Crop Advisory Chatbot System for Soybean Farmers

Mou Sarkar, Pratham Prajapati, Rahul Dewangan, S Abhinav Raj, and Sanjay Chatterji

Indian Institute of Information Technology Kalyani, Kalyani, India

Abstract. This paper presents the design, implementation, and evaluation of a *Soybean Crop Advisory Chatbot* that utilizes Retrieval-Augmented Generation (RAG) techniques and Large Language Models (LLMs) to insight the farmers with precise, context-aware recommendations. The system integrates diverse data sources (research articles, extension bulletins, crop tables) into a unified knowledge base. Using semantic embedding and vector storage (via ChromaDB), the chatbot retrieves relevant information in response to user queries and formulates answers through a language model pipeline (LangChain) with prompt tuning for clarity and farmer-friendly language. Key challenges, such as extracting English content from bilingual PDFs, merging sentence fragments, choosing optimal text chunk sizes, and simplifying technical language for non-expert users, are addressed with custom processing strategies. We report development details including system architecture, data preprocessing, embedding generation, and prompt design. Sample queries and responses demonstrate the chatbot's capabilities. Evaluation on test queries indicates high retrieval precision and user-friendly performance, suggesting the system's potential to improve soybean farming practices. The work discusses the limitations and future enhancements.

Keywords: Soybean Crop Advisory, Chatbot, Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), Semantic Embedding, ChromaDB, Data Preprocessing, Prompt Tuning, LangChain.

1 Introduction

Agricultural development is a key driver of economic growth, food security, and rural livelihoods. In countries like India, smallholder farmers rely heavily on accurate and timely crop advisory to make critical decisions regarding sowing, pest control, fertilization, and irrigation. Among major crops, soybean plays an essential role in India's Kharif season. However, despite well-documented practices and government-issued advisories from institutions such as ICAR, farmers struggle to access, understand, and act upon the information due to language barriers, format limitations, and lack of personalization.

David C. Wyld et al. (Eds): ACINT, CSITAI, ITEORY, MATHCS, AICiViL, ENERGY – 2025
pp. 35-46, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152504

The rise of AI-powered chatbots, notably those based on Large Language Models (LLMs), offers a novel solution. By converting unstructured, bilingual advisory documents into a searchable, semantically rich knowledge base, and pairing it with natural language generation capabilities, such system can provide on-demand, personalized support to farmers.

In this paper, we discuss an intelligent chatbot interface that can interpret farmer queries and return simplified advice based on actual ICAR advisories using Retrieval-Augmented Generation (RAG) architecture. The queries about prediction of weather, crop pesticides in case of disease occurance in specific crop. The chatbot operates offline and uses open-source tools like LangChain, ChromaDB, and Ollama.

This work makes the following key contributions:

- Proposes a practical solution for converting government PDF advisories into chatbot-usable structured English content.
- Develops a full pipeline including PDF scraping, language filtering, sentence merging, and chunk embedding.
- Implements a Retrieval-Augmented Generation (RAG) framework using LangChain and Ollama.
- Demonstrates a lightweight, offline-compatible chatbot using open-source tools.
- Evaluates the chatbot with farmer-style questions and provides performance benchmarks on relevance and usability.

2 Related Works

Agricultural extension services have historically depended on field agents disseminating knowledge through physical pamphlets, training programs, and local demonstration farms. Although conventional approaches are effective to some extent, these traditional methods often suffer from scalability, personalization, and timeliness [1, 2].

Recent studies underscore the transformative role of Information and Communication Technologies (ICT) in extension. Tools such as India's *mKisan* SMS platform and Digital Green's community-based dissemination systems have significantly improved farmers' access to real-time information such as weather updates, pest alerts, and market prices [3, 4]. ICT-enabled systems allow location-specific advisory through mobile phones and provide AI-powered predictive insights for smarter decision-making. However, literature notes challenges remain—especially in rural areas—such as the digital divide, low digital literacy, and the lack of support for regional languages [3].

AI-powered chatbots are emerging as a promising modality for agricultural extension. These systems allow farmers to access agricultural advice through natural language conversations, often available 24/7. For example, a pilot chatbot in

Ethiopia provided support to extension agents by answering agricultural questions in Amharic, leveraging a Q&A corpus [5]. Similarly, a Marathi-language chatbot in India employed NLP techniques and a Naïve Bayes classifier to deliver real-time crop and pest-management advice [7].

