

AN OFFLINE RAG SYSTEM PROTOTYPE FOR SAFE QUERYING OF UNSTRUCTURED DOCUMENTS

David Vlastník¹, Jaromír Landa¹

¹Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 602 00 Brno, Czech Republic

DOI: https://doi.org/10.11118/ecos2026_040

ABSTRACT

In the context of large language models (LLMs) being utilized within organizations that manage sensitive data, two notable limitations of commercial cloud solutions become evident. Firstly, there is the risk of confidential information being disclosed when content is transmitted to external servers. Secondly, there is the issue of the models' propensity to hallucinate in the absence of reliable context. The present paper proposes and implements a prototype of a fully offline RAG (Retrieval-Augmented Generation) system using exclusively open-source components. The architecture is predicated on the Agentic RAG approach, which substitutes linear query processing with a multi-agent system orchestrated using the LangGraph framework. A hybrid (dense + sparse) approach over the Qdrant vector database is used for search, and the parallel relational database DuckDB is used for tabular data. The system is containerized using Docker Compose and runs a local LLM via the Ollama platform. The evaluation of the output quality was conducted through a comparative analysis with three existing tools (LM Studio, AnythingLLM, NotebookLM) on a set of ten extensive PDF reports on climate change. The evaluation process involved the utilization of five complex questions, necessitating the synthesis of information from multiple sources. The implemented prototype achieved a level of accuracy comparable to the cloud-based NotebookLM tool and significantly outperformed solutions employing local LLMs. It also maintained transparency of answers through citations of source documents and ensured that data remained on the user's local device.

Keywords: retrieval-augmented generation, large language models, agentic RAG, vector databases, hybrid search, open-source LLM, data protection, semantic search, on-premises deployment

JEL Code: L86, O33



This article is licensed under CC BY-SA 4.0 Attribution-ShareAlike 4.0 International license. <https://creativecommons.org/licenses/by-sa/4.0/deed.en>

1 INTRODUCTION

Recent years have seen a rapid development of generative artificial intelligence and large language models (LLMs), which are gradually transforming the way users and organizations work with large-scale text data. The foundational technological principle underpinning these models is the Transformer architecture (Vaswani, 2023), which has facilitated a substantial enhancement in the quality of language understanding and generation. Commercial tools such as NotebookLM¹ now commonly allow users to upload documents and ask complex questions about their content. However, it should be noted that this functionality is operated as a cloud service, which necessitates the transmission of document content to external servers. For organizations handling confidential information, personal data, or corporate know-how, such a processing regime poses a serious security risk, particularly in sectors subject to regulatory obligations regarding data sovereignty.

A second limitation of LLMs that hinders their deployment in processes requiring accuracy is their tendency toward so-called hallucinations, whereby convincing formulations are produced, but lack substantiation or are false. The prevalence of hallucinations has been documented in the extant literature as a systemic problem inherent to all current generative models (Zhang *et al.*, 2025; Huang *et al.*, 2025). The likelihood of hallucinations increases in circumstances where the model lacks access to the current domain context and is compelled to rely exclusively on parameters acquired during the training process.

A potential solution to the aforementioned risks is the deployment of open-source language models in conjunction with a Retrieval-Augmented Generation (RAG) architecture, executed at a local level. RAG first searches the provided documents for text segments relevant to the user's query and only then passes them to the language model as a basis for the response (Chai, 2024). The dual benefit of this approach is twofold: firstly, it narrows the scope for the model to fabricate information (the response must be grounded in the found context); and secondly, it increases the verifiability of the output by allowing for a reference to a specific source document. Concurrently, the local deployment of components guarantees that the content of the documents and the user queries remain within the organization's isolated environment.

The practical implementation of this approach is encumbered by several interrelated limitations. It is evident that locally executable models generally possess a more limited context window and a comparatively diminished capacity to synthesize information from multiple sources in comparison to the most advanced commercial models. Furthermore, a naive implementation of RAG, where the system performs a single semantic search and directly passes the obtained context to the model, fails for more complex queries that require combining information from a larger number of text fragments or switching between semantic search and aggregation operations on structured data (Jeong *et al.*, 2024). In order to overcome these limitations, more advanced variants of RAG are emerging. One such example is Agentic RAG, which shifts from a linear pipeline to the orchestration of specialised agent nodes (Xu *et al.*, 2025).

