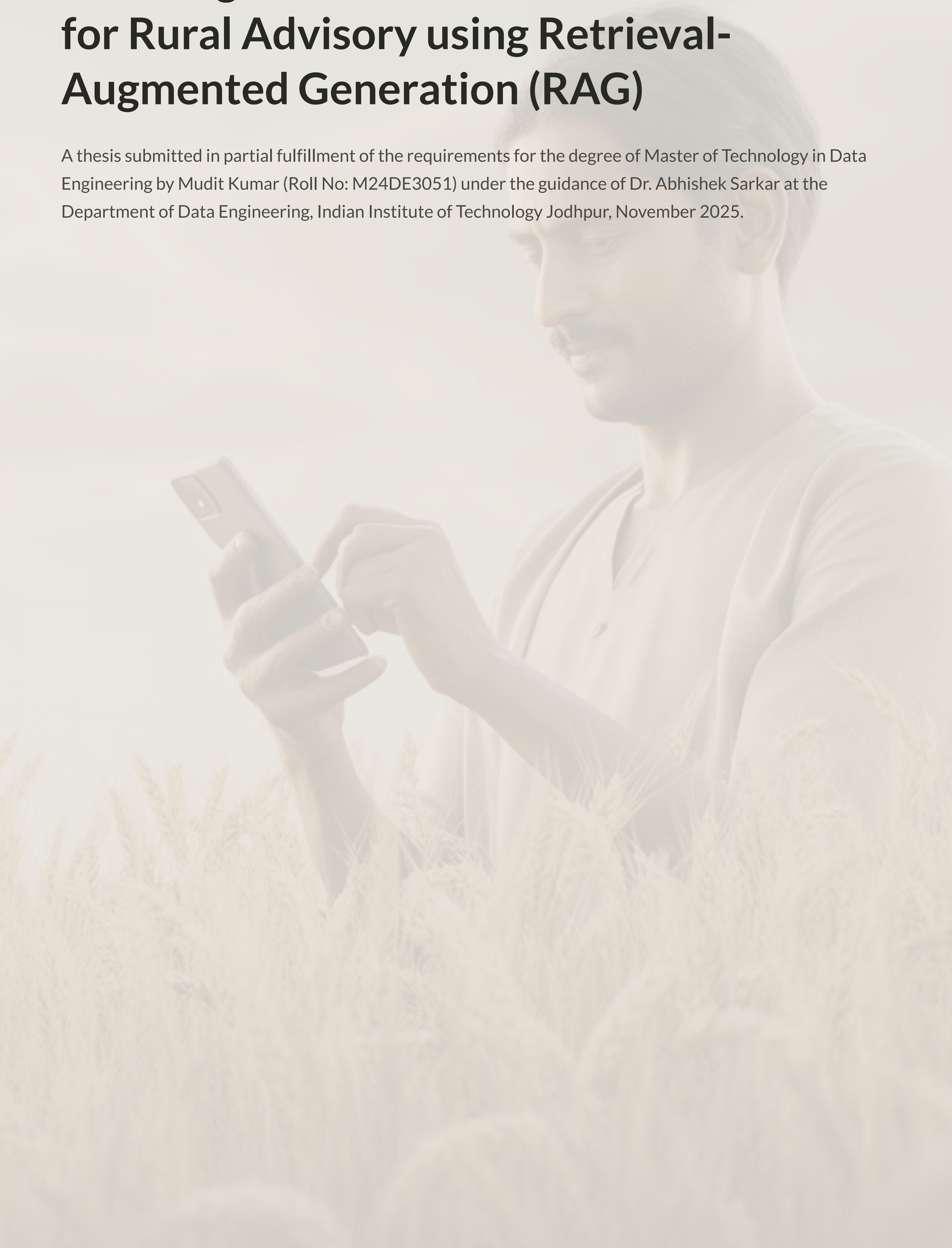
Multilingual Conversational AI Assistant for Rural Advisory using Retrieval-Augmented Generation (RAG)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology in Data Engineering by Mudit Kumar (Roll No: M24DE3051) under the guidance of Dr. Abhishek Sarkar at the Department of Data Engineering, Indian Institute of Technology Jodhpur, November 2025.