These implementations demonstrate that AI-based chatbots can improve inclusivity by providing responses in local languages and presenting expert knowledge in a simplified manner. Moreover, language-specific training has shown to increase engagement in rural contexts [8].

However, early agricultural chatbots relied heavily on rule-based systems or supervised models. A review of chatbot designs shows reliance on keyword matching or sequence-to-sequence (seq2seq) RNNs trained on curated datasets [5]. For instance, Niranjan et al. developed a multi-phase chatbot combining question analysis, document processing, and answer extraction using RNNs [4]. While these methods achieved reasonable accuracy in intent recognition, they demanded large labeled datasets and lacked flexibility to incorporate new knowledge on the fly. Furthermore, managing complex free-text queries and generating fluent, trustworthy responses remain open research challenges [3].

Traditional LLMs depend solely on the information encoded during training with fixed cutoffs, whereas RAG models initially retrieve applicable content from a knowledge repository and generate responses informed by this real-time context. Retrieval-Augmented Generation (RAG) is a modern paradigm that enhances language models by supplying relevant context retrieved at runtime [9]. Unlike traditional LLMs which depend solely on their internal information (limited by pre-training cutoffs), RAG-based models initially retrieve relevant content from a knowledge base and then generate grounded responses using that context. This architecture improves factual consistency and reduces hallucination in generated answers [9].

RAG is particularly effective in open-domain question answering, where up-to-date context is essential. When applied to agriculture, RAG allows systems to incorporate the latest information from crop bulletins, research papers, and soil reports, providing more relevant responses to farmers' queries about diseases, cultivation practices, or environmental conditions [10].

Recent work such as *Farmer.Chat* (2024) has adopted this architecture. The system integrated both structured crop data and unstructured literature (research papers, videos) into a multilingual retrieval pipeline, achieving a context-retrieval precision of 71% and faithfulness in 80% of outputs [1]. A survey of agricultural assistants based on RAG identified trustworthiness, bias reduction, and explainability as ongoing concerns, but concluded that RAG is the most scalable method for intelligent crop advisory [11].

Our work builds upon these insights by adopting a RAG-based chatbot pipeline, incorporating semantic retrieval and locally hosted LLMs. This hybrid architecture ensures that soybean-specific advisory is both accurate and grounded in actual field-tested recommendations.

3 System Design and Architecture

The overall system architecture consists of three major components:

1. **Data Ingestion and Preprocessing**
2. **Embedding and Vector Storage**
3. **Query/Answer Pipeline**

Figure 1 details the high-level architecture of the proposed system. Documents (PDFs, websites, databases) are collected and pre-processed into clean English-only text segregating the stop words and special characters. The text is then segmented into smaller units ("chunks") and encoded into high-dimensional semantic vectors (embeddings) that are stored in a vector database. When a user submits a query, the system converts the query into an embedding, performs a semantic search over the database to retrieve the most relevant chunks, and forwards the result along with the user query to a language model. The model generates a response that is rigorously informed by the relevant context retrieved during the process.

4 Data Collection and Preprocessing

To develop a robust and knowledge-rich chatbot for soybean advisory, we gathered domain-specific textual data from multiple credible sources.

- **Extension Bulletins and Guidelines:** Advisory PDFs from agricultural universities, research institutes, and government agencies such as ICAR. These documents contained information on soybean cultivation techniques, pest and disease management, irrigation scheduling, and harvesting methods.
- **Research Articles:** Open-access academic publications and technical reports related to soybean genetics, soil nutrition, agronomic practices, and climate resilience. Only English-language papers were selected for consistency.
- **Web Content:** Selected text from agricultural knowledge portals, FAQs from agribusiness companies, and online crop advisory websites. These sources enriched the database with practical farmer-oriented content.

4.1 Bilingual and Mixed-Language PDFs

Many PDF bulletins included bilingual content, primarily English and Hindi within the same document. This presented a significant challenge, as many paragraphs or captions included English numerals or scientific names embedded within Hindi sentences.

To ensure the chatbot could function with an English language model, we implemented a two-step language-filtering strategy.