The goal of this thesis is to design and implement a prototype RAG system that processes unstructured text data in various formats (PDF, DOCX, PPTX, CSV, XLSX), allows users to:

- ask complex queries across the entire set of documents simultaneously,
- runs all data processing components (text extraction, embedding models, and LLM) exclusively locally without relying on cloud services,
- provides citations of source documents for each answer, thereby increasing the verifiability of the output.

¹<https://notebooklm.google.com/>

The implemented prototype is then subjected to a comparative evaluation against three existing tools—LM Studio², AnythingLLM³, and Google’s cloud-based NotebookLM⁴ using the same set of large-scale documents. The following section describes the proposed architecture, the technologies used, and the evaluation procedure. The Results section presents the comparison results and key observations. The Discussion and Conclusions section interprets the findings, analyzes architectural trade-offs, and summarizes recommendations for practical deployment.

2 METHODOLOGY AND DATA

The system is divided into two pipelines. The ETL pipeline is responsible for processing documents, with text extraction performed using the Docling library, which has been demonstrated in an independent comparison (Javanmard, 2025) to be the most accurate open-source tool for converting PDF, DOCX, and PPTX documents. Semantic segmentation is achieved through the use of the SpaCy library with the `en_core_web_sm` model, which divides the text based on cosine similarity between adjacent sentences. Parallel storage is facilitated in the Qdrant vector database and the DuckDB relational database. Tabular data (CSV, XLSX) is stored in both databases, as semantic search is unable to efficiently handle aggregation queries on structured data.

The proposed system is composed of several parts:

- The **Router** is responsible for classifying the intent of the query, which is determined by the following categories: GENERAL – casual conversation; RAG – factual query; RAG_EXHAUSTIVE – enumeration request; RAG_SUMMARIZATION – summarization.
- The **Query Planner** determines the search strategy, which may be a semantic search over a vector database, an SQL query over a relational database, a hybrid combination of both, or the loading of all fragments of a specific source for summarization queries.
- **Analytical Query** is responsible for generating and executing SQL queries on a relational database (with the exception of modifying INSERT and DELETE statements) and formatting the retrieved rows into text. In the event of failure, the node is restarted, with an error message being appended to the prompt. In the case of the SQL strategy, the process advances directly to the Synthesizer node; in contrast, the HYBRID strategy passes control to the Research Worker node to obtain further context from the vector database.
- The **Research Worker** operating in parallel for each query generated by the Query Planner, performs a hybrid search using both dense and sparse embeddings. Documents that have been found in previous iterations are automatically excluded in order to prevent the creation of duplicates.
- **Scroll Retriever** circumvents the utilization of semantic search, instead retrieving all fragments from the source specified. These fragments are then reordered by the FlashRank model based on their relevance to the original query, and subsequently directed to the Fact Extractor node.
- **Retrieval Grader** employs the FlashRank model to deduplicate and reorder fragments, with the number of retained fragments being dynamically derived from the size of the context window utilized by the LLM.

²<https://lmstudio.ai>

³<https://anythingllm.com/>

⁴<https://notebooklm.google.com/>

- The **Fact Extractor** then extracts useful information from the fragments and applies a cosine similarity check (threshold value 0.15) between the extracted information and the original text to prevent hallucination at this stage.
- The **Synthesizer** is responsible for generating the final answer, and it is required to provide a list of the names of the source files for each statement.
- The **Completeness Checker** and **Hallucination Grader** form self-correcting mechanisms: the former initiates supplementary searches, if necessary (up to three iterations), while the latter verifies that the response is based exclusively on the provided context; if not, it triggers a regeneration of the response (up to two correction attempts, after which a warning about the potential unverified nature of the information is appended to the response).

