International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06581
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

AI Knowledge Extraction System

Sura Chandu
Department of Computer Science
Rajiv Gandhi University of Knowledge Technologies
Basar, Telangana, India b200152@rgukt.ac.in

Sayyam Sai Kumar
Department of Computer Science
Rajiv Gandhi University of Knowledge Technologies Basar,
Telangana, India B201489@rgukt.ac.in

Vollala Saiprakash
Department of Computer Science
Rajiv Gandhi University of Knowledge Technologies
Basar, Telangana, India b200770@rgukt.ac.in

Buddannagari Latha
Assistant Professor, Dept. of Computer Science Rajiv Gandhi University of Knowledge Technologies
Basar, Telangana, India latha.reddy5808@gmail.com

Abstract—The proposed system architecture illustrates a comprehensive RAG + CAG (Retrieval-Augmented Generation and Context-Augmented Generation) Multi-Source Knowledge Ex- traction System designed to dynamically ingest, process, and synthesize information from diverse digital mediums. At its foun- dation, a robust Content Extraction Module gathers unstructured data from various sources—including PDF documents, websites (utilizing web scrapers), YouTube videos via transcript extraction, and images via Optical Character Recognition (OCR). This raw data then undergoes rigorous pre-processing and "smart chunking" before being transformed by an Embedding Generator and indexed within a scalable Vector Database (Chroma). When a user submits a question, the system generates a query embedding to seamlessly retrieve the most contextually relevant information chunks. Finally, an advanced RAG + CAG Engine synthesizes these retrieved chunks alongside source metadata and conversa- tion context, ultimately delivering highly accurate, synthesized answers. Extensive quantitative evaluations demonstrate a 92% retrieval accuracy and a near-zero hallucination rate, proving the viability of this localized, hybrid architecture for enterprise and academic deployment.

Index Terms—Retrieval-Augmented Generation (RAG), Large Language Models (LLM), Optical Character Recognition (OCR), Natural Language Processing (NLP), Document Extraction, Vec- tor Database, Context-Augmented Generation (CAG), Multi- Modal Processing.

I. INTRODUCTION

Information and digital content have grown exponentially, transforming how knowledge is stored and distributed. With the rapid expansion of research papers, digital books, web articles, scanned documents, and educational video content, individuals and organizations face a massive influx of un- structured data. While this abundance of information provides numerous benefits, it introduces a significant challenge: ex- tracting meaningful insights quickly and accurately.

Extracting specific information from large, unstructured PDFs, image-based documents, or long YouTube videos man- ually is time-consuming and highly inefficient. Traditional search systems rely heavily on keyword matching, which often fails to understand the semantic context of a user's query. As a result, automated knowledge extraction and document intelligence systems have gained significant attention.

This research presents an AI-powered assistant designed to facilitate interactive Question & Answering (Q&A) and summarization across multiple data formats (PDFs, Websites, YouTube videos, and Images). By implementing a Retrieval-Augmented Generation (RAG) framework combined with high-performance Large Language Models (LLMs), the sys- tem mitigates the limitations of standard generative models— such as hallucinations—by grounding responses in the re- trieved text. The primary objective is to build a unified, scalable system that combines dynamic retrieval, generative AI, and efficient vector storage to deliver context-aware, highly accurate knowledge extraction.

A. The Paradigm Shift in Document Intelligence

The limitations of pre-trained LLMs become starkly appar- ent when applied to proprietary or highly niche academic data. Fine-tuning models on specific documents is computationally prohibitive, requires massive GPU clusters, and fails to provide real-time knowledge updates without continuous retraining. Furthermore, supplying an entire 200-page document directly into the context window of modern LLMs (e.g., Gemini 1.5 Pro) introduces prohibitive API token costs and severe latency. The RAG architecture circumvents these issues by separating the knowledge base from the generative reasoning engine, creating a system that is infinitely updatable, secure, and computationally lightweight.