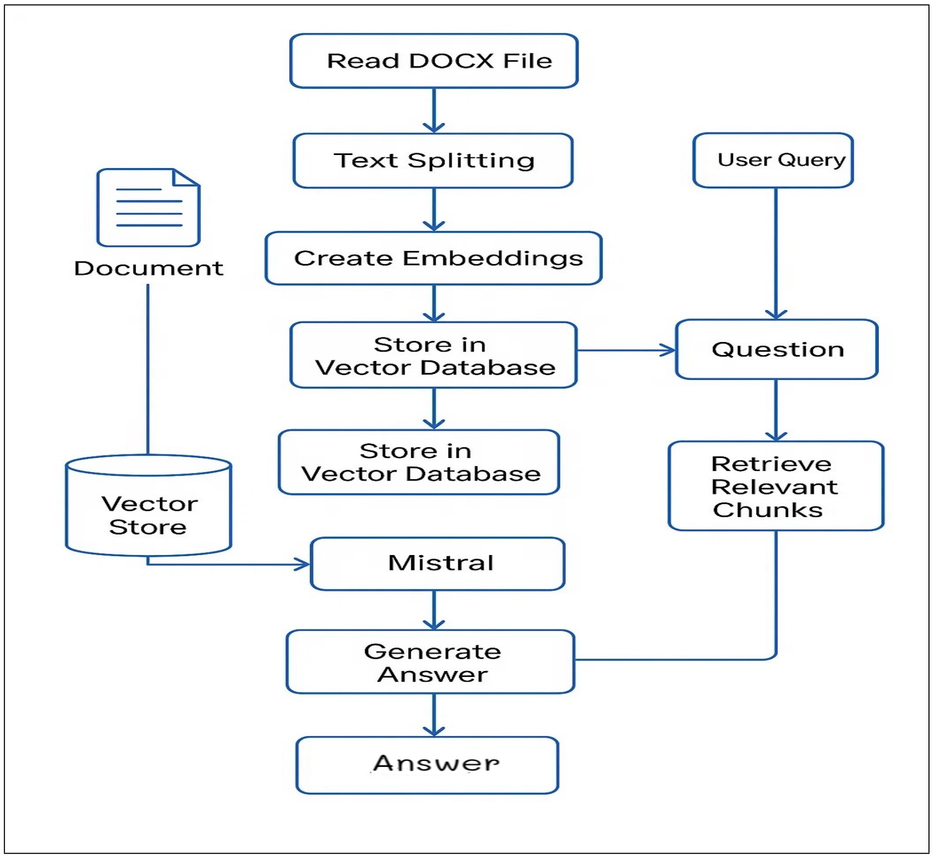

Fig. 1. High-level block diagram of the RAG-based chatbot pipeline.

1. **Automated Filtering:** We used a line-by-line language detection function (`langdetect`) to discard text not in English. Non- English dialects, or regional variations that differ significantly from standard English.
2. **Manual Proofing:** A sample of filtered outputs are manually reviewed to confirm that relevant advisory content is preserved and noise is eliminated.

4.2 PDF Parsing and Text Cleaning

Extracting structured text from PDF files is a complex task due to inconsistent layouts, embedded fonts, and non-standard encoding. We used **PyMuPDF (also known as MuPDF)** for parsing PDF pages into raw text. PDF files often break sentences across lines or columns, which disrupts semantic flow. For example, a sentence may continue onto the next line without proper punctuation or spacing. To resolve this we have checked whether a line ended without punctuation and whether the following line began with a lowercase word. If so, we merged the two lines with a space, treating them as part of the same sentence. This heuristic converts fragmented lines into readable, complete sentences that preserves sentence integrity.

Most bulletins included repeating headers (e.g., institute name, logo) and footers (e.g., page numbers, file names). We have identified them through recurring patterns across pages. These repetitive words are programmatically removed by matching fixed strings at the beginning or end of each page.

PDFs often contain special characters such as Bullet points (•), degrees (°), or non-breaking spaces and Symbols that interfere with tokenization or indexing. These are either manually replaced with standard ASCII equivalents or removed entirely to ensure clean and consistent data formatting.

Each document is converted into plain '.txt' format. We further removed the Licensing boilerplate text. The references and disclaimers are not useful for advisory. They are removed along with the empty lines or unusually short, irrelevant fragments.

