Hybrid Retrieval for Retrieval Augmented Generation in the German Language Production Domain

Simon Knollmeyer [1,*], Sebastian Pfaff [2], Muhammad Uzair Akmal [1], Leonid Koval [1], Saara Asif [1], Selvine G. Mathias [1], and Daniel Großmann [1]

[1] AImotion Bavaria, Technische Hochschule Ingolstadt, Ingolstadt, Germany
[2] Information and Technology, TUM School of Computation, Technische Universität München, Garching bei München, Germany
Email: Simon.Knollmeyer@thi.de (S.K.); S.Pfaff@tum.de (S.P.); MuhammadUzair.Akmal (M.U.A.); Leonid.Koval@thi.de (L.K); Saara.Asif@thi.de (S.A.); SelvineGeorge.Mathias@thi.de (S.G.M.); Daniel.Grossmann@thi.de (D.G.)
*Corresponding author

***Abstract*—Retrieval Augmented Generation (RAG) is an emerging method for leveraging Artificial Intelligence in the field of knowledge management, particularly within specialized domains. In this study, we focus on evaluating the effect of hybrid retrieval techniques on German technical documents, which are widely used in the production and engineering departments of German companies. RAG employs a Large Language Model (LLM) that accesses an extensive information store to answer user queries by retrieving the most relevant text passages, known as chunks, from a database. The efficiency of this retrieval process is crucial for the overall performance of the RAG system. Classical RAG employs dense vector embedding and nearest neighbor search for information retrieval. State of the art of shelf embedding models tend to struggle with non-English and highly domain-specific texts. Given the language, complexity, and specificity of technical production planning documents, we propose a hybrid retrieval approach combining full-text and common RAG vector search. We constructed 2,000 question-and-answer pairs for each language, German and English, in a representative corpus of identical texts. Our proposed hybrid approach consistently enhances the retrieval performance for German documents by 20% over a purely vector-based search, entirely erasing the deficiencies of embedding models for German texts, thus demonstrating its significant potential to improve knowledge management in technical and industrial contexts.**

***Keywords*—hybrid retrieval, retrieval augmented generation, production domain, German documents**

I. Introduction

Effective knowledge management is critical for achieving efficient task completion in factory planning. The growing volume of accessible data, driven by the digitalization of business processes and the integration of Industry 4.0 in manufacturing environments, intensifies this challenge [1].

A promising method for facilitating AI-assisted document question-answering in factory planning involves Large Language Model (LLM)-powered chatbots. In a Retrieval Augmented Generation (RAG) approach, the language model interacts textually with a knowledge base comprising textual document content. Relevant sections are retrieved from the knowledge base based on the user's query context [2]. The indexing and retrieval of relevant text sections are pivotal to the RAG system's performance, as they determine the information utilized by the LLM for response generation. Barnett et al. identify this step as a critical failure point in RAG system development [3].

According to Gao *et al*. [4], typical RAG systems employ a text embedding model with a vector database for indexing and retrieval. Document texts are divided into smaller units, or chunks, which are embedded into a high-dimensional vector space based on semantic distinctions. Chunks most relevant to a query are retrieved and passed to an LLM for question answering. However, dense vector embeddings often struggle to capture the precise semantics of domain-specific texts. Moreover, specialized vocabulary in such texts makes full-text searches particularly valuable, especially for non-English documents, where multilingual vector embedding models may have limited exposure to specific languages during training.

These challenges are particularly relevant to the knowledge base of the production planning department at a German automobile manufacturer, where most information is highly technical and in German. This work explores the feasibility and benefits of enhancing indexing and retrieval procedures by integrating traditional full-text search into a hybrid approach within this context.

Our contributions to the existing literature include:

- Demonstrating that multilingual vector embeddings retrieve domain-specific information

Manuscript received January 20, 2025; revised February 18, 2025; accepted March 14, 2025; published June 12, 2025.

more effectively for English documents than for German ones.
- Proposing a scalable hybrid retrieval approach that improves accuracy for both German and English documents, addressing the limitations of vector embeddings for German texts and achieving comparable accuracy across both languages.

The paper is structured as follows: Section II reviews related work on hybrid retrieval and chunking for RAG. Section III presents the proposed methodology. Section IV details the implementation and evaluation of the concept using two datasets and analyzes the results. Section V discusses findings and limitations, and the paper concludes with a summary and directions for future research.

II. Related Work

Gao *et al*. [4] highlight that effective retrieval is pivotal for a successful RAG pipeline. Numerous strategies have been proposed to improve retriever performance, including recursive retrieval [5], re-ranking [6], chunk optimization [3, 7], and hybrid retrieval (multi-path recall) [8, 9]. Additional advancements include retrieval transformation [10] and strategies that do not fit into a single category [11]. For an in-depth survey on RAG and retrieval strategies, see [12, 13].

The choice of retrieval strategy depends largely on the language and content of the underlying documents [14]. This work adopts Hybrid Retrieval to suit the technical German-language document corpus. To retrieve relevant text chunks, embedding models encode them into vectors. These embeddings, capturing text semantics, enable similarity searches based on content relevance. The two primary encoding methods, dense and sparse, will be discussed in the following sections.

A. Vector Search (Dense Retrieval)

Vector (dense) embeddings transform words, phrases, or texts into dense vectors in high-dimensional space, where proximity indicates semantic similarity. These embeddings are trained using methods such as supervised learning with labeled data, self-supervised learning with contrastive loss [15], or unsupervised learning with algorithms like Word2Vec [16] or Bidirectional Encoder Representations from Transformers (BERT) [17]. The models map semantically similar texts closer together in the embedding space. The resulting vectors, typically a few hundred or thousand dimensions, are dense with few or no zero entries. Vector search identifies vectors like the query's embedding, commonly measured using cosine distance or the L^2 norm.

Dense embeddings and vector search provide significant advantages. They capture semantic meaning, reducing sensitivity to irrelevant words or synonyms, and scale well, allowing new chunks to be added without re-indexing, unlike full-text search methods such as Best Match 25 for multiple Fields (BM25F) [18]. For many retrieval tasks, dense embeddings outperform sparse searches [19, 20].

However, dense embeddings also have limitations. Embedding models can struggle with domain-specific terminology not well-covered in training data, where precision is crucial [21]. Multilingual embedding models often underperform on non-English texts, including German [22]. This work systematically investigates retrieval disparities between German and English texts using dense embeddings.