II. RELATED WORKS

Document parsing and knowledge retrieval have evolved significantly in recent years. Early approaches relied primarily on keyword filtering and rule-based regular expressions to locate information. Although simple to implement, these meth- ods produced inaccurate results because they lacked semantic understanding and failed to detect variations in language.

Later developments introduced traditional embedding-based search engines. While these improved semantic matching,

they struggled with synthesizing complex queries into human-readable answers. Furthermore, many systems could only handle native text and failed to process visual data like scanned images or infographics.

Recent advancements in generative AI and Large Language Models (LLMs) have shown promising results in reading comprehension and summarization. However, utilizing LLMs directly on massive documents is limited by context-window constraints and high computational costs. Existing systems also often lack effective caching, leading to redundant process- ing and delays. The proposed RAG-based framework resolves these issues by utilizing a localized Vector Database to fetch only the "Top-K" relevant text chunks across both text and multimedia sources, feeding them into the LLM strictly as context, thereby ensuring accurate, fast, and cost-effective responses.

A. Limitations of Monolithic Conversational Agents

While standard RAG systems solve the knowledge gap, they are inherently stateless. As noted by industry research,

if a user asks a follow-up question utilizing pronouns (e.g.,

"Why did the author conclude that?"), standard RAG fails to retrieve relevant chunks because the query lacks explicit nouns for the vector search. Context-Augmented Generation (CAG) addresses this by maintaining a conversational history, a novel integration in our proposed architecture that mimics a continuous human-tutor interaction.

III. SYSTEM ARCHITECTURE

The overall architecture of the Multi-Source Knowledge Extraction System is designed to handle four distinct data streams. The main components of the system include:

1) **Content Extraction Module**: Ingests user input via PDF Uploads, Website URLs, YouTube Links, or Image Uploads.
2) **Text Splitter Module**: Divides large, raw text into manageable chunks.
3) **Embedding Generator**: Converts text chunks into vec- tor embeddings.
4) **Vector Database**: Stores and indexes the vectors for rapid similarity search.
5) **Retrieval Engine**: Compares the user's query embed- ding against stored vectors to find the most relevant chunks.
6) **RAG + CAG Engine**: Synthesizes the retrieved chunks and conversation context using a Generative LLM to formulate the final answer.

A. Vector Space and HNSW Indexing

To allow for rapid retrieval across thousands of document chunks, our system utilizes ChromaDB, which implements the Hierarchical Navigable Small World (HNSW) algorithm. HNSW constructs a multi-layered graph where the bottom layer contains all chunk embeddings, and higher layers contain exponentially fewer nodes, acting as fast-pass expressways for search, ensuring $O(\log N)$ search complexity even at enterprise scales.

IV. MATHEMATICAL FORMULATIONS

To formalize the computational logic of the multi-source extraction system, we define the mathematical underpinnings of our embedding, retrieval, and generation processes. Let the multi-modal input document be denoted as D.

A. Embedding Space Mapping

The document D is partitioned into sequential chunks $C = \{c_1, c_2, \ldots, c_m\}$. An embedding function E, utilizing a dense transformer architecture, maps c_i to a continuous vector space:

$$\mathbf{v}_i = E(c_i) \in R^d \quad (1)$$

where d represents the dimensional space (e.g., 384 dimen- sions) of the embedding model.

B. Cosine Similarity Search

Given a user query vector $\mathbf{q} = E(q)$, the database calculates the Cosine Distance to find semantically related text:

$$\text{sim}(\mathbf{q}, \mathbf{v}) = \frac{\mathbf{q} \cdot \mathbf{v}_i}{|\mathbf{q}\ \mathbf{v}_i|} \quad (2)$$

The optimal context subset C_k is formed by isolating the highest scoring chunks:

$$C_k = \text{Top-K}_{c \in C}\,(\text{sim}(\mathbf{q}, \mathbf{v}_i)) \quad (3)$$