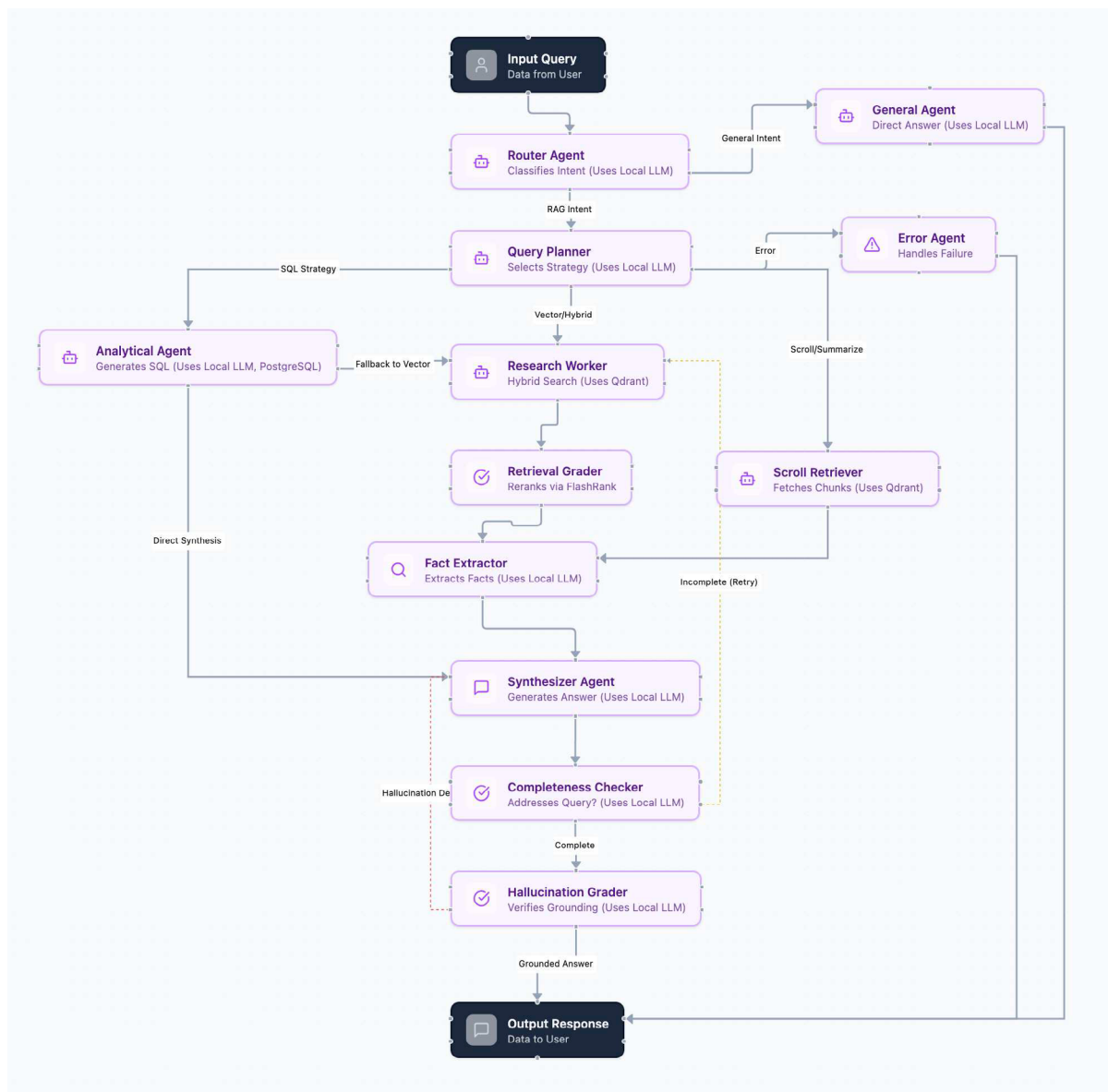


Fig. 1: A detailed visualization of the multi-agent system used within RAG. (Vlastník, 2026)

The Ollama platform is utilized for the operation of local LLMs, ensuring seamless integration with Python and providing support for a comprehensive range of open-source models. The system is entirely containerized using the Docker Compose tool, a practice that facilitates its deployment across diverse operating systems without the necessity of installing dependencies.

For the purpose of comparative analysis, a set of ten extensive PDF documents containing reports on climate change and meteorological phenomena was selected. It should be noted that the size of some of these files exceeded 100 MB. A comparative analysis was conducted on four tools:

- **LM Studio** is a local tool with an integrated RAG plugin.
- **AnythingLLM** is an open-source platform that utilizes LanceDB and the Ollama or LM Studio platforms as a backend for LLM.
- **NotebookLM** is a cloud solution from Google that serves as a reference benchmark.
- **The implemented prototype.**

For tools supporting the local model, minstral-3:14b was utilized in all cases. The evaluation was conducted using five complex questions, the answers to which required the synthesis of information from multiple datasets simultaneously (e.g., the trend in Antarctic Sea ice minimum extent between 2022 and 2024 across three independent reports).

