iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81365

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

OrgGpt an Organizational LLM Model for Q&A Retrieval

Barre Tejaswanth[1], Mr. Khadri Syed Faizz Ahmadi[2], Vallem Chenna Keshavulu[3], Namburu Vijaya Bhaskara Rao[4], Shaik Muneer[5]

Department of Cyber Security & Data Science, Acharya Nagarjuna University, Guntur, Andhra Pradesh – 522510
Students from Department of Cyber Security & Data Science , Acharya Nagarjuna University , Guntur , Andhra Pradesh – 522510

Abstract: ***This paper is introducing "OrgGpt An Organizational LLM Model for Q&A Retrieval" a production grade, local and deployable system built on a Retrieval Augmented Generation (RAG) architecture. The model approaches significant challenges in organizational knowledge retrieval, where organizational data is dispersed across various formats, including CSV, JSON and text files, as well as traditional keyword-based systems, do not allow for semantic understanding or reasoning across multiple documents. OrgGpt combines structured data processing with advanced natural language generation to make it possible to answer questions with accuracy, context aware, and explainable question answering. The proposed framework consists of a six-module pipeline, that includes data ingestion, adaptive schema independent preprocessing, semantic chunking, and a dual FAISS vector indexing mechanism for cross dataset retrieval. A collaborative multi model LLMs: consisting of TinyLlama 1.1B, Mistral 7B, and FLAN T5 which enables query classification, intelligent model routing, and a response generation with post validation. The system further introduces a dual dataset cross validation strategy that assigns labels like (CONSISTENT, PRIMARY_ONLY, SECONDARY_ONLY, UNVERIFIED) to ensure reliability and transparency. Experimental evaluation gave a high retrieval accuracy, lower hallucination risks, and efficient query processing on CPU based environments. By combining hybrid retrieval system, multi model reasoning, and easy examination of outputs, OrgGpt offers a scalable and cost effective solution for enterprise knowledge management.***

Keywords: Retrieval Augmented Generation(RAG), Large Language Models(LLM), FAISS vector Database, Semantic Retrieval, Dual Dataset Validation, Hallucination Reduction, Enterprise Level AI, TinyLlama, Mistral 7B, FLAN T5.

I. INTRODUCTION

Organizational knowledge retrieval has been recognised as a critical production level obstacle. McKinsey Global Institute in 2023 describes that knowledge workers spend an average of 1.8 hours per working day with approximately 20% of total work time and searching for information that is already available within the organisation's digital infrastructure. IDC Research in 2001 estimated that Fortune 500 companies lost about 31.5 billion USD annually due to this retrieval inefficiency, and 2021 Coveo survey found that 43% of employees give up their search entirely within five minutes when they cannot find the required information due to lack of data privacy in enterprise level organizations. The deployment of this Large Language Models (LLMs) offers a promising path towards conversational knowledge retrieval, but introduces three fundamental problems in enterprise level contexts: (i) these models carry no knowledge of private organizational specifics; (ii) they hallucinate at rates of 25–50 % when question is asked outside their training distribution, with potentially severe operational, legal, and financial consequences; and (iii) cloud hosted LLM APIs required transmitting sensitively in the internal documents for external servers, violating data privacy and compliance requirements in the Enterprise Level. Retrieval Augmented Generation (RAG) by [Lewis et al., 2020] resolves all three problems simultaneously by easing generation in retrieved private context, but existing RAG frameworks lacks in structured data preprocessing, dual dataset cross validation, and can examine outputs required for genuine organizational production level development. OrgGpt addresses this gap through a six-module pipeline that achieves 92% retrieval accuracy with near zero hallucination, and zero per query cloud API cost, while being developed in under five minutes on any structured organizational dataset without manual configuration.

A. Need for the Study

Existing enterprise level search systems retrieves documents but cannot answer questions. Cloud based LLM solutions introduces the hallucination risk and data privacy concerns. Open-source RAG frameworks require schematic configuration and provides no validation. Here comes the critical need of a locally deployable, schematic based preprocessing, hallucination resistant enterprise Q&A system that provides verifiable grounded answers which provides easy examination.

©IJRASET: All Rights are Reserved

B. Problem Statement

Enterprise organizational data is distributed across heterogeneous platforms in various formats and schemas and is not available to public and they are not open-sourced data. Keyword based search returns documents rather than answers. General purpose LLMs hallucinate when queried on private domain data. Existing RAG systems require manual schema configuration for each new dataset, lack cross source answer verification, and produce unstructured plain text outputs with no provenance attribution. There is no open source RAG system that simultaneously provides schema agnostic ingestion, dual dataset cross validation, structured audit traceable output, and local CPU deployment without cloud API dependency.