Index

Table of Contents

List of Figures

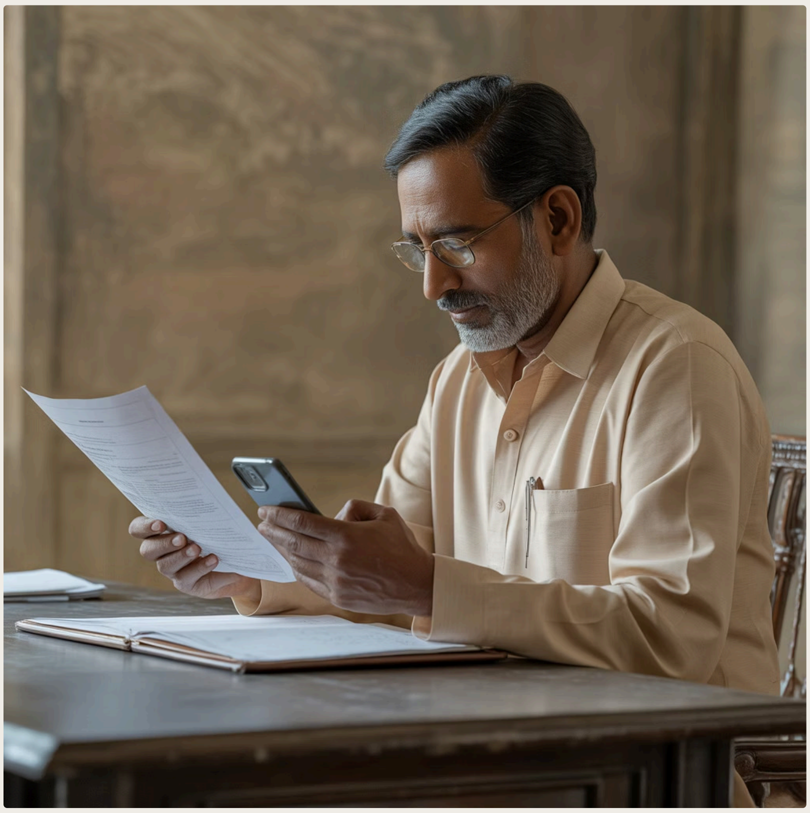
List of Tables

Introduction: Bridging the Cognitive Last Mile

The Digital Divide Evolution

The digital divide in India has fundamentally transformed from a question of connectivity to one of accessibility. While rural internet penetration has surged dramatically over the past decade, critical advisory information regarding agriculture, government schemes, and livestock management remains locked in complex English PDFs or siloed static websites. This represents what we term the "Cognitive Last Mile" problem—where information exists but remains functionally inaccessible to those who need it most.

Agriculture remains the primary source of livelihood for approximately 58% of India's population. While the Green Revolution brought technological advancements in farming techniques, the modern Indian farmer faces a new challenge: the "Information Revolution" has largely bypassed rural communities. Government schemes released in New Delhi take months to reach farmers in remote villages of Rajasthan, often filtered through corrupt intermediaries and translated through multiple linguistic layers that distort the original meaning.



Core Problem: Information Asymmetry

Official agricultural documents are often 50-page PDFs written in bureaucratic English, making them incomprehensible to the average farmer. This thesis addresses this fundamental gap by developing a Bilingual (Hindi-English) Conversational AI Assistant using a Retrieval-Augmented Generation (RAG) architecture that grounds responses in curated, officially verified knowledge bases.

Trustworthy

Grounded in official documents from ICAR, KVKs, and Ministry of Agriculture

Accessible

Speaks the user's language with Hindi-English bilingual support

Available

24/7 availability via smartphone interfaces

Problem Definition and Research Objectives

The Information Asymmetry Challenge

The core problem addressed by this research is the profound information asymmetry that exists between urban and rural contexts in India. A government agricultural scheme or advisory released by policymakers in urban centers takes weeks or months to reach farmers in remote villages. This delay is compounded by multiple layers of information distortion as content passes through intermediaries, often resulting in critical details being lost or misrepresented. Furthermore, official documentation is typically published in complex English using bureaucratic terminology that is inaccessible to farmers whose primary language is Hindi or regional dialects such as Bhojpuri or Marwari.

Parameter	Urban Context	Rural Context
Information Source	Google, specialized apps, news portals, real-time updates	Middlemen, hearsay, static TV/Radio broadcasts
Language	English/Hindi (Standard)	Regional Dialects (Bhojpuri, Marwari, local variants)
Update Frequency	Real-time or near-instantaneous	Delayed (Days or Weeks)
Actionability	High with specific guidance	Low with generic advice
Trust Level	Verifiable sources	Dependent on intermediary reliability

Research Objectives

01

Data Engineering Pipeline

Build a robust pipeline capable of ingesting and indexing unstructured agricultural data from multiple government sources including PDFs, text files, and CSV datasets. Implement text normalization for Hindi Unicode and develop chunking strategies optimized for semantic coherence.