LLM-as-a-judge script was implemented for the automated evaluation of responses. The evaluation model receives the user's query, the generated response, the extracted context, and the reference answer. It then assesses the output on a 1–5 scale across two dimensions: relevance (whether the response addresses the query and aligns with the reference answer) and fidelity (whether the response is based exclusively on the extracted context). The evaluation is deemed successful if both dimensions attain a score of at least 4. To mitigate evaluation indeterminism, the evaluation model's temperature is set to zero (PatronusAI, 2026).

3 RESULTS

Table 1 summarizes the overall results of the comparative evaluation. Detailed observations regarding each tool are discussed below.

Tab. 1 A Comparison of RAG-Supporting Tools Using a Set of Five Complex Questions Across Ten PDF Reports on Climate (Vlastník, 2026)

	LM Studio	AnythingLLM	NotebookLM	Prototype
Document processing	Limited (limit of 31.48MB)	Complete	Complete	Complete
Question 1	Completely incorrect	Incomplete	Complete	Full
Question 2	Completely incorrect	Completely incorrect	Complete	Correct
Question 3	Completely incorrect	Incomplete	Complete	Full
Question 4	Completely incorrect	Incomplete	Complete	Full
Question 5	Completely incorrect	Incomplete	Complete	Full
Provide citations	Yes	No	Yes	Yes
Final evaluation	Unsatisfactory	Partially satisfactory	Satisfactory	Satisfactory

LM Studio

The tool encountered an error during the dataset preparation phase, resulting from a hard limit of 31.48 MB per file. The language model was unable to access the relevant information required to provide correct responses to the test questions, with the result that it generated mostly hallucinations. This finding indicates that integrated RAG functionalities in tools primarily designed for local chat with LLMs may be ineffective for working with large documents, even if this is not immediately apparent.

AnythingLLM

The process of document uploading was accomplished without encountering any issues and required a time period of approximately minutes. This can be attributed to the utilization of fixed text segmentation, a method that is less computationally demanding in comparison to semantic segmentation (Qu *et al.*, 2025). It was evident that while some responses were factually correct, others contained what could be termed ‘confident hallucinations’, i.e. information that was not found in any of the provided documents. A significant systemic deficiency is the absence of citations in the generated responses, which hinders the distinction between unsuccessful searches and failed syntheses of the responses, thereby complicating system tuning in a production environment.

NotebookLM

The cloud-based tool processed all documents with great rapidity and provided correct answers to the five test questions. Consequently, it functioned as a reference benchmark for the attainment of quality. From the perspective of the original thesis assignment, the fundamental drawback remains the necessity of sending documents to a cloud environment, thereby eliminating the protection of sensitive data — precisely the limitation that motivated this work.

The implemented prototype

A direct comparison of the two solutions reveals that the proposed solution performs comparably to the NotebookLM tool. The system was capable of searching for relevant information across multiple documents, linking them together, and generating responses that included citations of the source files. In a single subcase, the language model demonstrated a prioritization of detailed meteorological data over summary humanitarian and economic statistics, resulting in a partially incomplete response. The most significant limitation of the prototype is document processing time: the complete processing of the test set took approximately two hours, which is a consequence of using semantic text segmentation. While this strategy does extend the ETL phase, preliminary observations suggest that it improves the quality of subsequent responses by preserving the semantic coherence of stored fragments (Qu *et al.*, 2025).

The impact of hardware configuration is a subject that merits further investigation. The development and initial testing were conducted on a MacBook M3 Pro with 18 GB of shared memory. On this particular platform, the system attained a success rate of 80–90% by employing local models with 8B parameters. An analysis of unsuccessful tests indicated that the search generally yielded relevant information; however, the model was incapable of synthesizing it correctly. Subsequent repetition of the evaluation on a device equipped with a dedicated Nvidia GeForce RTX 5090 graphics card and larger VRAM capacity yielded identical models achieving success rates of 95–100%. The probable cause of the issue appears to be disparate memory management. On the MacBook, shared memory was required to support the operating system, the database

and the agent system. In certain instances, this could result in the clipping of the context window when a substantial number of fragments were supplied. This phenomenon was compensated for in the implementation by dynamically configuring the number of fragments and the context size according to the currently used model.