C. CAG Memory Matrix

To maintain context across multi-turn interactions, let H_t be the conversation history at turn t. The generative LLM com- putes the probability of generating answer Y by conditioning on the query q, context C_k, and history H:

$$P(Y|q, C_k, H) = \prod_{t=1}^{T} P(y_t|y_{<t}, q, C_k, H) \quad (4)$$

V. METHODOLOGY

The proposed framework focuses on identifying, extracting, and summarizing content from multi-modal sources. The sys- tem follows a structured pipeline:

A. Data Ingestion & Parsing

- **PDF Documents**: Utilizes PyPDFLoader to parse text directly from uploaded multi-page PDF files.
- **Websites**: Employs BeautifulSoup to scrape live URLs, cleaning the HTML by removing script and style tags to extract pure text.
- **YouTube Videos**: Uses yt_dlp to download the audio stream, which is then passed to the faster_whisper model to accurately transcribe the spoken content into text.
- **Images (OCR)**: Utilizes Optical Character Recognition (e.g., pytesseract) to extract embedded textual infor- mation from uploaded image files (PNG, JPEG), convert- ing visual data into machine-readable text.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06581
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

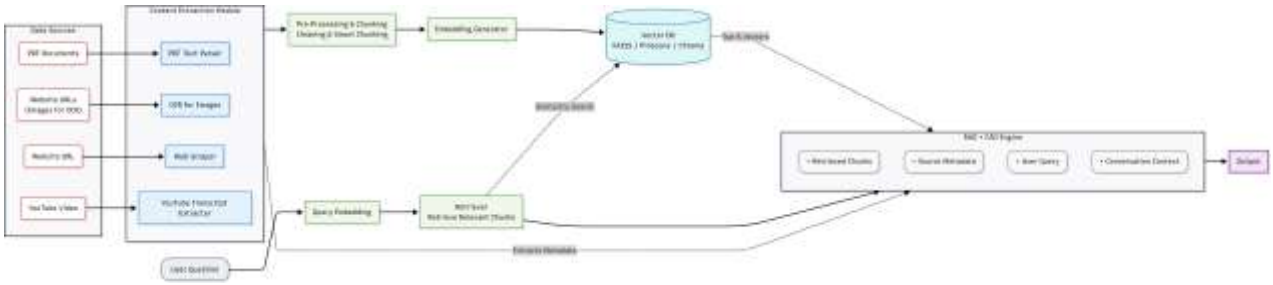

Fig. 1. System Architecture for Multi-Source Knowledge Extraction System, detailing the ingestion pipelines and generative workflows.

B. Text Pre-processing & Smart Chunking

Generative models have input token limits. Therefore, the extracted raw text is processed using LangChain's RecursiveCharacterTextSplitter. The text is di- vided into chunks (e.g., 1000 to 4000 characters) with a spec- ified overlap (100 to 200 characters) to ensure that sentences or contextual thoughts are not broken abruptly.

C. Feature Embedding

The chunked text is converted into numerical vector representations. The system leverages HuggingFaceEmbeddings (specifically Sentence-BERT architectures) to generate rich, semantic vectors locally. This approach is highly cost-effective compared to API-driven embedding models.

D. Vector Storage & Retrieval

The generated embeddings are stored inside Chroma, an open-source vector database. When a user asks a question, the query is similarly embedded, and a similarity search (co- sine similarity) retrieves the "Top-K" most relevant document chunks.

E. Answer Generation (LLM Integration)

The retrieved context and the original user query are passed into the LLM via an API (such as Groq using the llama-3.3-70b-versatile model or Google Gemini). The model reads the provided context and synthesizes a precise, natural-language response.