The cleaned text corpus is too large to input into a language model at once. Hence, it is segmented into smaller **chunks** for efficient embedding and retrieval. We selected a target chunk length of **400–500 words** to strike a balance between contextual richness and computational efficiency. We used sentence-level segmentation to split content while preserving grammatical boundaries. A greedy algorithm iterated over the text, accumulating full sentences until the length threshold was reached. Each chunk was tagged with metadata, including the original source document and its index.

5 Embedding and Vector Storage

To enable semantic search over textual content, we transform each advisory chunk into a fixed-length numerical vector known as an **embedding**. These embeddings capture the semantic meaning of a sentence or passage and allow for similarity-based comparisons between queries and document content.

We utilized a pre-trained sentence embedding model accessible via `Ollama` or `HuggingFace Transformers`. These models generate high-dimensional vectors where texts that are semantically similar result in vectors close in vector space. The proximity of vectors is typically measured using **cosine similarity**.

Example: If a user query is *"soybean irrigation schedule"*, the embedding of that query should be close to embeddings of chunks discussing *"watering frequency"* or *"soil moisture management"*.

By representing both advisory content and user queries as vectors, we reduce the retrieval problem to a nearest-neighbor search in vector space. This method captures not just exact word matches but also synonyms, rephrasings, and concept-level similarity.

5.1 Leveraging Vector Database: ChromaDB

Once the embeddings are generated, they should be stored and indexed properly so that they can be quickly retrieved. This includes storage formats that enable efficient storage, and allows fast retrieval from large embedding datasets. We use **ChromaDB**, an open-source vector database designed for scalable and high-performance embedding search in AI applications.

Key Features of ChromaDB:

- Persistent storage of embeddings and associated metadata (e.g., document source, chunk ID).
- Support for cosine similarity and other nearest-neighbor algorithms.
- Easy-to-use Python interface for insertion, search, and filtering.

We created a collection named `SoybeanKB` in ChromaDB. For each text chunk:

1. The chunk is encoded into an embedding vector.
2. Metadata (source name, chunk number) is attached.
3. The vector and metadata are inserted into the collection using `collection.add()`.

The resulting vector index contains several thousand advisory chunks and supports semantic search in real time.

5.2 Similarity Search and Retrieval

At inference time, our chatbot system accepts a user query in natural language. Then it converts the query into an embedding using the same encoder as used during indexing. Finally, it sends this query embedding to ChromaDB.

ChromaDB then performs a **nearest-neighbor search** to retrieve the top-k chunks whose embeddings are closest to the query. We typically set $k = 5$ to 10 to balance response quality and brevity.

The retrieved chunks are concatenated to form the **context block** that is passed to the language model for response generation. This context grounding significantly boosts factual accuracy and domain relevance of the final answer.

6 Chatbot Development

We implemented the chatbot system using the **LangChain** framework, which enables the orchestration of multi-step language model pipelines. LangChain supports integration with vector databases, embedding models, and prompt engineering in a modular fashion.

The complete pipeline for the chatbot involves the following stages:

1. **User Query Handling:** The user submits a natural language query (in English) via the chat interface.
2. **Query Embedding:** The input query is converted into a vector embedding using the same model that was applied during the document indexing phase.
3. **Retrieval:** The query vector is used to perform a similarity search in ChromaDB, retrieving the top-k most relevant advisory chunks.
4. **Prompt Construction:** A prompt is formulated that includes the retrieved context and a user instruction. A sample prompt template is shown below:

   ```
    You are an agricultural expert for soybean crops. Based on the
   following information excerpts, answer the question in simple
   terms suitable for a farmer.
   Context: "[Insert retrieved chunks here]"
   Question: "[User's question]"
   Answer:
   ```
5. **LLM Generation:** This constructed prompt is provided to the local language model (e.g., Vicuna-13B via Ollama) which produces a response based on the given context.
6. **Response Postprocessing:** The raw model output is processed to remove artifacts, redundant phrases, or formatting issues. This ensures the final response is clean and user-friendly.

To ensure accessibility, we tuned the prompts to encourage clarity, brevity, and layman's language. The instruction *"in simple terms suitable for a farmer"* consistently guided the model toward producing farmer-friendly advice.

6.1 Prompt Tuning and Safety

To ensure responsible and useful responses, we included safety instructions and behavioral constraints directly within the prompts. This included directives such as:

- Avoid speculation or hallucination—only answer using the provided context.
- If no relevant information is available, say *I don't know.*
- Be succinct, practical, and clear in answering.