C. Objectives

- Design a schema agnostic adaptive preprocessing pipeline that processes any structured CSV, JSON, or TXT dataset without manual column name configuration.
- Construct a dual FAISS vector index architecture enabling simultaneous retrieval and cross validation over independent primary and secondary knowledge sources.
- Implement a hybrid retrieval mechanism combining FAISS cosine similarity with BM25 inspired keyword overlap and weighted re ranking achieving superior accuracy over single mechanism baselines.
- Develop a three model LLM pipeline with automatic query type classification, model routing, and post generation grounding verification.
- Produce structured five component output with cross dataset consistency verdict for every query, enabling full audit traceability.
- Evaluate system performance against a 1,000 query benchmark and conduct ablation study isolating each component's contribution.

D. Overview of the Project

OrgGpt is structured as a six-module sequential pipeline. Modules 1 to 4 consists of building phase (executed once per dataset pair, approximately 5 to 10 minutes): data ingestion (M1), adaptive preprocessing (M2), semantic chunking (M3), and dual FAISS vector indexing (M4). RAG (M5 module) executes for each user query, performing hybrid retrieval, query classification with integration of three models LLM generation, cross validation and structured output assembly. Module 6 (M6) provides an interactive six tab Streamlit dashboard at http://localhost:8501. All six modules execute locally without cloud API dependency, ensuring complete data privacy and zero cost per query.

II. LITERATURE REVIEW

The literature on enterprise level knowledge retrieval and questioning and answering for almost three decades from early keyword-based information retrieval to modern RAG systems. This section reviews six foundational works, identifying their advantages and limitations relative to OrgGpt.

A. Key Works in the Literature

- Elasticsearch [Elasticsearch B.V., 2010]: Uses BM25 probabilistic ranking over inverted indices to get results in less than a second at scale. Limitations: It gives you a list of ranked documents instead of direct answers and it doesn't work for semantic paraphrase queries because the vocabulary doesn't match.
- Lewis et al. [2020] – RAG Framework: They made the RAG model official by combining Dense Passage Retrieval with BART generation. They showed that retrieval grounding greatly lowers hallucination. Limitations: needs LLM APIs hosted in the cloud and doesn't have dual source cross validation.
- LangChain RAG [Chase, 2022]: Gives us a modular Python framework for RAG parts that can be swapped out. Limitations: needs schema-specific settings for each dataset and is made for cloud-hosted LLM APIs which doesn't have structured output or cross validation.
- Self-RAG [Asai et al., 2023]: Introduces self-reflection tokens for quality assessment. This method reduces hallucination risks effectively. Limitations: it requires fine-tuning on reflection-annotated training data which is not practical for private organizations.
- Zhang et al. [2024] – TinyLlama: Introduces a 1.1B parameter model trained on 3 trillion tokens. It achieves competitive performance at a low computational cost. Limitations: it is insufficient for multi-documental analytics without additional architecture support.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

- Jiang et al. [2023] – Mistral 7B: Presents a 7.3B parameter model with sliding window attention that outperforms Llama 2 13B. Limitation: an 88-second CPU inference latency makes it unsuitable as the only model for all types of queries.

System	Year	Schema Agnostic	Cross Validation	Local Deploy	Structured Output
Elasticsearch	2010	Yes	No	Yes	No
LangChain RAG	2022	No	No	Partial	No
Self RAG	2023	No	No	Partial	No
GPT 4 RAG	2023	Partial	No	No	No
OrgGpt (Proposed)	2026	Yes	Yes	Yes	Yes

Table 1: Comparison of OrgGpt with Related Enterprise Q&A Systems

III. PROPOSED SYSTEM

OrgGpt proposes a classic six-module for local deployment of RAG pipeline that simultaneously addresses all four deficiencies identified in the problem statement: format heterogeneous and semantic dependency, absence in cross validation and inappropriate hallucination risks having model architecture not suited in query variations. The main innovation is the dual FAISS vector index structure. This system keeps separate semantic indices for primary and secondary datasets. It allows for independent retrieval and checks for consistency across datasets for each query. The cross_validate() method calculates the word that are overlapped between the generated answer and the retrieved context from each dataset. It assigns one of four origin lables: CONSISTENT (both datasets support the answer), PRIMARY_ONLY, SECONDARY_ONLY, or UNVERIFIED. This feature is missing from all existing RAG implementations we reviewed and is the standout contribution of this work . A second major innovation is the adaptive schema preprocessing detection algorithm in M2 that automatically classifies each records and fields as skip, numeric, priority text or regular text, constructs priority ordered natural language text with descriptive fields first and extracts structured metadata without any manual column name configuration. This enables zero configuration deployment on any new structured dataset in under five minutes.