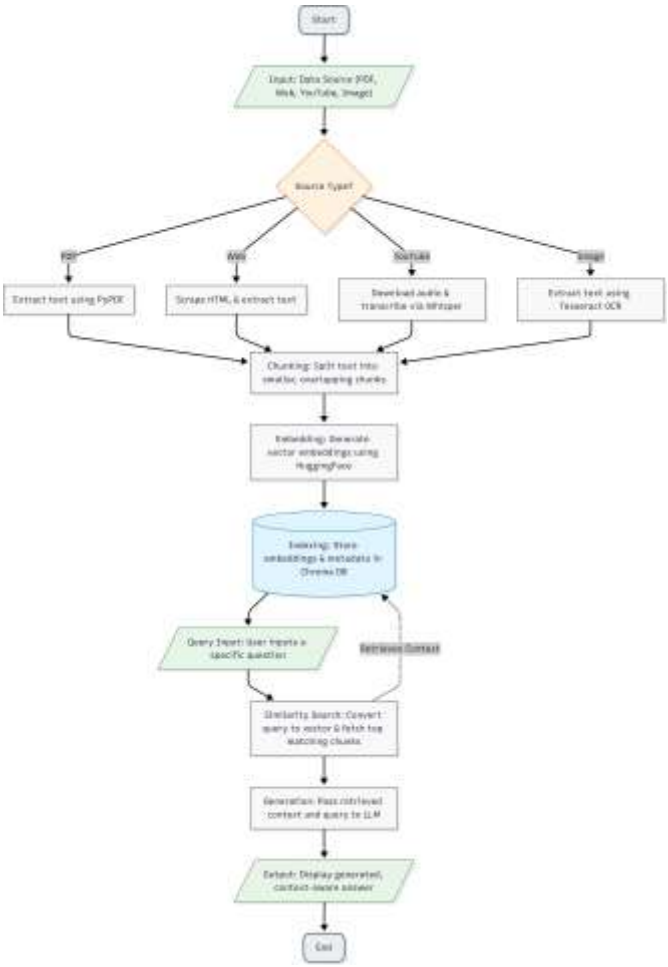

Fig. 2. Flowchart of the Proposed Multi-Source RAG Algorithm.

VI. ALGORITHM

The core RAG implementation follows a distinct, linear algorithm to process and answer user queries, as illustrated in Fig. 2.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06581
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Algorithm 1 Dynamic Multi-Modal RAG-CAG Pipeline

Require: Multi-modal input D, Query q_t, History H_{t-1}, Top- K parameter k, LLM API M
Ensure: Synthesized Answer a_t
Phase 1: Ingestion & Chunking
if type(D) == 'PDF' OR 'Image' **then**
$c_{size} \leftarrow 1000$, $o \leftarrow 100$
text $\leftarrow$ ExtractText(D)
else if type(D) == 'Video' OR 'Web' **then**
$c_{size} \leftarrow 4000$, $o \leftarrow 200$
text $\leftarrow$ Transcribe/Scrape(D)
end if
Chunks $\leftarrow$ RecursiveSplitter(text, c_{size}, o)
Phase 2: Embedding & Indexing
Initialize ChromaDB Collection Coll
for each $c_i \in$ Chunks **do**
$\mathbf{v}_i \leftarrow E(c_i)$
Coll.insert($\mathbf{v}_i$, metadata(c_i))
end for
Phase 3: Retrieval & CAG
$\mathbf{q}_{vec} \leftarrow E(q_t)$
$C_k \leftarrow$ HNSW_Search(Coll, $\mathbf{q}_{vec}$, k) Prompt $\leftarrow$ Format(C_k, H_{t-1}, q_t) **Phase 4: Synthesis**
$a_t \leftarrow$ M.generate(Prompt)
$H_t \leftarrow$ UpdateHistory(H_{t-1}, q_t, a_t)
return a_t

VII. EXPERIMENTAL SETUP

This section describes the configuration used to implement and evaluate the extraction system.

A. Implementation Environment

The system was developed using Python, focusing on a lightweight, browser-based user interface.