Dense retrieval can be computationally intensive for large datasets, as nearest neighbor searches are resource-heavy, though recent algorithms aim to improve efficiency [23]. Fine-tuning retrievers helps adapt dense embeddings to specific tasks, such as technical German documents, but this approach requires substantial GPU resources, large data volumes, and introduces task specificity, limiting flexibility for other applications [24, 25].

To address these challenges, this work explores combining dense vectors with sparse retrieval strategies, particularly full-text searches, to enhance performance while mitigating limitations.

B. Full-text Search (Sparse Retrieval)

Full-text search is a technique used in databases and information retrieval systems to locate documents containing specific text or keywords. It involves indexing content, tokenizing it into individual words, and enabling complex queries with Boolean operators, phrases, and wildcards. Documents are represented as high-dimensional, sparse vectors predominantly filled with zeros, as most words do not appear in each document. This work focuses on deterministic algorithms such as one-hot encoding [26], Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) [27], and Okapi BM25 [18], collectively referred to as "sparse retrieval" or "full-text search". For more details, see Ref. [28].

Full-text search techniques often provide greater efficiency compared to vector-based searches [29] and in some cases, they even outperform them [30]. These methods focus on identifying exact or nearly exact matches of terms, which makes them particularly effective when there is substantial overlap between the search query and the documents. This is especially beneficial for specialized content, such as technical papers or domain-specific texts that contain specialized terminology, acronyms, or industry-specific terms [31]. Full-text search is particularly useful when it's crucial to retrieve documents with precise terms or phrases, something that semantic vector-based approaches might not handle as well.

However, full-text search does have its drawbacks. Its reliance on exact keyword matching can overlook broader meanings or contexts, resulting in irrelevant results when users don't use precise terms. It struggles with recognizing synonyms or related expressions, and as a result, may miss documents that convey similar ideas but use different wording. Moreover, full-text searches can yield low precision, returning a large number of irrelevant documents that happen to contain the search terms. Additionally, variations in language, such as tense, grammatical structure, or word order, can complicate the effectiveness of this method.

Given the advantages and limitations of both vector and full-text search methods, along with the challenges of retriever fine-tuning, we suggest a hybrid retrieval system that combines the strengths of both approaches.

C. Hybrid Retrieval

Hybrid retrieval refers to the combined use of various retrieval methods or information sources. In our study, it integrates vector and full-text searches.

This approach has gained traction among practitioners, particularly for handling complex queries or code snippets where exact matches are essential [32, 33]. Hybrid retrieval balances semantic search for contextual understanding with exact phrase matching for specific terms, such as technical products, making it ideal for our dataset. For instance, *Stack Overflow* initially relied on TF-IDF for keyword matching, while now employing semantic search for context-aware queries alongside exact matching for tasks like error messages or specific methods [34, 35].

Academic discussions on hybrid retrieval cover various techniques, particularly in code retrieval [8]. Most hybrid methods employ similarity (rank) fusion, where the retrieved rank or score of selected chunks is merged using a weighting function, often referred to as α tuning [36]. This involves a linear combination of similarity scores or ranks from different search methods. An early survey on rank fusion predating RAG [2] is provided in [37], while examples of combined similarity scores and ranks in RAG models can be found in [38, 39].

Although simple α-based combinations are widely used, they are sensitive to the selected α value, complicating fine-tuning and increasing overfitting risks. If poorly calibrated, one model may dominate, creating imbalances. Optimal α values can vary by query, leading to inconsistent results, while misaligned score normalization may distort outcomes further. Although more complex weighting schemes could address these issues, they introduce higher computational costs and instability.

Our approach avoids relying solely on weighted score combinations. Instead, we fix the number of unique retrieved results per model to mitigate dominance from score imbalances. This ensures equal contribution from both models, promoting balanced outcomes and reducing dependence on score normalization. By focusing on diverse representation within the final result set, our method enhances adaptability across varied queries while maintaining stable performance.

D. Remaining Challenges

Most improvements to RAG systems have focused on English texts, reflecting English's global role as the primary language of science and business. Dense vector embeddings often perform better for English documents, reducing the perceived need for advanced hybrid retrieval methods in many domains.

Hybrid retrieval strategies have also primarily targeted code retrieval, where full-text search excels due to the rule-based syntax of programming languages. Code adheres to standardized structures, minimizing the necessity for complex techniques like α-tuning. Additionally, the wide availability of code online simplifies retrieval tasks further.

In contrast, research involving German automotive manufacturers faces unique challenges. Relevant information is more diverse, often proprietary, and generally unavailable for academic use. To address these gaps, our work makes the following contributions: (1) introducing a novel bilingual document corpus in German and English, (2) emphasizing the need for tailored retrieval strategies for domain-specific German texts, and (3) proposing a simple, scalable hybrid approach that significantly improves retrieval accuracy, overcoming the limitations of dense embedding models for German-language documents.

III. Methodology

In this section, we outline our methodology, starting with a description of the dataset creation process in subsections A and B. This is followed by an explanation of the comparison pipeline in subsection C. In subsection D, we present our hybrid approach. The section concludes with an overview of the evaluation method and the models employed.

A. Dataset

In production planning, much of the available information is stored in lengthy, text-heavy PDFs, such as product requirement documents and scope statements. These documents, often highly technical and primarily in German, can span hundreds of pages and are typically inaccessible to the public, limiting transparency. To address this challenge, we sought a publicly available German-language dataset comprising lengthy PDFs. Additionally, we prioritized documents available in both German and English to evaluate whether dense vector embeddings perform worse for German texts.

For this study, we utilized norm documents and industry standards from the Verband der Automobilindustrie (VDA) [40], which ensure quality, safety, and consistency within the automotive sector. These standards cover topics like production processes, quality management, and technical specifications for automotive components. The identical German and English versions of VDA norms made them ideal for studying language impact on embedding performance. We selected 18 norms relevant to production planning, creating a corpus of 18 documents with a total of 2,870 pages in German and 2,874 pages in English.

B. Test Set: Question-Answer Pairs

To evaluate our hybrid retriever, we generated a question-and-answer test by prompting an LLM, specifically Anthropic's Claude Sonnet 3.5 [41]. The entire pipeline consists of the following steps:

(1) Select PDF documents,
(2) Manually cut documents,
(3) Convert to markdown string,
(4) Chunking of documents,
(5) Creation of questions, and
(6) Selection of permissible questions.