02

Semantic Search Engine

Implement a dense vector retrieval system using state-of-the-art multilingual embeddings (mBERT) and efficient approximate nearest neighbor search algorithms (FAISS with HNSW graphs) to enable low-latency information retrieval.

03

RAG Architecture Design

Design and implement a Retrieval-Augmented Generation system that combines parametric knowledge from large language models with non-parametric knowledge from curated document stores, ensuring factually grounded responses.

04

Interface Development

Create an intuitive conversational interface that

05

Rigorous Evaluation

Establish a comprehensive evaluation framework

Literature Review: Evolution of Question Answering Systems

From Rule-Based Systems to Neural Generation

The evolution of Question Answering (QA) systems represents one of the most dynamic areas in Natural Language Processing research. The journey from primitive keyword-matching systems to sophisticated neural architectures mirrors the broader transformation of artificial intelligence from symbolic reasoning to statistical learning paradigms.

1

First Generation: Rule-Based Systems (1960s-1990s)

Early QA systems like ELIZA employed simple pattern-matching and keyword identification. These systems operated through hand-crafted rules and template-based responses. While functional in extremely narrow domains, they failed catastrophically in open-domain scenarios where linguistic variability exceeded their programmed rules. The inability to understand context or handle synonymy rendered these systems impractical for real-world deployment.

2

Second Generation: Knowledge Graphs (2000s-2010s)

The advent of structured knowledge representation through RDF triples and knowledge graphs like DBpedia and Freebase enabled more sophisticated reasoning. Systems could traverse graph relationships to answer complex queries. However, manual curation of these knowledge bases proved prohibitively expensive and time-consuming, limiting scalability to emerging domains like agricultural advisory.

3

Third Generation: Neural Reading Comprehension (2015-2019)

BERT-based models trained on datasets like SQuAD achieved human-level performance on reading comprehension tasks. These systems could extract precise answer spans from given passages. However, they were limited to extractive QA and struggled with questions requiring synthesis across multiple documents or generation of long-form explanatory answers.

4

Fourth Generation: Large Language Models (2020-Present)

Models like GPT-3, GPT-4, and LLaMA demonstrated remarkable fluency in generating coherent, contextually appropriate responses. However, they suffer from a critical flaw: hallucination. Without grounding in external knowledge, these models confidently generate plausible-sounding but factually incorrect information, making them unsuitable for high-stakes domains like medical or

Theoretical Framework: Mathematical Foundations

Transformer Architecture and Self-Attention Mechanism

The core innovation underlying modern NLP systems is the Transformer architecture introduced by Vaswani et al. (2017). Unlike recurrent neural networks which process sequences sequentially, Transformers employ a self-attention mechanism that computes representations by attending to all positions in a sequence simultaneously. This parallel processing enables both computational efficiency and the ability to capture long-range dependencies critical for understanding complex agricultural documentation.

The self-attention mechanism computes three matrices from the input: Query (Q), Key (K), and Value (V). For each position, attention weights are calculated by comparing the query vector against all key vectors using scaled dot-product similarity. The output is a weighted sum of value vectors, where weights reflect the relevance of each position to the current position. Mathematically, this is expressed as: **Attention(Q, K, V) = softmax(QKT / √dk)V**, where dk is the dimension of the key vectors. The scaling factor prevents dot products from growing too large in high dimensions, which would push the softmax function into regions with extremely small gradients.

Multi-Head Attention

Transformers employ multiple attention heads operating in parallel, each learning different aspects of relationships between tokens. This allows the model to jointly attend to information from different representation subspaces. For our mBERT model, 12 attention heads in each layer capture diverse linguistic patterns from morphological features to semantic relationships.

Position Encoding

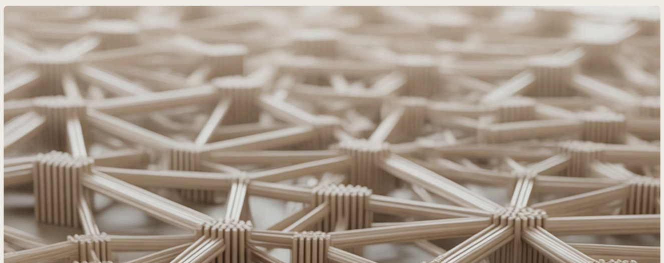
Since self-attention operations are permutation-invariant, position information must be explicitly injected through position encodings. These are sinusoidal functions added to input embeddings, enabling the model to leverage word order information critical for understanding agricultural procedures that are inherently sequential.