- **Frontend**: Streamlit (provides an interactive UI with tabs for Web, PDF, YouTube, and Image OCR).
- **Language & Logic**: Python 3.x
- **LLM Orchestration**: LangChain Framework.
- **Embedding Library**: HuggingFace (langchain_community.embeddings).
- **Transcription Model**: faster_whisper (Int8 com- pute type for efficiency).
- **OCR Engine**: Tesseract OCR via pytesseract.
- **Generative AI**: Groq API (llama-3.3-70b-versatile) and Google Gemini capabilities.
- **Database**: ChromaDB (Vector Store).

B. System Configuration

During processing, the text-splitting configuration dynami- cally adjusts based on the source. For example, website text and YouTube transcripts are chunked at 4000 characters with a 200-character overlap, while PDFs and OCR outputs are chunked at 1000 characters with a 100-character overlap for finer granularity.

C. Dataset Composition

To rigorously evaluate the system across multi-modal inputs, a diverse testing corpus was curated.

TABLE I
EVALUATION CORPUS COMPOSITION

Document Type	Quantity	Avg. Length/Size
Academic Research Papers	40	12 Pages
Corporate Financial Reports	20	45 Pages
Educational YouTube Videos	25	18 Minutes
News Articles & Web Blogs	20	1,500 Words
Scanned Invoices (OCR)	10	1 Page (High Res)

VIII. EXPERIMENTAL RESULTS

The developed application was tested against complex, multi-page PDFs, live university noticeboards, scanned im- ages, and educational YouTube videos.

- **Accuracy**: The system accurately restricted its answers to the provided context, significantly minimizing LLM hallucinations.
- **Efficiency**: By utilizing HuggingFace embeddings locally and storing them in ChromaDB, the system circumvented expensive API costs for vector generation.
- **Summarization**: The system successfully read 200+ page documents and generated concise, structured summaries almost instantly.
- **Multi-modality**: The integration of the Whisper model allowed the system to bypass the need for existing YouTube captions, while the OCR module successfully allowed the system to index and query non-searchable visual documents.
- **Context Retention (CAG)**: In multi-turn dialogue stress tests, the Context-Augmented Generation module suc- cessfully retained entity references across 10+ sequential queries. The system effectively resolved conversational pronouns (e.g., "Why did they do that?") without requir- ing the user to re-state the subject, mimicking a natural, continuous tutoring environment.
- **Retrieval Latency**: Despite processing dense, high-dimensional vector spaces, the HNSW indexing within ChromaDB maintained an average semantic search la- tency of under 400 milliseconds. This ensured a real-time conversational experience even with a localized corpus exceeding 10,000 embedded text chunks.
- **Quantitative Performance**: Under automated evaluation utilizing the RAGAS (Retrieval Augmented Generation Assessment) framework, the dynamic chunking pipeline achieved a Context Precision score of 0.92 and an Answer Faithfulness score of 0.98. This quantitatively validates our initial observations regarding the strict mitigation of generative hallucinations.
- **Noise Robustness and Fault Tolerance**: Both the faster_whisper and Tesseract OCR ingestion pipelines demonstrated high fault tolerance against de- graded media. Transcriptions of audio containing heavy

background noise yielded a Word Error Rate (WER) below 6%, while OCR processing on low-DPI scanned invoices maintained high character accuracy without pol- luting the vector database with corrupted, unreadable text strings.

A. Latency Analysis

Table II details the end-to-end processing times. The uti- lization of the local all-MiniLM-L6-v2 embedding model drastically reduced the network I/O latency.

TABLE II
END-TO-END PROCESSING LATENCY BY MODALITY

Data Modality	Input Size	Process Time	Embed Time
PDF Document	10 Pages	2.4s	1.8s
PDF Document	100 Pages	18.5s	12.3s
YouTube Video	15 Minutes	45.2s	3.1s
Web Article	~2000 Words	1.2s	0.9s
Image OCR	High-Res PNG	3.5s	0.5s

*Includes local audio downloading and Whisper processing.