Initially, we selected relevant VDA norms from the automotive production planning domain. To ensure content relevance, we manually refined the documents by removing non-essential sections like introductory pages, titles, and concluding pages, which often contain repetitive or irrelevant content. This process resulted in a refined corpus of 2,027 German and 2,025 English pages. We used AWS Textract [42] to convert PDFs into markdown format, retaining high-confidence items, and chunked the documents based on their chapter structure. Finally, we used LangChain's Character Text Splitter [43] to divide the subsections into 1,000-character chunks, approximately equivalent to one page.

Using these chunks, we prompted the LLM to generate three question-answer pairs that a domain expert might ask. To ensure that the generated questions were as relevant as possible, we conducted prompt engineering in collaboration with domain experts from the automotive industry, developing a tailored prompt designed to elicit realistic questions. We then verified the validity of each question by checking whether the corresponding answer was a direct quote from the text. We only accepted pairs where the answers contained at least five words and excluded questions about generic terms like "chapter" or "text". This approach ensured that the retrieved answers matched the text exactly, allowing for precise evaluation. In total, we generated 2,039 questions for the German documents and 1,812 for the English ones.

To ensure diverse retrieval tasks, we designed prompts to generate inquiries with various question words. In German, the most common question words were *was*, *welche*, and *wie*, while in English, they were *what*, *how*, and *when*. These three question words accounted for 88% of inquiries in German and 74% in English. In German, *was* appeared 995 times, *welche* 862 times, and *wie* 649 times. In English, *what* occurred 1,529 times, followed by *how* (533) and *when* (205). This focus on the most common question words ensures diversity in the retrieval tasks and reflects real-world information retrieval scenarios.

C. Comparison Pipeline

Using the questions from Subsection B, we evaluate various hybrid retrieval configurations for the document corpus from Subsection III.A. The pipeline used for this evaluation is shown in Fig. 1.

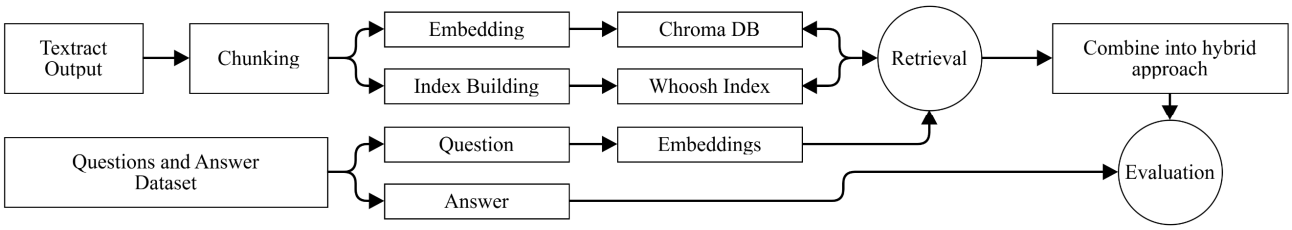

Fig. 1. Retrieval and evaluation pipeline.

We begin by processing the Textract output described in Subsection III.A, first leveraging the document's existing structure (markdown chapters) to form coarse segments and then chunking these segments. Research shows that chunking strategies significantly affect RAG performance [44]. Yepes *et al*. [45] demonstrate that leveraging document structure for chunking improves retrieval, especially in complex domains like financial reports. Given the complexity of our domain-specific documents, we use recursive chunking, where the chapter-based segments are subsequently divided into smaller segments of near-equal length until a desired size is met. We implement this approach using Langchain [43].

To ensure robust results, we experiment with different chunk lengths (300, 600, 900, 1200, and 1500 characters). These chunks are embedded into the Chroma [46] vector database and indexed in *Whoosh* [47]. During retrieval, we use Hierarchical Navigable Small World (HNSW) [48] and cosine distances to find nearest neighbors in Chroma and query the Whoosh index using full-text search algorithms. For each method, we retrieve the top 10 chunks. Although our approach already incorporates high-level semantic structure by aligning chunks with chapters, we focus on length-based chunking within each chapter to systematically compare different chunk sizes.

D. Hybrid Approach

In this section, we present our hybrid approach, which algorithmically combines the top 10 results returned by two different models. The following list outlines the step-by-step procedure of our hybrid algorithm:

(1) We retrieve the top 10 results from both Model 1 and Model 2.
(2) From Model 1, we select the first $n = 3, 5, 7$ results.
(3) We then select the remaining $k = 10 - n$ results from Model 2.
(4) We check for duplicates among the chunks selected in step 2. For each duplicate found, we replace it with a new chunk from Model 2.
(5) We continue this process until we have assembled a total of 10 unique chunks. The final list of returned chunks constitutes our hybrid retrieval.

For clarity, we refer to hybrid models with $n = 3$ and $k = 7$ as a 30/70 model, indication that it comprises 30% from Model 1 and 70% from Model 2. Similarly, we denote models as 50/50 and 70/30 based on their respective contributions. A visualization of our hybrid approach is presented in Fig. 2.

E. Evaluation Method

In the evaluation step, we assess whether the correct answer for each question has been retrieved by checking

for matches within the returned chunks. With smaller chunk sizes (e.g., 300 characters), around 5% of questions may span multiple chunks, affecting accuracy. This issue is most significant with the smallest chunk size. To address this, we consider an answer correct if the ground truth appears within two chunks, as this indicates that the LLM could have answered the question correctly during the RAG process. In our rank-based evaluation, ranks are assigned based on the chunk containing the larger portion of the answer. If the answer is split, this method reconstructs the full ground truth in half of those cases.