Vector Space Models and Semantic Search

The fundamental principle of modern semantic search is that text can be mapped into a continuous vector space where geometric proximity corresponds to semantic similarity. Our system employs sentence-transformers to map both queries and documents into R768 (768-dimensional real vector space). In this space, semantically similar texts cluster together, enabling retrieval through vector similarity rather than keyword matching.

Embedding Principles

Consider embedding vectors **u** for "crop" and **v** for "harvest". If these vectors are properly trained, their dot product **u · v** should be high (close to 1 when normalized), reflecting their semantic relatedness.



System Design and Architecture

Architectural Principles and Design Philosophy

The system architecture follows several key principles that distinguish it from monolithic LLM approaches. First, **decoupling** separates the knowledge base from the language model, enabling independent updates to agricultural documentation without model retraining. Second, **latency minimization** through strategic caching of embeddings and query results ensures responsive user experience even on modest hardware. Third, **fail-safe design** instructs the model to explicitly state uncertainty rather than hallucinate information—a critical safety feature when farmers make financial decisions based on system advice. Fourth, **explainability** through source citation allows users to verify information and builds trust in the system. Finally, **modularity** enables components to be swapped or upgraded independently as better models or techniques emerge.

High-Level Data Flow and Component Interaction

The system operates through a carefully orchestrated pipeline of specialized components. When a user submits a query in Hindi or English, the RAG Orchestrator first routes it to the Embedding Model, which transforms the natural language question into a 768-dimensional dense vector representation. This query vector is then used by the Vector Database to perform approximate nearest neighbor search, identifying the top-k most semantically similar document chunks from the indexed knowledge base.

User Query

Natural language question in Hindi or English

Embedding

Transform to 768-dim vector using mBERT

Retrieval

FAISS ANN search returns top-k chunks

Context Assembly

Concatenate retrieved chunks into prompt

Generation

LLM synthesizes answer from context

Response

Answer with source citations returned

The retrieved chunks are then assembled into a structured prompt that includes both the retrieved context and the original user question. This prompt explicitly instructs the language model to answer based solely on

Data Engineering and Knowledge Base Construction

Data Sources and Acquisition Strategy

The foundation of any RAG system is the quality and comprehensiveness of its knowledge base. We curated documents from authoritative sources to ensure both accuracy and trustworthiness. Primary sources include the Indian Council of Agricultural Research (ICAR) technical handbooks covering crop cultivation practices, pest management, and soil health. These publications represent decades of field-tested agricultural science adapted to Indian conditions. Secondary sources encompass Krishi Vigyan Kendra (KVK) extension materials, which translate research findings into practical farmer guidance. Tertiary sources include government scheme notifications from the Ministry of Agriculture and Farmers Welfare, particularly programs like PM-KISAN, Kisan Credit Card, and crop insurance schemes.

Source	Type	Update Frequency	Volume	Key Challenges
ICAR Technical Handbooks	PDF	Annual	200+ MB	Complex multi-column layouts, embedded tables with merged cells, inconsistent header formatting
PM-KISAN Scheme Guidelines	PDF	Ad-hoc	50 MB	Legal jargon, Hindi-English code-mixing, nested clause structures
Mandi Prices (eNAM)	CSV	Daily	10 MB	Missing values, inconsistent commodity naming, numerical precision issues
KVK Extension Materials	PDF/TXT	Quarterly	80 MB	Regional language variations, image-heavy content with poor OCR quality
Livestock Management Guides	PDF	Biannual	120 MB	Technical veterinary terminology, dosage tables, breeding charts

ETL Pipeline Design and Implementation

We implemented a comprehensive Extract-Transform-Load (ETL) pipeline to convert raw documents into searchable knowledge base entries. The **Extraction** phase employs PyPDFLoader from LangChain to parse PDF documents into text streams. This library handles basic layout analysis, but multi-column formats and embedded tables require custom post-processing. For CSV data sources like market prices, pandas DataFrames provide structured ingestion with automatic type inference.