For both embeddings and generation, we used a locally hosted open-source LLM via **Ollama**. This provided the following advantages:

- Fully offline capability.
- Cost efficiency without relying on external APIs.
- Integration with LangChain via `OllamaEmbeddings` and `ChatOllama` modules.

We also implemented context truncation to manage token length limitations. When the retrieved chunks exceeded the model's context window, the least relevant chunks are dropped.

7 Sample Queries and Responses

This section presents actual examples of interactions with the chatbot, demonstrating its capability to provide helpful and understandable answers grounded in advisory documents.

Example 1

User: "What is the best time to plant soybean in my region?"

Chatbot: Soybean is usually planted at the beginning of the rainy season. In most areas, this means sowing the seeds just as the rains start regularly, so the young plants get enough water. Planting too early when it's dry or too late when heat is high can reduce yield. Follow local extension guidelines: for example, if rains come in June, plant in early June.

Example 2

User: "How do I control soybean aphids without harming the crop?"

Chatbot: The soybean aphid is a common pest on young plants. A safe method is to use insecticides labeled for soy, applied when aphids are first noticed. Be sure to use the recommended dose and apply in early morning or evening to protect beneficial insects. Alternatively, planting resistant soybean varieties or encouraging ladybugs (a natural predator) can help. Always read the label or ask an expert before applying chemicals.

Example 3

User: "Can soybean grow in alkaline soil?"

Chatbot: Soybean prefers neutral to slightly acidic soils, but it can tolerate mildly alkaline soil. If your soil is alkaline (pH above 7), you should add soil amendments before planting. For example, mixing sulfur or well-rotted manure into the soil can lower pH over time. Make sure to take a soil test first and follow recommended adjustments for soybean.

Example 4

User: "Why are some soybean leaves turning yellow?"

Chatbot: Yellow leaves in soybean can be caused by several factors. A common reason is lack of nutrients, especially nitrogen or iron, causing chlorosis. Another cause could be disease like soybean mosaic virus. Check if the plant has a nutrient deficiency: applying a balanced fertilizer can help. If it looks like spots or veins are affected, consult a local expert for a possible disease treatment.

Example 5

User: "How much fertilizer does soybean need?"

Chatbot: Soybean needs nutrients like phosphorus and potassium, but it usually fixes its own nitrogen. A soil test can tell you what's missing. Generally, you might apply a moderate amount of phosphorus (for root growth) and potassium. For example, about 20–30 kg per hectare of P and K each is common, but follow local recommendations. Overfertilizing can harm the crop or the environment.

8 Results and Evaluation

To evaluate the performance of our chatbot system, we tested it against a set of representative soybean-related questions sourced from farming forums, agricultural FAQs, and extension documents. Each question was designed to reflect real-world information needs, such as crop schedules, pest control, or soil requirements.

For each test query, we manually examined whether the retrieved chunks from ChromaDB were relevant to the topic. For example, a query on *"optimal soybean planting date"* successfully retrieved advisories and bulletins containing sowing time guidance, which the language model then summarized appropriately.

We adopted a retrieval evaluation metric called **context precision**, defined as the fraction of retrieved content that directly contributed to answering the user's query. Inspired by evaluation methods used in prior RAG systems [11], we:

- Manually labeled retrieved passages as "relevant" or "irrelevant" to the query.
- Computed precision as the number of relevant chunks divided by the total number retrieved.

The system achieved an average **context precision of 71%**, with most unrelated content involving tangential topics such as general legume cultivation. Fortunately, the prompt's instructions often guided the language model to ignore off-topic chunks during answer generation.

Evaluating chatbot responses is more subjective than evaluating retrieval. We combined informal human judgment with qualitative criteria, as automatic metrics like BLEU or ROUGE do not correlate well with answer helpfulness in open-ended tasks.

We assessed 50 randomly selected responses using the following attributes:

- **Accuracy:** Was the answer factually correct and consistent with retrieved context?
- **Clarity:** Was the answer easy to read, avoiding technical jargon?
- **Relevance:** Did the answer directly address the user's question?

Findings:

- **78%** of answers were *accurate*.
- **67%** were rated as *highly relevant* to the original query.
- Most answers were phrased clearly; when not, postprocessing and prompt tuning helped improve clarity.