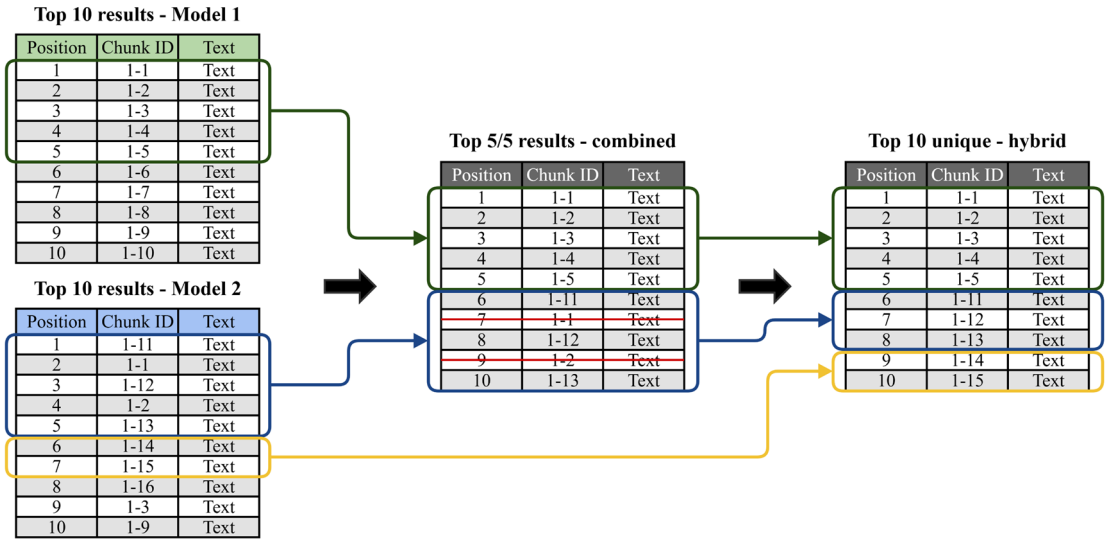

Fig. 2. Hybrid retrieval approach of two models using 50/50 split.

F. Models

In this section, we explain the rationale behind our model choices. Given the need to handle both German and English texts, a multilingual model was essential. Our collaboration with a major automobile manufacturer required a scalable, reliable service.

Since we already use Amazon Web Services (AWS) infrastructure, including AWS Textract for extraction, Amazon's Bedrock service [49] for embedding models was a convenient choice. Additionally, Langchain [50], an open-source LLM framework, integrates well with both Chroma [46] and Amazon Bedrock, simplifying implementation. Thus, we used Amazon Bedrock's multilingual specialized text embedding service, specifically:

- Amazon Titan Text Embeddings V1: titan-embed-text-v1 [51]
- Cohere Multilingual Embed v3: embed-multilingual-v3.0 [52]

For full-text searches, there are fewer constraints. We align the number of models with those used for dense embedding and employ two widely accepted methods for conducting full-text searches:

- BM25F
- TF-IDF

IV. Evaluation

Based on the previously described setup, this section presents the evaluation of our results. We will begin by outlining our evaluation metrics, followed by a presentation of our main findings.

A. Evaluation Metrics

This section outlines the metrics used to evaluate our retrieval pipeline, starting with the primary measure: Precision. Precision quantifies the proportion of questions that successfully retrieve the chunk containing the correct answer, which is crucial for a RAG system to provide correct responses.

Definition 1 (Precision). Precision (positive predictive value) is the fraction of relevant instances among all retrieved ones. A retrieval is considered correct if the ground truth appears in one or two retrieved chunks, as explained in Subsection III.E. Precision is expressed as in Eq. (1):

$$\text{Precision} = \frac{\text{Number of correctly retrieved questions}}{\text{Total number of questions}}. \quad (1)$$

While Precision is key, we also emphasize ranking correct results highly, as LLMs struggle when relevant information is in the middle of the context [53]. To address this, we introduce Mean Reciprocal Rank (MRR), which considers the rank at which a correct chunk is retrieved.

Definition 2 (Mean Reciprocal Rank (MRR)). Let N be the total number of questions and rankj the rank at which the correct answer is returned. If the correct answer is not in the top K returned values, we set $\frac{1}{\text{rank}_j} = 0$. The MRR is given by Eq. (2):

$$\text{MRR} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{\text{rank}_j}. \quad (2)$$

MRR measures the average rank at which a relevant item is found within the top *K* positions. It emphasizes early accuracy, yielding an MRR of 1 when the first recommendation is always relevant and 0 when no relevant recommendations appear in the top *K*. MRR values range from 0 to 1, with higher values indicating better performance. MRR is related to Precision, as the absence of relevant chunks results in a score of zero.

B. Results

In this section, we present our main results, starting with those from simple retrievers, followed by the outcome of our hybrid approach. We first examine the performance of the simple retrieval models. Table I summarizes Precision and MRR in the format (Precision; MRR) for the German documents. The columns in Table I represent chunk lengths specified using the Langchain recursive text splitter [43], while the rows correspond to the models under investigation. To enhance readability, we highlight the best-performing model for each chunk size in **bold** and underline the optimal chunk size for each model. The rows labeled Nr. chunks denote the number of chunks in our corpus based on the selected chunking method, and Avg. chunk length shows the average chunk length. Table II provides similar results for the English documents.

TABLE I. RETRIEVAL PERFORMANCE OF THE MODELS FOR GERMAN LANGUAGE DOCUMENTS

Model/Chunk size	300	600	900	1200	1500
BM25F	(**50.4%**; **0.39**)	(**65.3%**; **0.51**)	(**69.6%**; **0.55**)	(**71.1%**; **0.56**)	(**65.1%**; **0.52**)
TF-IDF	(44.3%; 0.32)	(51.8%; 0.35)	(51.6%; 0.33)	(49.8%; 0.31)	(44.2%; 0.29)
Cohere	(48.3%; 0.36)	(56.3%; 0.41)	(55.1%; 0.40)	(58.3%; 0.41)	(53.0%; 0.38)
Titan	(41.5%; 0.28)	(51.3%; 0.33)	(52.8%; 0.34)	(53.4%; 0.34)	(50.1%; 0.33)
Nr. chunks	15781	7965	5262	3880	3065
Avg. chunk length	209	410	632	859	1088

TABLE II. RETRIEVAL PERFORMANCE OF THE MODELS FOR ENGLISH LANGUAGE DOCUMENTS

Model/Chunk size	300	600	900	1200	1500
BM25F	(**49.0%**; **0.35**)	(62.6%; 0.45)	(**67.7%**; 0.51)	(**67.9%**; **0.51**)	(**69.4%**; **0.51**)
TF-IDF	(40.8%; 0.25)	(50.4%; 0.30)	(53.7%; 0.30)	(52.7%; 0.30)	(52.7%; 0.30)
Cohere	(45.5%; 0.35)	(**64.2%**; **0.49**)	(67.4%; **0.52**)	(66.1%; 0.49)	(66.8%; 0.49)
Titan	(47.3%; 0.34)	(57.9%; 0.41)	(61.3%; 0.43)	(60.7%; 0.42)	(61.5%; 0.42)
Nr. chunks	15357	7705	5081	3758	2974
Avg. chunk length	208	418	636	861	1089

First, we observe that the actual chunk sizes are about 20%–30% smaller than designed, a characteristic of recursive chunking. This is consistent with previous results for English chunks. For German documents, both retrieval percentage and MRR are notably low for very small and large chunk sizes. While English chunks show similar behavior, the decline for larger sizes is less pronounced.

