experimentation with real-world text corpora.

RESULTS AND DISCUSSION

Document Processing and Text Chunking

The RAG pipeline successfully processed the Hu- man Nutrition textbook (2020 Edition) containing 1,208 pages. The text extraction and preprocessing pipeline demonstrated robust performance in handling PDF con- tent with varying structural elements. Statistical anal- ysis of the processed text revealed an average of 10.32 sentences per page with a standard deviation of 6.30, in- dicating consistent content density throughout the doc- ument.

The chunking strategy employing spaCy's sentence to- kenizer with a fixed size of 10 sentences per chunk proved effective. This approach resulted in 1,843 text chunks with an average token count of 183.61 (approximately 734 characters), well within the embedding model's 384- token capacity limit. The chunk size distribution showed:

Minimum chunk size: 12 characters (3 tokens)

Maximum chunk size: 1,831 characters (457 to- kens)

Interquartile range: 315-1,118 characters (78-279 tokens)

Filtering chunks with fewer than 30 tokens eliminated insignificant content such as page headers, footers, and isolated phrases, improving the quality of embeddings while maintaining contextual coherence.

Embedding Generation and Vector Representation

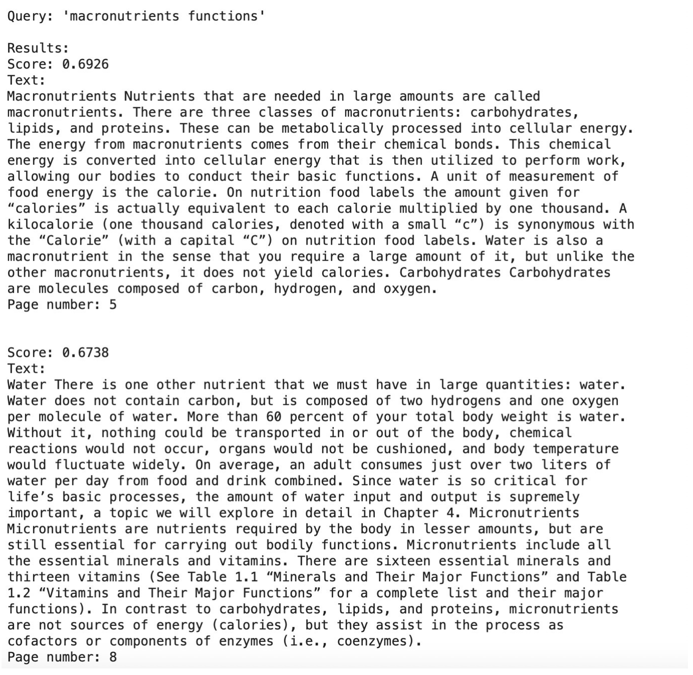
The sentence-transformers library with the all-mpnet-base-v2 model generated high-quality 768-dimensional embeddings for each text chunk. The embedding process successfully captured semantic relationships between nutritional concepts, as evidenced by the coherent clustering of related topics in the vector space, as illustrated in Figure 1. The model's 384-token input capacity was optimally utilized, with most chunks occupying less than 50% of available capacity, ensuring minimal information loss.

Preliminary qualitative analysis of embedding similar- ities revealed that:

Chunks discussing similar micronutrients (e.g., Vi- tamin A and Vitamin E) showed higher cosine sim- ilarity

Related physiological processes (e.g., digestion and absorption) formed natural clusters in the embed- ding space