To evaluate user experience, we conducted a small-scale usability test with the Agricultural students (who simulated farmers' needs). We also take the help of a few soybean growers in a local extension community for this task.

Participants interacted with the chatbot and rated its helpfulness. Key feedback included:

- The **conversational interface was intuitive** and required minimal training.
- Farmers appreciated the chatbot's tendency to give **simple and actionable advice**.
- Some requested **multilingual or voice-based support** to improve accessibility for non-literate users.

A notable observation was the difficulty in interpreting retrieved advisory chunks in technical language. This was resolved by ensuring that the LLM was instructed to paraphrase using layman's terms.

Overall, the feedback suggested that the chatbot could reduce farmers' reliance on periodic in-person consultations with extension agents, especially for routine issues.

9 System Limitations

While the system performed reliably, several limitations were observed:

- **Hallucinations:** The LLM occasionally introduced fabricated details when retrieval yielded weak context. This was mitigated by instructing the model to acknowledge uncertainty (e.g., "I don't know").
- **Ambiguous Queries:** Some queries, like *"Is it time to harvest?"*, lacked sufficient context. The model responded with general maturity indicators but could not tailor advice without user clarification.
- **Context Length Limitations:** When many relevant chunks were retrieved, they occasionally exceeded the model's input limit. We resolved this by ranking and truncating the least relevant portions.

These limitations highlight that while retrieval-augmented generation improves factuality, user judgment and occasional expert consultation are still necessary for critical decisions.

10 Conclusion

This paper work has demonstrated the feasibility and effectiveness of using a Retrieval-Augmented Generation (RAG)-based chatbot system for delivering soybean crop advisory. By combining semantic search with a large language model (LLM), we built an intelligent assistant that retrieves relevant information from agricultural bulletins and generates context-aware responses tailored to farmers' queries.

Many farmers are more comfortable interacting in regional languages. We plan to support Hindi and other languages in future either by using translated documents or multilingual embedding models to index and retrieve content appropriately. To assist users with low literacy, we aim to incorporate voice-based interaction. This includes voice input (speech-to-text) and audio output (text-to-speech) to make the chatbot more inclusive.

References

1. N. Singh et al., "Farmer.Chat: Scaling AI-Powered Agricultural Services for Smallholder Farmers," in Proc. [Conference], Jun. 2024. (Available as arXiv:2409.08916)
2. D. J. Samuel et al., "AgroLLM: Connecting Farmers and Agricultural Practices through Large Language Models for Enhanced Knowledge Transfer and Practical Application," arXiv:2503.04788, Mar. 2025.
3. P. Rana et al., "Integration of ICT in Agricultural Extension Services: A Review," [Journal], 2025.
4. P. Y. Niranjan et al., "A Survey on Chat-Bot System for Agriculture Domain," in *Proc. 2019 1st Int. Conf. Advances in Information Technology (ICAIT)*, 2019, pp. 99–103.
5. A. Jomole and D. Dana, "Chatbot-Based Agricultural Extension Service Model Using Deep Learning," M.S. thesis, Wolaita Sodo Univ., Ethiopia, Jun. 2024.
6. D. K. Agarwal et al., "Soybean: Introduction, Improvement, and Utilization in India—Problems and Prospects," *Agric. Res.*, vol. 2, no. 4, pp. 293–300, Dec. 2013.
7. K. Jadhav and A. Dhamal, "Agriculture Chatbot Marathi," *Int. J. of Research Publication and Reviews*, vol. 6, no. 4, pp. 343–349, Apr. 2025.
8. A. Vizniuk et al., "A Comprehensive Survey of Retrieval-Augmented Large Language Models for Decision Making in Agriculture: Unsolved Problems and Research Opportunities," *J. Artif. Intell. Soft Comput. Res.*, vol. 15, no. 2, Mar. 2025.
9. Y. Huang et al., "A Survey on Knowledge-Oriented Retrieval-Augmented Generation," arXiv:2503.10677, Mar. 2025.
10. P. Krampah, "Introduction to ChromaDB — Vector Store for Generative AI LLMs," Medium, Sep. 2023.
11. A. Venkataswamy, "PDF Hell and Practical RAG Applications," Unstract Blog, May 2024.