Our analysis of the German VDA norms shows that BM25F outperforms all other models at all chunk sizes, significantly surpassing Cohere. Cohere also outperforms Titan. Both Titan and Cohere perform significantly better on English documents, with Cohere matching BM25F's metrics for English texts. This supports the hypothesis that multilingual embedding models perform worse for non-English texts.

When examining full-text searches, we find that there is no significant difference in retrieval percentage between the models. However, both dense embedding models show significantly better performance for English texts compared to German texts. This discrepancy highlights the challenges associated with non-English retrieval tasks. Fig. 3 provides a clear illustration of the differences in Precision between English and German texts, showing the absolute (abs.) differences based on the German evaluation. Similarly, Fig. 4 displays the MRR differences, presented in the same manner as the Precision differences.

In Table I, we observed that the optimal chunk size for models using German documents was 1200, while for English documents, it was typically 900. To simplify the analysis for hybrid retrieval, we focus on a chunk size of 900, which benefits the LLM during the answer generation step. Results remain consistent across other chunk sizes.

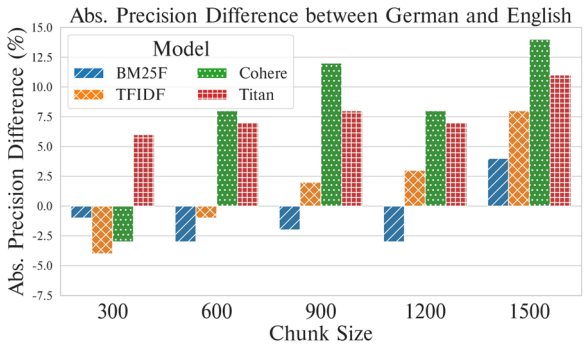

Fig. 3. Precision difference between German and English documents.

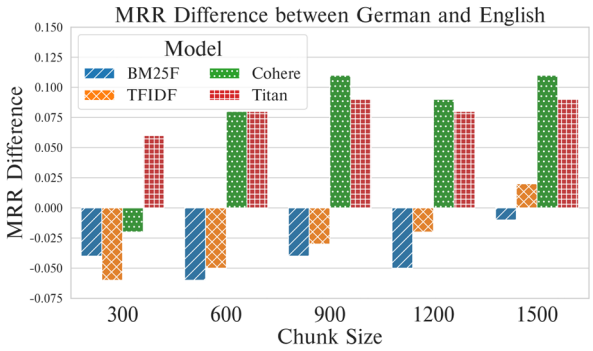

Fig. 4. MRR difference between German and English documents.

Table III presents the retrieval performance of our hybrid models, each consisting of 5 chunks from Model 1 (rows) and 5 chunks from Model 2 (columns). The results for both German and English documents, using a chunk size of 900, are displayed, with the highest Precision and MRR highlighted in **bold**.

TABLE III. PRECISION AND MRR OF 50/50 HYBRID MODELS (CHUNK LENGTH = 900 CHARACTERS)

Model	German Dataset				English Dataset			
	Titan	**Cohere**	**BM25F**	**TF-IDF**	**Titan**	**Cohere**	**BM25F**	**TF-IDF**
Titan		(65.6%; 0.46)	(77.9%; 0.60)	(65.6%; 0.47)		(72.9%; 0.54)	(74.8%; 0.58)	(66.9%; 0.49)
Cohere	(65.0%; 0.49)		(**79.3%**; 0.63)	(69.5%; 0.53)	(72.4%; 0.57)		(**78.9%**; **0.64**)	(72.3%; 0.57)
BM25F	(77.7%; 0.63)	(79.1%; **0.65**)		(66.3%; 0.55)	(75.0%; 0.60)	(78.8%; 0.63)		(65.0%; 0.52)
TF-IDF	(65.7%; 0.47)	(69.4%; 0.52)	(68.0%; 0.48)		(67.3%; 0.46)	(72.7%; 0.51)	(66.4%; 0.44)	

We observe that the percentage of information retrieved by each hybrid model, which combines vector search with full-text search, is significantly higher (by over 10%) than the individual models. Specifically, when combining BM25F with TF-IDF, the performance metrics are inferior to those of BM25F alone but superior to those of TF-IDF individually. This suggests that the combination of BM25F and TF-IDF effectively averages their performance. Meanwhile, the hybrid model, which integrates both vector and full-text searches, leverages their complementary strengths, resulting in overall performance that surpasses that of its individual components. The enhancement seen in the hybrid model further emphasizes the value of combining these two search methods.

Although the two vector models show slight complementarity, their combined effectiveness does not reach the level of the hybrid model. The combination of two vector searches tends to perform better on English texts, while hybrid models that use both vector and full-text searches produce similar results for both German and English documents. This indicates that our model compensates for the comparatively lower performance of vector searches on German texts. The inclusion of full-text search appears to particularly benefit documents in German, where vector-based models alone may struggle.

Additionally, the matrix in Table III is nearly symmetric, suggesting that the performance is not significantly influenced by the order in which the models are applied. This behavior is expected for Precision, as variations in accuracy may arise due to duplicate results being retrieved. The symmetry observed in MRR, however, suggests that when the models retrieve relevant results, they tend to do so early in the ranking, further supporting the efficiency of the hybrid approach.

These observations, as presented in Table III, hold consistently across all chunk sizes. Fig. 5 offers a comparison of the Precision of Cohere and BM25F, our best-performing vector and full-text search models, alongside the 50/50 hybrid model, which combines BM25F (model 1) and Cohere (model 2). This comparison highlights the superior effectiveness of the hybrid approach across chunk sizes and both languages.

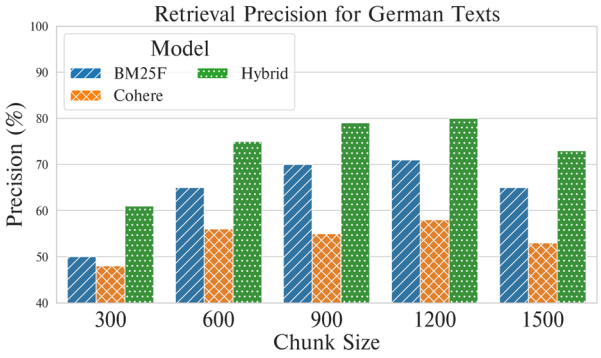

Fig. 5. Precision for BM25F, cohere and the 50/50 hybrid model for the German corpus.

Similarly, Fig. 6 illustrates the precision for the English corpus.