The following conclusions can be drawn from the comparison. Firstly, it is imperative to acknowledge the pivotal role that document preprocessing plays in ensuring the efficacy of the entire system. The failure of the LM Studio tool when confronted with file size limitations serves to underscore the limitations inherent in the ETL phase, thereby negating the potential benefits of even a sophisticated LLM. Secondly, hallucinations were identified as the predominant cause of failure in the LM Studio and AnythingLLM tools. In the implemented prototype, their occurrence was significantly reduced by combining more precise hybrid search, extraction of relevant facts, and self-correcting mechanisms to verify the completeness and accuracy of the response. Thirdly, cross-document synthesis necessitates a high-quality search. In the event of retrieval failure, a more powerful language model is unable to compensate for the information deficit. Fourthly, it is imperative to incorporate citations in order to ensure the verifiability and calibration of the system. In conclusion, a local solution that is built on open components has demonstrated competitiveness against the cloud-based NotebookLM, despite utilizing a considerably reduced language model and offering a mere command-line interface.

4 DISCUSSION AND CONCLUSIONS

The transition from the Naive RAG to the Agentic RAG approach (Xu *et al.*, 2025) was identified as a pivotal architectural decision. In the context of more complex queries, the linear sequence of context retrieval and response generation reaches a limit imposed by the model's context window and the one-time nature of the search. The replacement of the pipeline with a multi-agent system featuring a planning node (Query Planner), parallel search (Research Worker), filtering (Retrieval Grader), and iterative supplementary search (Completeness Checker) resulted, according to observations made during the development process, in the most significant enhancement to the system's overall success rate in a single step. These findings are consistent with the broader trend of RAG architectures shifting from a static to an agent-based paradigm, as described in the literature (Jeong *et al.*, 2024).

A secondary, yet substantial, decision was the integration of semantic and lexical search components. The utilization of metadata extracted by a language model as a pre-filter, a process which was subjected to initial testing, paradoxically resulted in outcomes that were inferior to those achieved by a pure hybrid search combining dense and sparse embeddings, a combination that is natively supported by the Qdrant database. The probable cause of this issue is the lack of reliability of smaller local models in the process of keyword extraction, in addition to errors in the specialized agent that had to link these words between the query and the metadata in the database.

The processing of structured data necessitated the integration of a parallel relational database. Semantic search has not been designed for aggregate queries on tables of the type “select the ten highest values”. Following an unsuccessful attempt to deploy PostgreSQL, whose syntax and schemas smaller local models could not consistently generate, the built-in analytical database DuckDB was selected. This proved effective even with less powerful models when interacting with the SQL agent.

The evaluation yielded several management-relevant implications for organizations considering the deployment of RAG on sensitive documents. Firstly, it is evident that local deployment with open LLMs is currently a viable alternative to cloud solutions, at least for organizations that can tolerate longer document processing times and the absence of a graphical user interface. Secondly, the accuracy of responses is contingent on computational power, particularly graphics memory capacity. The disparity in memory capacity between 18 GB of shared memory on a laptop and the dedicated VRAM of a high-performance graphics card resulted in a notable enhancement in success rates, from 80–90% to 95–100%, for models of identical design. Consequently, investment in hardware constitutes an indispensable component within the cost calculation for such a solution. Thirdly, the choice between RAG and model fine-tuning remains open. RAG is suitable where the document corpus grows dynamically and a cited response is required. Conversely, fine-tuning is advantageous when consistent style and deep domain expertise are needed (Ailaitte, 2026).

The selected architectural design is also subject to inherent limitations. Reasoning models, which engage in an internal chain of reasoning prior to outputting a result, have been demonstrated to be ineffective within Agent RAG. The high latency of these models, which is exacerbated by the number of agent nodes, results in a disproportionate overhead for simple decision-making tasks (such as intent classification in Router). In the context of multi-agent systems, it is therefore more appropriate to employ standard models and, if necessary, reserve the reasoning model solely for the final synthesis of the response. The limitations of the system are evident in a number of ways. Firstly, the test set is limited in size, comprising a mere five questions across a single domain, namely climate reports. Secondly, the absence of formal statistical evaluation is notable. Thirdly, the restriction to a command-line interface hinders usability for non-technical users.