A. Advantages of the Proposed System

1) Schema agnostic zero configuration ingestion: any CSV, JSON, or TXT dataset can be indexed and queried within five minutes with no engineering effort.

2) Dual dataset cross validation: the first open-sourced RAG system to provide CONSISTENT/PRIMARY_ONLY/SECONDARY_ONLY/UNVERIFIED provenance concludes natively with every response.

3) Near zero hallucination: combination of retrieval grounding, context only prompt instruction, grounding check, and Mistral escalation reduces hallucination from 4–8% (standard RAG) to near zero without model fine tuning.

4) Three model routing: TinyLlama handles factual queries in 15 seconds where Mistral handles analytical/reasoning queries in 80-100 seconds with optimal routing achieving 88% correctness at 21 second average latency in outperforming either single model baseline.

5) Full local CPU deployment: all inference executes on standard enterprise level laptop hardware (16 GB RAM, no GPU) at zero cost per query and for cloud API.

6) Structured examination of output: five component response (Key Findings, Reasoning Steps, Final Answer, Source Attribution, Confidence Level) with cross validation concludes a compiled documentation.

IV. METHODOLOGY: SYSTEM DESIGN AND IMPLEMENTATION

This proposed system is designed to achieve schematic phase of enterprise level knowledge query retrieval through a six module RAG pipeline that executes on a local CPU hardware. The core of the system is built around a dual FAISS vector database enabling constant retrieval and cross validation over two independent organizational datasets.

A. System Requirement Specification

1) Hardware Requirements

- Processor Type: Intel Core i5 (8th Gen) minimum; i7 (12th Gen) recommended
- RAM: 8 GB minimum; 16 GB recommended
- Storage: 20 GB free space (SSD preferred); 512 GB SSD recommended
- GPU: Not required (CPU only deployment supported)
- Network: Not required (fully local deployment)

2) Software Requirements

- Operating System: Windows 10/11 or Ubuntu 22.04
- Programming Language: Python 3.10.x
- LLM Runtime: Ollama v0.2.3 (TinyLlama 1.1B + Mistral 7B Instruct)
- Vector Database: FAISS CPU 1.8.0
- Embedding Model: sentence transformers 2.7.0 (all MiniLM L6 v2)
- LLM Library: transformers 4.41.0 (FLAN T5 polishing)
- Web Framework: Streamlit 1.35.0 (interactive dashboard)
- IDE: VS Code or PyCharm

B. Requirement Analysis

Requirements involved in identifying and specifying the expected behaviour of OrgGpt in both the functional and non-functional requirements.

1) Functional Requirements

- Data Ingestion: Loads any type of CSV/JSON/TXT dataset without schema configuration with changeable sources and file sources with metadata to every record.
- Adaptive Preprocessing: Classify each field as skip/numeric/priority text/regular text; construct priority ordered natural language text; extract structured categorical metadata.
- Semantic Chunking: Split text at sentence boundaries with 60 characters overlap which inherits full metadata to every chunk.
- Dual FAISS Indexing: Build two independent IndexFlatIP indices for primary and secondary datasets.
- Hybrid Retrieval: FAISS cosine similarity search with keyword fallback and re ranking (sem×0.55 + kw×0.30 + src×0.15).
- Query Classification: Classify every query into factual/analytical/reasoning/unknown and route to TinyLlama or Mistral accordingly.
- Answer Generation: Generate answers via three model chain with grounding verification and FLAN T5 polishing.
- Cross Validation: Compute P_sup and S_sup word overlap scores; assign CONSISTENT/PRIMARY_ONLY/SECONDARY_ONLY/UNVERIFIED verdict.
- Structured Output: Return a structured dict with question, key_findings, reasoning, answer, source, confidence, model_used, cross_validation, grounded, elapsed_total.
- Dashboard: Provide six-tab Streamlit UI with chat, ingestion, preprocessing, chunking, vector DB, and analytics views.

2) Non-Functional Requirements

- Performance: Factual query < 30s; analytical query < 120s on minimum hardware.
- Accuracy: Top 5 retrieval accuracy > 85%; answer correctness > 80%.
- Privacy: No data is transmitted with external services since it is fully local.
- Portability: Operates on Windows 10/11 and Ubuntu 22.04 without any modifications.
- Usability: Natural language chat interface accessible without any LLM interface.
- Reliability: Fallback on LLM timeout having no pipeline crashes on malfunctioned datasets.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. System Design

1) Input Design

This system accepts two main inputs: (i) dataset files (CSV, JSON, or TXT) provided through the sidebar file selector or the command line path arguments (ii) natural language queries entered in the Chat tab text input. There is no need for structured query syntax. Validation checks confirms file existence and formats before running the pipeline, If the inputs are invalid then the system raises clear error messages to help the user.