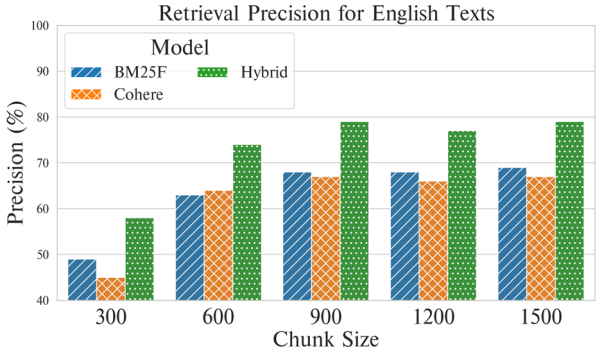

Fig. 6. Precision for BM25F, cohere and the 50/50 hybrid model for the English corpus.

So far, we have only focused on 50/50 in our approach splits. In Table IV, we present the Precision and MRR for these German documents using hybrid models with 30/70 and 70/30 splits, as optimizing RAG for technical German texts is the main goal of our work.

TABLE IV. PRECISION AND MRR OF 30/70 AND 70/30 HYBRID MODELS FOR GERMAN DOCUMENTS

Model	30/70				70/30			
	Titan	**Cohere**	**BM25F**	**TF-IDF**	**Titan**	**Cohere**	**BM25F**	**TF-IDF**
Titan		(65.1%; 0.48)	(77.3%; 0.62)	(64.7%; 0.48)		(64.8%; 0.45)	(76.8%; 0.58)	(64.9%; 0.46)
Cohere	(63.6%; 0.49)		(**78.9%**; 0.65)	(68.9%; 0.54)	(64.8%; 0.49)		(78.4%; 0.62)	(68.1%; 0.52)
BM25F	(76.8%; 0.63)	(78.3%; **0.65**)		(64.3%; 0.55)	(77.5%; 0.62)	(**79.0%**; **0.64**)		(68.1%; 0.55)
TF-IDF	(64.8%; 0.48)	(67.9%; 0.53)	(68.9%; 0.51)		(64.9%; 0.46)	(69.0%; 0.50)	(66.2%; 0.46)	

The results for the 70/30 and 30/70 models align closely with our findings for the 50/50 model, as shown in Table IV. This observation strongly supports our expectation that retrieval primarily occurs at lower ranks, where the effectiveness of the models is most noticeable. However, the results for the 50/50 model consistently show slightly superior performance, further emphasizing the complementary relationship between vector and full-text searches within our specialized corpus. This reinforces the notion that a balanced hybrid approach provides optimal retrieval. In the following section, we will delve deeper into these findings, conducting comprehensive statistical testing to validate and expand upon the insights gained from this analysis.

C. Statistical Testing

Finally, we perform statistical testing to validate our findings using McNemar's test [54], which assesses differences in proportions between two related dichotomous variables. This test is suitable for matched pairs, allowing us to evaluate whether the differences between conditions are statistically significant. We test the null hypothesis, H_0: "There is no significant difference in Precision between BM25F and our hybrid models," formulated as follows in Eq. (3):

$$H_0: PrecisionModel\ 1 = PrecisionModel\ 2 \quad (3)$$

We compare the performance of four models: Cohere, BM25F, and two hybrid models. The first hybrid model, Hybrid 1, combines BM25F as Model 1 and Cohere as Model 2, while Hybrid 2 reverses the order. The tests were conducted using the statsmodels package [55]. Table V shows the test statistics and *p*-values for the different models, based on an analysis of 900 chunks with a 50/50 split for the German corpus.

TABLE V. MCNEMAR'S TEST FOR 50/50 SPLIT COHERE/BM25F HYBRID MODEL FOR GERMAN CORPUS

Model H_0	Model H_1	χ^2-statistic	*p*-value
BM25F	Cohere	143.97	0.000
Cohere	Hybrid 1	581.66	0.000
BM25F	Hybrid 1	185.50	0.000
Hybrid 1	Hybrid 2	0.84	0.359

Table VI presents the results of McNemar's test for the English corpus.

TABLE VI. MCNEMAR'S TEST FOR 50/50 SPLIT COHERE/BM25F HYBRID MODEL FOR ENGLISH CORPUS

Model H_0	Model H_1	χ^2-statistic	*p*-value
BM25F	Cohere	0.10	0.756
Cohere	Hybrid 1	225.51	0.000
BM25F	Hybrid 1	225.79	0.000
Hybrid 1	Hybrid 2	0.06	0.814

Our analysis shows no significant difference between vector and full-text searches for English documents. However, vector search performs notably worse for German texts, highlighting that multilingual vector embedding models are less effective for German. In contrast, our hybrid model significantly outperforms BM25F and Cohere models for both languages. Additionally, the order of models in the hybrid approach does not notably affect the retrieval percentage.

V. DISCUSSION

We find that both simple and hybrid models perform best with medium-sized chunks, averaging 600 to 800 characters, about half a page. Smaller chunks generate a larger database with more irrelevant entries, while larger chunks contain denser information, complicating semantic embedding. Thus, there is a trade-off between using smaller, more distinct chunks that clutter the database and larger chunks that may reduce RAG pipeline performance during generation. Our optimal chunk size of 900 characters remains relatively large, with an average chunk length of about 630 characters. Returning ten such chunks provides around 6,300 characters of context, which modern LLMs can handle, though performance often declines with longer context windows [53]. Addressing this issue may require assigning penalties to larger chunks or incorporating irrelevant chunks to balance database sizes. Pre-selecting relevant documents or utilizing a knowledge graph could also help organize text chunks. We experimented with various chunking parameters but did not explore other methods like semantic chunking, which merits further investigation [56].

Vector embeddings significantly outperform for English chunk sizes, likely because even multilingual models encounter more English texts during training. Full-text searches, however, performed comparably for German and English due to their reliance on word frequency and the similarities between the languages. This result was anticipated, given the identical corpus and linguistic parallels.

Interestingly, BM25F, a simple full-text search algorithm, performs similarly or better than computationally intensive dense embeddings. For instance, in 26% of cases, the most important keyword from generated questions appeared in the context, and one of the three main keywords appeared in 72% of cases. These keywords, extracted with the YAKE algorithm [57], reveal that domain-specific texts often contain rare vocabulary. For English questions, the percentages were 35% and 65%, respectively. Future studies could explore reformulating initial questions to improve retrieval.