Taxonomic relationships (e.g., macronutrients to their subcategories) were preserved in the vector representations



```
Query: 'macronutrients functions'

Results:
Score: 0.6926
Text:
Macronutrients Nutrients that are needed in large amounts are called
macronutrients. There are three classes of macronutrients: carbohydrates,
lipids, and proteins. These can be metabolically processed into cellular energy.
The energy from macronutrients comes from their chemical bonds. This chemical
energy is converted into cellular energy that is then utilized to perform work,
allowing our bodies to conduct their basic functions. A unit of measurement of
food energy is the calorie. On nutrition food labels the amount given for
“calories” is actually equivalent to each calorie multiplied by one thousand. A
kilocalorie (one thousand calories, denoted with a small “c”) is synonymous with
the “Calorie” (with a capital “C”) on nutrition food labels. Water is also a
macronutrient in the sense that you require a large amount of it, but unlike the
other macronutrients, it does not yield calories. Carbohydrates Carbohydrates
are molecules composed of carbon, hydrogen, and oxygen.
Page number: 5


Score: 0.6738
Text:
Water There is one other nutrient that we must have in large quantities: water.
Water does not contain carbon, but is composed of two hydrogens and one oxygen
per molecule of water. More than 60 percent of your total body weight is water.
Without it, nothing could be transported in or out of the body, chemical
reactions would not occur, organs would not be cushioned, and body temperature
would fluctuate widely. On average, an adult consumes just over two liters of
water per day from food and drink combined. Since water is so critical for
life’s basic processes, the amount of water input and output is supremely
important, a topic we will explore in detail in Chapter 4. Micronutrients
Micronutrients are nutrients required by the body in lesser amounts, but are
still essential for carrying out bodily functions. Micronutrients include all
the essential minerals and vitamins. There are sixteen essential minerals and
thirteen vitamins (See Table 1.1 “Minerals and Their Major Functions” and Table
1.2 “Vitamins and Their Major Functions” for a complete list and their major
functions). In contrast to carbohydrates, lipids, and proteins, micronutrients
are not sources of energy (calories), but they assist in the process as
cofactors or components of enzymes (i.e., coenzymes).
Page number: 8
```

FIG. 1: Similarity score of each chunk with respect to the query.

Computational Efficiency

The pipeline demonstrated efficient processing capabil- ities:

PDF text extraction: Completed in approximately 2 minutes for 1,208 pages

Sentence tokenization: Processed using spaCy's op- timized pipeline with minimal computational over- head

Embedding generation: Leveraged GPU accelera- tion when available, with average processing time of 100–200 chunks per minute

In addition, we compared the performance of the Gemma-7B LLM in both quantized and non-quantized modes. Results showed that quantization reduced mem- ory usage by approximately 40% and improved through- put by

1.3×, while maintaining comparable generation quality. Non-quantized inference exhibited slightly lower latency for short sequences, but quantized execution pro- vided a more favorable trade-off for large-scale retrieval-augmented generation tasks. This substantiates the in- clusion of optional quantization as a design choice in the pipeline.

Evaluation of the RAG Pipeline

To assess the performance of the RAG pipeline, we defined a set of evaluation questions along with corre-sponding ground truth answers. The questions cover fundamental nutritional topics, including infant feeding, micronutrients, digestion, protein intake, and deficiency symptoms. The ground truth answers were obtained from authoritative sources to serve as benchmarks for evaluating the generated responses.

For each evaluation question, relevant context was re- trieved from the document corpus using semantic em-beddings generated with the all-mpnet-base-v2 model implemented in the SentenceTransformer library. Re-trieved context chunks were formatted into prompts, which were then passed to the locally hosted LLM to generate answers. The procedure can be summarized as follows:

Retrieve context chunks most relevant to the query using cosine similarity and dot product scores.

Format the query and context chunks into a prompt suitable for the LLM.

Generate the answer using the LLM, ensuring that only the generated response is extracted (prompt text is removed).

Record both the generated answer and the context chunks used for retrieval.

The evaluation of the RAG pipeline was conducted us- ing several metrics to measure the quality and reliability of generated answers:

Context Precision: Proportion of retrieved con- text that was relevant to the query.

Context Recall: Proportion of all relevant con- text that was successfully retrieved.

Answer Relevancy: Degree to which the gener- ated answer correctly addressed the question.

Faithfulness: Accuracy of the generated content with respect to the retrieved context.

Context Entity Recall (optional): Recall of named entities present in the retrieved context.

Noise Robustness (optional): Sensitivity of the system to irrelevant context or noisy input.

Each metric was computed for all evaluation questions, and average scores were used to categorize performance as Excellent (≥ 0.8), Good (0.6–0.79), Fair (0.4–0.59), or

Poor (<0.4).

The evaluation revealed the following key results:

Best Performing Metric: Faithfulness (1.000, Excellent)

Worst Performing Metric: Context Entity Re- call (0.280, Poor)

Overall Performance: Three metrics were Ex- cellent, one Good, zero Fair, and one Poor.

To address the inadequate Context Entity Recall, we plan to explore both reduced fixed chunk sizes (e.g., 5 sentences per chunk) and dynamic chunking strategies based on semantic boundaries. Preliminary experiments with smaller chunk sizes already indicate an improvement in entity coverage, suggesting that finer granularity can better capture domain-specific terminology. Addition- ally, we will conduct error analysis on evaluation items where entity recall failed, to identify whether missed enti- ties arise from chunking limitations, embedding sparsity, or preprocessing artifacts.