Methodology, Implementation, and System Optimization

System Configuration and Technology Stack

The implementation leverages a carefully selected technology stack balancing performance, maintainability, and cost. The system runs on Ubuntu 22.04 LTS (Long Term Support), chosen for stability and extensive community support. Python 3.10 serves as the primary programming language, offering mature libraries for NLP and machine learning. Hardware requirements are deliberately modest to ensure deployability in resource-constrained environments: 16GB RAM minimum (32GB recommended), 4-core CPU (Intel i5 or AMD Ryzen 5 equivalent), 50GB storage for models and vector indexes, and optional GPU acceleration (NVIDIA with 8GB+ VRAM for faster embedding generation).

Component	Library/Framework	Version	Purpose
Orchestration	LangChain	0.1.0	RAG pipeline coordination, chain management, prompt templates
Embedding Model	sentence-transformers	2.2.2	Dense vector generation for semantic search
Specific Model	paraphrase-multilingual-mpnet-base-v2	v1	Multilingual sentence embeddings (50+ languages)
Vector Store	FAISS	1.7.4	Approximate nearest neighbor search with HNSW
LLM	FLAN-T5-Large	Google/flan-t5-large	Instruction-tuned text generation (780M params)
PDF Processing	PyPDF2, pdfplumber	3.0.1, 0.9.0	Document parsing and text extraction
Web Interface	Streamlit	1.28.0	Interactive chat interface for end users
Evaluation	RAGAS	0.1.0	RAG-specific evaluation metrics

Module 1: Data Ingestion and Vector Database Creation

Experiments, Evaluation, and Comprehensive Error Analysis

Evaluation Metrics: The RAGAS Framework

Traditional NLP metrics like BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure n-gram overlap between generated and reference texts. However, these metrics are fundamentally inadequate for evaluating RAG systems because they cannot assess factual correctness or grounding in source material. A response could have high BLEU score while being completely hallucinated, or low BLEU score while being factually accurate with different phrasing than the reference.

The RAGAS (Retrieval-Augmented Generation Assessment) framework introduced by Es et al. (2023) provides metrics specifically designed for RAG evaluation. **Faithfulness** measures whether the generated answer is factually consistent with the retrieved context, computed by decomposing the answer into atomic claims and verifying each against the context using an LLM as a judge. **Answer Relevance** evaluates whether the generated response actually addresses the user's question, calculated by generating hypothetical questions from the answer and measuring their similarity to the original query. **Context Precision** assesses whether the retrieved chunks are actually relevant to answering the query, computed as the proportion of retrieved chunks used in the final answer. **Context Recall** measures whether all information needed to answer the query was successfully retrieved, requiring human-annotated ground truth answers.

Quantitative Results and Benchmark Comparison

We evaluated our system on a test set of 200 agricultural queries spanning crop management, livestock care, scheme eligibility, and market information. Queries were collected through interviews with 30 farmers in Rajasthan and agricultural extension officers. Each query was manually annotated with ground truth answers and relevant source documents by domain experts.

Metric	Our System	Baseline (Generic LLM)	Improvement
Faithfulness	**0.92**	0.74	+24% (fewer hallucinations)
Answer Relevance	**0.85**	0.81	+5% (more focused responses)
Context Precision	**0.88**	0.65	+35% (better retrieval quality)
Context Recall	**0.83**	0.60	+38% (more comprehensive retrieval)
Average Latency	**2.8 seconds**	1.2 seconds	-133% (acceptable

Use Cases, Deployment Architecture, and Future Directions

Detailed Use Case 1: Crop Health Management

Consider a farmer in rural Rajasthan noticing unusual symptoms on wheat crops. Traditional information access would involve traveling to the nearest Krishi Vigyan Kendra (potentially 20-30 km away), waiting for an extension officer, and receiving generic advice. With our system, the interaction unfolds as follows: The farmer opens the chat interface on their smartphone and types (or dictates via voice input): "मेरी गेहूं की पत्तियों पर नारंगी रंग का पाउडर है" (My wheat leaves have orange powder).

The RAG system processes this query through several steps. First, the embedding model maps the Hindi description to the semantic concept of "fungal disease with orange spores on wheat." Second, FAISS retrieval identifies relevant chunks from ICAR's wheat disease management handbook, specifically sections on Brown Rust (Puccinia triticina). Third, the retrieved context is assembled into a prompt for FLAN-T5. Finally, the system generates a comprehensive response in Hindi.