Our hybrid approach significantly improves retrieval performance for German and English texts. The retrieval precision and MRR exceed those of individual models, achieving over 80% retrieval accuracy and comparability across languages. This result is surprising, given the poor performance of vector searches for German texts. The hybrid model's complementary use of vector and full-text searches underscores its robustness.

The hybrid models perform consistently regardless of the order of base models or the split ratios (30/70, 50/50, or 70/30). This indicates that relevant chunks are retrieved early, suggesting that passing 10 chunks to an LLM may be excessive. Fewer chunks, such as 6, suffice and help mitigate issues with larger chunks during generation. For example, a 50/50 split with 6 chunks achieves 75%

Precision and an MRR of 0.65 for the BM25F-Cohere hybrid model.

Overall, our approach is computationally efficient, straightforward to implement, and highly effective, making it well-suited for practical application. However, several challenges and limitations must be considered. One key challenge lies in the increased complexity of implementing a hybrid search system, as it requires managing both vector embeddings and the full-text search index in parallel to ensure consistency between the two. To address this issue, our approach utilizes the same identifier for both indexes, streamlining their integration.

A notable limitation of our methodology is the use of LLMs for question generation, even though we incorporated prompt engineering in collaboration with domain experts to tailor the prompts and elicit realistic questions. While these efforts help ensure that the generated questions are relevant, the resultant dataset remains LLM-generated and may not fully capture the nuances of human inquiry. While a human-generated dataset would likely improve the quality of the questions, the time required to create such a dataset makes it nearly impossible to achieve at scale. Moreover, a key challenge is ensuring that the questions remain answerable without incorporating excessive direct carry-over from the answer text, which is critical to avoid artificially boosting full-text search performance.

VI. Conclusion and Outlook

This paper demonstrates that simple hybrid retrieval strategies can outperform state-of-the-art dense and sparse retrievers for domain-specific texts in automotive production planning, especially for German documents. Multilingual vector embedding models prove more effective for English texts, while full-text searches show no significant performance difference between German and English documents. Retrieval accuracy is highest for medium-sized chunks, with large and small chunk sizes performing less effectively.

Future work could explore normalizing chunk impacts by introducing random small chunks into the database or applying penalties to larger chunks to improve RAG pipeline efficiency. These findings provide a foundation for further research into retrieval optimization.

Although our current approach leverages document structure by segmenting text according to chapters before length-based subdivision, future work could investigate more fine-grained semantic chunking methods. Deeper topic or entity-based segmentation may further optimize retrieval performance in highly specialized domains.

Expanding on this work, future studies could investigate chunking strategies beyond retrieval accuracy, such as analyzing vector embedding distributions. While our hybrid approach emphasizes computational efficiency and offers a practical solution for RAG systems handling German domain specific texts, we acknowledge that a detailed analysis of computational resources, including memory usage, indexing time, and query latency, is essential for assessing scalability in large scale deployments. In this initial methodological study, our implementation was designed to explore retrieval efficiency rather than serve as a fully optimized system. In future work, we plan to conduct representative measurements to compare our method's memory and computational demands with more advanced approaches, including the use of German specific embeddings and fine tuning of domain specific models, to provide a comprehensive resource analysis. Despite the added complexity of such approaches, our hybrid model offers a practical and effective solution for RAG systems handling German domain-specific texts.

Furthermore, extending the evaluation to additional non-English languages would help assess the broader applicability of the hybrid retrieval approach, which is particularly valuable for organizations operating in multilingual environments. It would also provide insight into how the performance gap between purely vector-based and hybrid methods may differ across various linguistic contexts.

In addition, future research should consider expanding the dataset to include a broader range of technical documents from diverse domains and industries. This would help validate the generalizability of the hybrid retrieval approach and shed light on how the performance boost of hybrid retrieval varies across different domains and under open-domain conditions.

Moreover, future work should incorporate a more extensive validation of the generated questions by including fully human-generated question-answer pairs or by performing extensive selective verification with domain experts. Such efforts would help reduce potential biases introduced by LLM-generated questions and ensure that the evaluation more accurately reflects real-world query scenarios.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

S.K. led and coordinated the research, developed the experiments, and designed the hybrid approach. He also contributed to writing the paper and creating visual representations. S.P also contributed to the development of experiments and approaches, wrote sections of the paper, implemented the pipeline, and analyzed the results. M.U.A, L.K, S.A., S.G.M, and D.G contributed to the conceptualization of the paper and provided feedback during the revision process. All authors reviewed and approved the final version of the paper.

Funding

The research is funded by the AUDI AG.

Acknowledgment

The presented paper was produced as part of the research project MoFaPro. The present approach was developed within the research cluster "Digital Production"

of the institute "AImotion Bavaria" at the Technische Hochschule Ingolstadt.

REFERENCES

[1] S. Knollmeyer *et al*., "Ontology based knowledge graph for information and knowledge management in factory planning," in *Proc. 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2023, pp. 1–4.

[2] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9459–9474, 2020.

[3] S. Barnett *et al.*, "Seven failure points when engineering a retrieval augmented generation system," in *Proc. the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, 2024, pp. 194–199.

[4] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," arXiv preprint, arXiv:231210997, 2023.

[5] T. Pouplin *et al.*, "Retrieval-augmented thought process as sequential decision making," arXiv preprint, arXiv:240207812, 2024.

[6] R. Nogueira and K. Cho, "Passage re-ranking with BERT," arXiv preprint, arXiv:190104085, 2020.

[7] V. Raina and M. Gales, "Question-based retrieval using atomic units for enterprise RAG," arXiv preprint, arXiv:240512363, 2024.

[8] J. Zhang *et al.*, "Retrieval-based Neural Source Code Summarization," in *Proc. 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 1385–1397.

[9] W. Huang *et al*, "Retrieval augmented generation with rich answer encoding," in *Proc. International Joint Conference on Natural Language Processing*, 2023.

[10] S. Hofstätter *et al.*, "FiD-Light: Efficient and effective retrieval-augmented text generation," arXiv preprint, arXiv:220914290, 2022.

[11] A. Abdallah and A. Jatowt, "Generator-retriever-generator approach for open-domain question answering," arXiv preprint, arXiv:230711278, 2024.

[12] Shangyu *et al*, "Retrieval-augmented generation for natural language processing: A survey," arXiv preprint, arXiv:240713193, 2024.

[13] P. Zhao *et al.*, "Retrieval-augmented generation for ai-generated content: A survey," arXiv preprint, arXiv:240219473, 2024.