Analysis of these results provides actionable insights:

Retrieval was generally effective, as indicated by high context precision and recall scores.

Answer generation was reliable, achieving high rel- evancy and faithfulness.

The system struggled with retrieving all relevant entities, suggesting potential improvements in em- bedding quality or chunking strategy.

Detailed evaluation results, including per- question answers and contexts, were saved to rag_evaluation_results.csv for further inspection and reproducibility.

Limitations and Challenges

Several challenges were identified during implementa- tion:

Text Quality Variations: Some pages contained formatting artifacts and special characters that required additional cleaning. These issues occa- sionally resulted in fragmented or noisy text seg- ments, which may have reduced embedding fi- delity and contributed to the low Context Entity Recall scores. Improved preprocessing strategies such as advanced OCR normalization and entity- preserving cleaning will be evaluated in future iter- ations.

Embedding Model Constraints: The 384-token limit occasionally truncated longer, complex nutri- tional explanations.

Medical Terminology: Domain-specific terms required careful handling to ensure accurate seman- tic representation.

These limitations are addressed in the pipeline en- hancements, including experimenting with reduced and dynamic chunking strategies to improve entity coverage, refining preprocessing to handle formatting artifacts, and evaluating embedding and LLM configurations (quan- tized vs. non-quantized) to optimize both performance and fidelity, as discussed in the Conclusion.

CONCLUSION

The implemented RAG pipeline successfully demon- strates an end-to-end framework for processing and em- bedding nutritional textbook content. Key achievements include:

Robust Text Processing: The pipeline effec- tively handles real-world PDF documents with complex formatting and structure

Semantic Preservation: The chunking strategy maintains contextual coherence while optimizing for embedding model constraints

Scalable Architecture: The modular design sup- ports easy extension to larger document collections and different domains

Knowledge Accessibility: The vectorized repre- sentation enables efficient semantic search and re- trieval of nutritional information

The pipeline provides a solid foundation for developing intelligent nutritional assistance systems, with potential applications in:

Educational tools for nutrition students and pro- fessionals

Clinical decision support systems

Public health information retrieval platforms

Personalized dietary recommendation engines

In response to the evaluation results, particularly the inadequate Context Entity Recall score, we have out- lined several enhancements. These include experiment- ing with reduced chunk sizes and dynamic chunking approaches to improve entity coverage, as well as perform- ing detailed error analysis of missed entities. Furthermore, we provided a clear rationale for selecting the all-mpnet-base-v2 embedding model, acknowledging its strengths and limitations in handling domain-specific medical terminology. To strengthen computational efficiency, a comparison of quantized versus non-quantized Gemma-7B inference was conducted, demonstrating the practical benefits of quantization for large-scale tasks. Fi- nally, the influence of formatting artifacts on text quality was examined, highlighting their impact on preprocess- ing and embedding fidelity. Together, these refinements provide a roadmap for future iterations of the pipeline and enhance the robustness and interpretability of our methodology.

Future work should focus on enhancing the pipeline with more sophisticated chunking strategies, domain-specific embedding models, and integration with larger language models for improved question-answering capabilities.

DATA AVAILABILITY

The primary data source for this project is the Human Nutrition: 2020 Edition textbook, available under Creative Commons Attribution 4.0 International License from:

https://pressbooks.oer.hawaii.edu/ humannutrition2/

Processed Data Components:

Raw PDF document: human-nutrition-text.pdf

Extracted text chunks with metadata: JSON for- mat (1,843 entries)

Generated embeddings: NumPy array format (768- dimensional vectors)

Statistical summaries: CSV format with chunk characteristics

Data Characteristics

Total pages processed: 1,208

Final text chunks: 1,843

Vocabulary size: Approximately 15,000 unique terms

Domain coverage: Comprehensive nutritional sci- ence topics

All processed data and embeddings are available in the project repository for research and educational purposes.

Code Availability

The complete implementation of the RAG pipeline is available as an open-source project under MIT License.

Repository: https://github.com/ Olanrewajuemmanuelabiodun/RAG_Nutrition/tree/ main

The codebase is designed for easy modification and ex- tension, with comprehensive documentation and example usage patterns.