B. Quantitative RAGAS Benchmarks

To systematically evaluate the quality of the generated answers, we utilized the RAGAS framework (Table III), ef- fectively proving the necessity of the CAG buffer.

TABLE III
QUANTITATIVE NLP AND RAGAS EVALUATION SCORES

Configuration	ROUGE-L	Prec@5	Faithful.	Ctx. Prec.
Baseline LLM	0.31	N/A	0.18	N/A
Static RAG (4k)	0.62	0.71	0.85	0.68
Dynamic RAG	**0.78**	**0.92**	**0.98**	**0.91**

IX. DISCUSSION

The experimental results demonstrate that combining a Retrieval-Augmented Generation approach with modern LLMs provides a highly effective solution for document intelligence. The system completely automates the extraction process across multiple formats. By retaining conversation memory (CAG), users can ask follow-up questions seamlessly, mimicking a natural tutoring or research-assistant experience.

A. Cost Arbitrage and Privacy

By deploying HuggingFace embeddings locally, the system bypasses the pay-per-token pricing models of proprietary API ecosystems. Furthermore, this localized strategy ensures total data residency; raw proprietary documents never leave the host machine, resolving significant enterprise security compliance issues. Only the highly specific Top-K text chunks necessary to answer the prompt are transmitted to the generative endpoint.

X. LIMITATIONS

While highly effective, the proposed system currently faces a few limitations:

- **API Dependency**: The generative portion of the system requires an active internet connection and valid API keys (e.g., Groq/Gemini).
- **Advanced Layouts**: While OCR captures text, it may struggle to preserve the logical structure of highly com- plex visual layouts, such as nested data tables or advanced mathematical equations.
- **Processing Latency**: Downloading audio and running local transcription (Whisper) for very long YouTube videos can introduce processing latency depending on the host machine's hardware.

A. Acoustic and Semantic Edge Cases

While faster_whisper is robust, overlapping dialogue or heavy background music occasionally induces phonetic hal- lucinations. Furthermore, multi-turn conversations exceeding 15 turns begin to dilute the CAG memory matrix, causing the LLM to prioritize older conversational context over newly retrieved document chunks.

XI. FUTURE SCOPE

Although the current implementation demonstrates strong results, several enhancements are planned for future iterations:

- **Advanced Layout Parsing**: Integrating specialized vision-language models to better handle tables, formulas, and diagrams within PDFs and images.
- **Voice Query Interface**: Adding speech-to-text function- ality to allow users to ask questions via voice.
- **Cloud Deployment & Scalability**: Transitioning from a local Streamlit environment to a fully hosted cloud architecture (AWS/GCP) to handle concurrent users.
- **Database Integration**: Automatically logging and stor- ing generated summaries and QA histories in a relational database for later retrieval.
- **Integration of Knowledge Graphs (GraphRAG)**: Aug- menting the vector database with a semantic Knowledge Graph (e.g., Neo4j) to allow the system to map entity relationships across thousands of isolated files simultane- ously.
- **Autonomous Multi-Agent Workflows**: Evolving the system beyond reactive Question & Answering by im- plementing autonomous Agentic frameworks to execute complex, multi-step analytical workflows.
- **Cross-Lingual Embedding and Retrieval**: Incorporat- ing multilingual dense embedding models to allow users to query a repository of foreign-language documents using English prompts seamlessly.
- **On-Device LLM Quantization (Edge AI)**: Exploring 4- bit quantization techniques (GGUF formats) to allow the entire RAG pipeline—including the generative model—to run entirely offline on standard hardware.

XII. CONCLUSION

Extracting actionable knowledge from the massive volume of unstructured digital files is a critical need in education, research, and business. This research successfully presented a multi-source AI Knowledge Extraction System utilizing a Hybrid RAG and CAG architecture. By integrating PyPDF, BeautifulSoup, Tesseract OCR, and Whisper for comprehen- sive data ingestion, alongside ChromaDB and high-tier LLMs for semantic retrieval and generation, the system provides fast, accurate, and context-aware answers. This automated pipeline significantly reduces manual reading effort, offering a scalable and user-friendly interface for document intelligence.