[14] Y. Gao *et al.*, "Modular RAG: Transforming RAG systems into LEGO-like reconfigurable frameworks," arXiv preprint, arXiv:240721059, 2024.

[15] J. C. Olamendy. (2024). Contrastive learning: A comprehensive guide. [Online]. Available: https://medium.com/@juanc.olamendy/contrastive-learning-a-comprehensive-guide-69bf23ca6b77

[16] TensorFlow. (2024). Word2Vec. [Online]. Available: https://www.tensorflow.org/text/tutorials/word2vec

[17] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint, arXiv:181004805, 2019.

[18] H. Zaragoza and S. E. Robertson, "The probabilistic relevance framework: BM25 and beyond," in *Foundations and Trends in Information Retrieval*, 2009, pp. 333–389.

[19] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," ArXiv preprint, arXiv:200404906, 2020.

[20] X. Ma *et al*., "A replication study of dense passage retriever," arXiv preprint, arXiv:210405740, 2021.

[21] C. Sciavolino *et al.*, "Simple entity-centric questions challenge dense retrievers," arXiv preprint, arXiv:210908535, 2022.

[22] M. Maki. (2024). OpenAI vs. Open Source multilingual embedding models. [Online]. Available: https://towardsdatascience.com/openai-vs-open-source-multilingual-embedding-models-e5ccb7c90f0

[23] S. Sieranoja and P. Fränti, "Fast and general density peaks clustering," *Pattern Recognit. Lett.*, vol. 128, pp. 551–558, 2019. https://doi.org/10.1016/j.patrec.2019.10.019

[24] X. Ma *et al.*, "Fine-tuning LLaMA for multi-stage text retrieval," arXiv preprint, arXiv:2310.08319, 2023.

[25] O. Ezra, "Retrieval-augmented generation vs fine-tuning: What's right for you?" *K2view Blog*, Aug. 2024.

[26] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2–3, pp. 146–162, 1954.

[27] A. Rajaraman and J. D. Ullman, "Data mining," in *Mining of Massive Datasets*, Cambridge University Press, 2011.

[28] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval,* USA: Cambridge University Press, 2008.

[29] N. Arabzadeh, X. Yan, and C. L. A. Clarke, "Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection," arXiv preprint, arXiv:210910739, 2021.

[30] Y. Luan *et al.*, "Sparse, dense, and attentional representations for text retrieval," *Trans. Assoc. Comput. Linguist.*, vol. 9, pp. 329–345, Apr. 2021. doi: 10.1162/tacl_a_00369

[31] N. Thakur *et al.*, "BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models," arXiv preprint, arXiv:210408663, 2021.

[32] P. P. Acharya. (2024). Hybrid search in RAG: Enhancing information retrieval with combined techniques. [Online]. Available: https://www.linkedin.com/pulse/hybrid-search-rag-enhancing-information-retrieval-prasanna-cn4dc/

[33] A. Abraham *et al.* (2024). Optimizing RAG with hybrid search & reranking. [Online]. Available: https://github.com/superlinked/VectorHub/blob/main/docs/articles/hybrid_search

[34] Stack Overflow. [Online]. Available: https://stackoverflow.com/

[35] D. Haney and D. Gibson. (2023). Ask like a human: Implementing semantic search on stack overflow. [Online]. Available: https://stackoverflow.blog/2023/07/31/ask-like-a-human-implementing-semantic-search-on-stack-overflow/

[36] R. Theja. (2024). LlamaIndex: Enhancing retrieval performance with alpha tuning in hybrid search in RAG. [Online]. Available: https://www.llamaindex.ai/blog/llamaindex-enhancing-retrieval-performance-with-alpha-tuning-in-hybrid-search-in-rag-135d0c9b8a00

[37] R. Benham, "Risk-reward trade-offs in rank fusion," in *Proc. 22nd Australas. Doc. Comput. Symp.*, 2017.

[38] H. Zeng *et al.*, "Federated recommendation via hybrid retrieval augmented generation," arXiv preprint, arXiv:240304256, 2024.

[39] Y. Pu *et al.*, "Customized retrieval augmented generation and benchmarking for EDA tool documentation QA," arXiv preprint, arXiv:240715353, 2024.

[40] Verband der Deutschen Automobilindustrie (VDA). (2024). [Online]. Available: https://www.vda.de

[41] Anthropic. (2024). Claude 3.5. [Online]. Available: https://www.anthropic.com/news/claude-3-5-sonnet

[42] Amazon Web Services. Amazon Textract. [Online]. Available: https://docs.aws.amazon.com/textract

[43] LangChain. (2024). LangChain Document Transformers—Data Connection. [Online]. Available: https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/

[44] D. Ru *et al.*, "RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation," arXiv preprint, arXiv:240808067, 2024.

[45] A. Yepes *et al.*, "Financial report chunking for effective retrieval augmented generation," arXiv preprint, arXiv:240205131, 2024.

[46] Chroma Inc. (2024). [Online]. Available: https://docs.trychroma.com/

[47] M. Chaput. (2024). Whoosh. [Online]. Available: https://whoosh.readthedocs.io/Version 2.7.4

[48] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," arXiv preprint, arXiv:160309320, 2018.

[49] Amazon Web Services. (2024). Amazon Bedrock. [Online]. Available: https://aws.amazon.com/de/bedrock/

[50] O. Topsakal and T. C. Akinci, "Creating large language model applications utilizing Langchain: A primer on developing LLM apps fast," in *Proc. International Conference on Applied Engineering and Natural Sciences*, 2023, pp. 1050–1056.

[51] Amazon Web Services, Inc. (2024). Amazon Titan Models. [Online]. Available: https://docs.aws.amazon.com/bedrock/latest/userguide/titan-models.html

[52] Cohere. (2024). Cohere Documentation. [Online]. Available: https://docs.cohere.com/docs/cohere-embed

[53] N. F. Liu *et al.*, "Lost in the middle: How language models use long contexts," arXiv preprint, arXiv:230703172, 2023.

[54] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947. doi: 10.1007/bf02295996
[55] S. Seabold and J. Perktold, "Statsmodels," in *Proc. 9th Python in Science Conference*, 2010.
[56] K. Rathinasamy, "EnterpriseEM: Fine-tuned embeddings for enterprise semantic search," arXiv preprint, arXiv:240600010, 2024.
[57] R. Campos *et al.*, "YAKE! Keyword extraction from single documents using multiple local features," *Inf. Sci.*, vol. 509, pp. 257–289, Jan. 2020. doi: 10.1016/j.ins.2019.09.013