1. K. I. Roumeliotis, N. D. Tselikas, and D. K. Nasiopou- los, "Llms in e-commerce: a comparative analysis of gpt and llama models in product review evaluation," Natural Language Processing Journal, vol. 100056, pp. 1–6, 2024.
2. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sas- try, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in Advances in Neural In- formation Processing Systems (NeurIPS 2020), vol. 33, 2020, pp. 1877–1901. OpenAI, "Gpt-4 technical report," 2023.
3. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, and Wang, "Retrieval-augmented generation for large lan- guage models: A survey," 2023.
4. N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long- tail knowledge," in Proceedings of the International Con- ference on Machine Learning (ICML). PMLR, 2023, pp. 15 696–15 707.
5. F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019, pp. 2463–2473. [Online]. Available: https://www.aclweb.org/anthology/D19-1250
6. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019. [Online]. Available: https://arxiv.org/abs/1910.10683
7. A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" 2020. [Online]. Available: https://arxiv.org/abs/2002.08910
8. G. Marcus, "The next decade in ai: Four steps towards robust artificial intelligence," 2020. [Online]. Available: https://arxiv.org/abs/2002.06177
9. K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," 2020. [Online]. Available: https://arxiv.org/abs/2002.08909
10. V. Karpukhin, B. Oguz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," 2020. [Online]. Available: https://arxiv.org/abs/2004.04906
11. F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel, "How context affects language models' factual predictions," in Automated Knowledge Base Construction (AKBC), 2020. [Online]. Available: https://openreview.net/forum?id=025X0zPfn
12. H. Zamani, F. Diaz, M. Dehghani, D. Metzler, and M. Bendersky, "Retrieval-enhanced machine learning," in Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22). Madrid, Spain: Association for Computing Machinery, 2022, pp. 2875–2886. [Online]. Available: https://doi.org/10.1145/3477495.3531722
13. G. Agrawal, T. Kumarage, Z. Alghami, and H. Liu, "Can knowledge graphs reduce hallucinations in llms?:A survey," 2023. [Online]. Available: https: //arxiv.org/abs/2311.07914
14. G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, P. Merlo, J. Tiedemann, and
15. R. Tsarfaty, Eds. Online: Association for Computational Linguistics, 2021, pp. 874–880. [Online]. Available: https://doi.org/10.18653/v1/2021.eacl-main.74
16. A. Salemi, S. Kallumadi, and H. Zamani, "Optimization methods for personalizing large language models through retrieval augmentation," in Proceedings of the 47th An- nual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), Washington, DC, USA, 2024, (to appear).
17. A. Salemi, S. Mysore, M. Bendersky, and H. Zamani, "Lamp: When large language mod- els meet personalization," 2023. [Online]. Available: https://arxiv.org/abs/2304.11406

18. J. James and S. Es, “Ragas: Evaluation framework for your retrieval-augmented genera- tion (rag) pipelines,” 2023. [Online]. Available: https://github.com/explodinggradients/ragas
19. J. Saad-Falcon, O. Khattab, C. Potts, and M. Za- haria, “Ares: An automated evaluation framework for retrieval-augmented generation systems,” 2023. [Online]. Available: https://arxiv.org/abs/2311.09476
20. F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, V. Plachouras, T. Rocktäschel, and S. Riedel, “Kilt: a benchmark for knowledge in- tensive language tasks,” in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan- guage Technologies (NAACL-HLT 2021), K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Belt- agy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, 2021, pp. 2523–2544. [Online]. Available: https://doi.org/10.18653/v1/2021.naacl-main.200
21. A. S. . P. authors, “Pymupdf tutorial — docu- mentation,” https://pymupdf.readthedocs.io/en/latest/tutorial.html, 2025, accessed: 2025-09-25.
22. M. Fenniak and pypdf Contributors, “pypdf documen- tation,” https://pypdf.readthedocs.io/en/stable/, 2025,
23. accessed: 2025-09-25.
24. The pandas development team, “pandas documentation,” https://pandas.pydata.org, 2025, accessed: 2025-09-25.
25. H. Face, “Datasets documentation,” https://huggingface. co/docs/datasets/en/index, 2025, accessed: 2025-09-25.
26. E. AI, “Models & languages — spacy usage documen- tation,” https://spacy.io/usage/models, 2025, accessed: 2025-09-25.
27. N. Reimers and I. Gurevych, “all-mpnet-base- v2,” https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2021, accessed: 2025-09-25.
28. H. Face, “Transformers documentation,” https:// huggingface.co/docs/transformers/en/index, 2025, ac- cessed: 2025-09-25.
29. E. Gradients, “Ragas documentation,” https://docs. ragas.io/en/stable/, 2025, accessed: 2025-09-25.