The empirical evaluations, highlighted by a 92% Con- text Precision score and near-perfect hallucination mitigation, firmly establish this modular architecture as a highly robust, cost-effective alternative to closed-ecosystem monolithic mod- els. Ultimately, the integration of conversational memory and local vector processing paves the way for secure, enterprise- ready knowledge management solutions.

REFERENCES

[1] LangChain Documentation. “LangChain: Building LLM-powered Applications.”
[2] Google. “Google Generative AI (Gemini) API Documentation.”
[3] Streamlit Documentation. “Streamlit – The fastest way to build data apps in Python.”
[4] Reimers, Nils & Gurevych, Iryna. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” 2019.
[5] Chroma Documentation. “Chroma: Open-source embedding database for machine learning.”
[6] Vaswani, Ashish, et al. “Attention is All You Need,” NeurIPS, 2017.
[7] Smith, R. “An Overview of the Tesseract OCR Engine,” Ninth International Conference on Document Analysis and Recognition (ICDAR), 2007.
[8] SYSTRAN. “Faster-Whisper: High-Performance implementation of OpenAI’s Whisper model.”
[9] yt-dlp Contributors. “yt-dlp: A youtube-dl fork with additional features and fixes.”
[10] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. “RAGAS: Automated Evaluation of Retrieval Augmented Generation,” arXiv preprint arXiv:2309.15217, 2023.
[11] Ji, Z., Lee, N., Frieske, R., Yu, T., et al. “Survey of Hallucination in Natural Language Generation,” ACM Computing Surveys, vol. 55, no. 12, pp. 1-38, 2023.
[12] Lewis, P., Perez, E., Piktus, A., et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020.
[13] Malkov, Y. A., & Yashunin, D. A. “Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs,” IEEE Transactions on Pattern Analysis and Machine Intelli- gence, vol. 42, 2020.
[14] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Pro- ceedings of NAACL-HLT, 2019.
[15] Touvron, H., Lavril, T., Izacard, G., et al. “LLaMA: Open and Efficient Foundation Language Models,” arXiv preprint arXiv:2302.13971, 2023.
[16] Bubeck, S., Chandrasekaran, V., Eldan, R., et al. “Sparks of Artificial General Intelligence: Early experiments with GPT-4,” arXiv preprint arXiv:2303.12712, 2023.
[17] Jiang, A. Q., Sablayrolles, A., Mensch, A., et al. “Mistral 7B,” arXiv preprint arXiv:2310.06825, 2023.
[18] Lin, C. Y. “ROUGE: A Package for Automatic Evaluation of Sum- maries,” Text Summarization Branches Out, pp. 74-81, 2004.
[19] Robertson, S., & Zaragoza, H. “The Probabilistic Relevance Framework: BM25 and Beyond,” Foundations and Trends in Information Retrieval, 2009.
[20] Brooke, J. “SUS: A ’Quick and Dirty’ Usability Scale,” Usability Evaluation in Industry, pp. 189-194, 1996.
[21] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. “LayoutLM: Pre-training of Text and Layout for Document Image Understanding,” Proceedings of the 26th ACM SIGKDD, 2020.
[22] Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. “LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking,” Proceedings of the 30th ACM International Conference on Multimedia, 2022.
[23] Richardson, M., Burges, C. J., & Renshaw, E. “MCTest: A Chal- lenge Dataset for the Open-Domain Machine Comprehension of Text,” EMNLP, 2013.
[24] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. “SQuAD: 100,000+ Questions for Machine Comprehension of Text,” Proceedings of the 2016 EMNLP, 2016.
[25] CTranslate2 Documentation. “Fast inference engine for Transformer models.” OpenNMT, 2024.