The present paper has demonstrated that a RAG system prototype built exclusively on open-source components and operated entirely offline can achieve a quality comparable to that of a leading cloud-based competitor when performing factual queries on a large collection of unstructured documents. The key building blocks that were identified in this study are: firstly, the Agentic RAG architecture; secondly, hybrid search over a vector database; thirdly, a parallel relational database for tabular data; and finally, a multi-stage self-correcting mechanism (which includes a cosine similarity check during fact extraction, response completeness verification, and hallucination check). The primary constraints pertain to the protracted document processing time, attributable to semantic segmentation, the reliance on local hardware performance, and a user interface constrained to the command line. Consequently, the subsequent enhancement of the system is centered on the optimization of the ETL pipeline and the development of a desktop application that would facilitate the utilization of the solution by non-technical users. From the perspective of broader implications for organizational competitiveness, the prototype demonstrates that protecting corporate information assets while utilizing generative AI does not necessarily entail a significant compromise in output quality — provided the organization is willing to invest in appropriate local hardware and accept the current functional limitations of open models.

Acknowledgements

This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Research Infrastructure for Young Scientists, this is co-financed from Operational Programme Research, Development and Education.

Several AI tools were used in the preparation of this article. They assisted with summarizing source material, translating passages between languages, and identifying typographical and stylistic issues in the text. All output produced by these tools was reviewed and verified. Summaries were checked against the original sources, translations were assessed for accuracy, and suggested

corrections were evaluated before being accepted. The tools served as an aid to the writing process; the final content reflects the authors' judgment and remains the authors' responsibility.

REFERENCES

- AILOITTE. 2026. RAG vs fine-tuning: which AI approach should businesses choose? *Ailoitte*. <https://www.ailoitte.com/insights/rag-vs-fine-tuning>.
- CHAI. 2024. RAG 101: Ep 1 — retrieval-augmented generation (RAG), why? *Medium*. [Accessed: 2026-03-07] <https://medium.com/@thechaixperiment/rag-101-ep-1-retrieval-augmented-generation-rag-basics-aaf7cafdd0ce>
- HUANG, L., YU, W., MA, W., ZHONG, W., FENG, Z., WANG, H., CHEN, Q., PENG, W., FENG, X., QIN, B., LIU, T. 2025. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*. 43(2), 1–55. <https://doi.org/10.1145/3703155>
- JAVANMARD, A. 2025. PDF data extraction benchmark 2025: comparing Docling, Unstructured, and LlamaParse for document processing pipelines. *Procyons*. [Accessed: 2026-01-24] <https://procyons.com/en/blogs/pdf-data-extraction-benchmark/>
- JEONG, S. *et al.* 2024. Adaptive-RAG: learning to adapt retrieval-augmented large language models through question complexity. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Volume 1: Long Papers. Mexico City: Association for Computational Linguistics, pp. 7036–7050.
- PATRONUSAI. 2026. RAG evaluation metrics: best practices for evaluating RAG systems. *Patronus AI*. [Accessed: 2026-03-10] <https://www.patronus.ai/llm-testing/rag-evaluation-metrics>
- QU, R., TU, R., BAO, F. S. 2025. Is semantic chunking worth the computational cost? In: *Findings of the Association for Computational Linguistics: NAACL 2025*. Albuquerque: Association for Computational Linguistics, pp. 2155–2177. <https://doi.org/10.18653/v1/2025.findings-naacl.114>
- VASWANI, A. 2023. Attention is all you need. *arXiv:1706.03762v7* <https://arxiv.org/html/1706.03762v7>
- VLASTNÍK, D. 2026. *Prototyp RAG (Retrieval-augmented generation) pro analýzu velkého množství nestrukturovaných textů*. Master's thesis. Brno: Mendel University in Brno. Faculty of Business and Economics, Department of Informatics.
- XU, X. *et al.* 2025. Agentic RAG with human-in-the-retrieval. In: *2025 IEEE 22nd International Conference on Software Architecture Companion (ICSA-C)*. Odense, Denmark. IEEE. <https://doi.org/10.1109/ICSA-C65153.2025.00074>.
- ZHANG, Y. *et al.* 2025. Siren's song in the AI ocean: a survey on hallucination in large language models. *Computational Linguistics*. 51(4), 1373–1418. <https://doi.org/10.1162/COLI.a.16>

Contact information

David Vlastník: e-mail: xvlastni@node.mendelu.cz

Jaromír Landa: e-mail: jaromir.landa@mendelu.cz