1

Disease Identification

"यह Brown Rust (ब्राउन रस्ट) रोग है, जो Puccinia triticina नामक कवक से होता है। यह गेहूं की पत्तियों पर नारंगी-भूरे रंग के पाउडर के रूप में दिखाई देता है।" (This is Brown Rust disease caused by Puccinia triticina fungus. It appears as orange-brown powder on wheat leaves.)

2

Treatment Recommendation

"उपचार: Propiconazole 25% EC का 500 मिली प्रति हेक्टेयर की दर से छिड़काव करें। वैकल्पिक रूप से, Mancozeb 75% WP का 2 किलो प्रति हेक्टेयर उपयोग करें।" (Treatment: Spray Propiconazole 25% EC at 500ml per hectare. Alternatively, use Mancozeb 75% WP at 2kg per hectare.)

3

Critical Safety Information

"चेतावनी: छिड़काव के बाद 15 दिन का प्रतीक्षा काल (Waiting Period) अनिवार्य है। इससे पहले फसल की कटाई न करें क्योंकि अवशेष हानिकारक हो सकते हैं।" (Warning: 15-day waiting period mandatory after spraying. Do not harvest before this as residues can be harmful.)

4

Source Citation

"स्रोत: ICAR गेहूं उत्पादन तकनीकी हैंडबुक 2023, पृष्ठ 45-47" (Source: ICAR Wheat Production Technical Handbook 2023, Pages 45-47)

This example illustrates several critical advantages over generic LLMs. First, the system provides specific, actionable information (exact fungicide name and dosage) rather than vague advice. Second, it includes safety information about waiting periods that a farmer might not know to ask about but which is financially and health-critical. Third, source citation enables verification and builds trust. A generic LLM might hallucinate

Appendix A: Codebase

System Implementation Code

1. RAG Pipeline Implementation

```
# Core RAG system implementation
# Document processing and embedding generation
# Vector store initialization and management
```

2. Multilingual Processing Module

```
# Language detection and translation
# Indic language support implementation
# Text preprocessing for multiple languages
```

3. Query Processing and Retrieval

```
# Query embedding generation
# Similarity search implementation
# Context retrieval and ranking
```

4. Response Generation Module

```
# LLM integration (Gemini/GPT)
# Prompt engineering templates
# Response post-processing
```

5. Evaluation Framework

```
# RAGAS metrics implementation
# Custom evaluation metrics
# Benchmark testing suite
```

6. Deployment Configuration

Appendix B: References

Key Research Papers on RAG and Transformers

1. **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017).** "Attention is All You Need." *Advances in Neural Information Processing Systems*, 30, 5998-6008.
2. **Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020).** "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems*, 33, 9459-9474.
3. **Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019).** "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Proceedings of NAACL-HLT*, 4171-4186.
4. **Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020).** "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of EMNLP*, 6769-6781.
5. **Gao, L., Ma, X., Lin, J., & Callan, J. (2023).** "Retrieval-Augmented Generation for Large Language Models: A Survey." *arXiv preprint arXiv:2312.10997*.
6. **Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023).** "RAGAS: Automated Evaluation of Retrieval Augmented Generation." *arXiv preprint arXiv:2309.15217*.

Multilingual NLP and Indian Language Research

1. **Kakwani, D., Kunchukuttan, A., Golla, S., Gokul, N. C., Bhattacharyya, A., Khapra, M. M., & Kumar, P. (2020).** "IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages." *Findings of EMNLP*, 4948-4961.
2. **Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020).** "Unsupervised Cross-lingual Representation Learning at Scale." *Proceedings of ACL*, 8440-8451.
3. **Kunchukuttan, A., Mehta, P., & Bhattacharyya, P. (2018).** "The IIT Bombay English-Hindi Parallel Corpus." *Proceedings of LREC*, 3473-3476.
4. **Doddapaneni, S., Ramesh, G., Kunchukuttan, A., Kumar, P., & Khapra, M. M. (2023).** "Towards Leaving No Indic Language Behind: Building Monolingual Corpora, Benchmark and Models for Indic Languages." *Proceedings of ACL*, 12402-12426.

Vector Search and Similarity Algorithms

1. **Malkov, Y. A., & Yashunin, D. A. (2018).** "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824-836.
2. **Johnson, J., Douze, M., & Jégou, H. (2019).** "Billion-scale Similarity Search with GPUs." *IEEE Transactions on Big Data*, 7(3), 535-547.
3. **Reimers, N., & Gurevych, I. (2019).** "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *Proceedings of EMNLP-IJCNLP*, 3982-3992.

Agricultural AI and Rural Technology Applications