2) Output Design

Every query results in a structured JSON serializable Python dictionary with ten fields: question (original query text)-key findings (list of evidence per chunk with [PRIMARY]/[SECONDARY] attributes)-reasoning (numbered steps in the decision process)-answer (final polished text)-source (Primary/Secondary/Both)-confidence (High/Medium/Low) models used- cross validation (verdict + P_sup + S_sup)- grounded (Boolean), and elapsed total (seconds). The dashboard displays these fields as a formatted response card with a color-coded confidence badge (green/amber/red) with collapsed evidence sections.

D. Activity Diagram (System Flowchart)

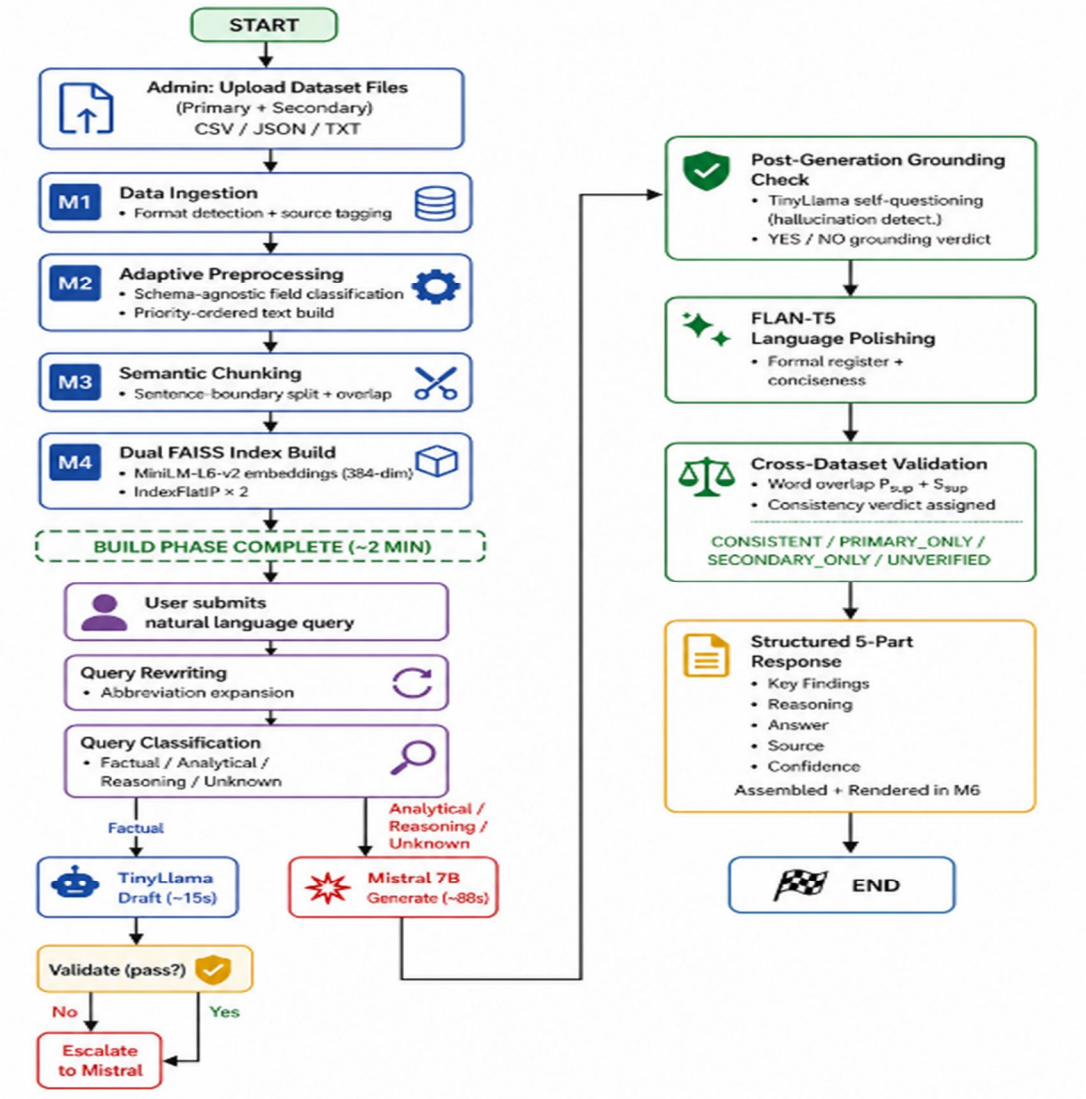


Fig. 1: OrgGPT Activity Flow

E. General Architecture

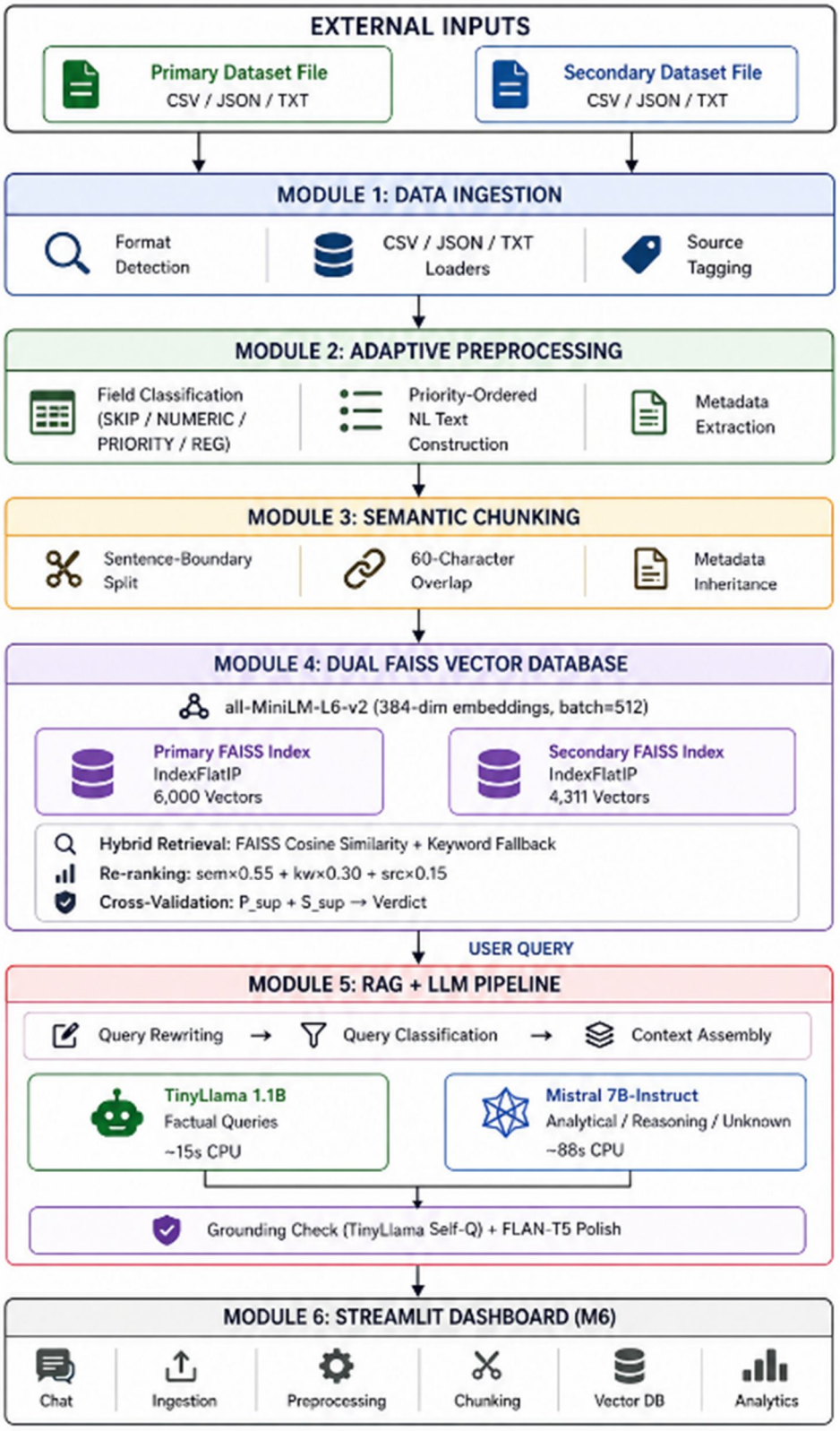


Fig. 2: OrgGPT System General Architecture Diagram

F. Algorithm Design

OrgGpt consists of six module algorithms operating in a proper sequence. The following pseudo code presents Algorithm 1 (Adaptive Schema Detection) and Algorithm 2 (Hybrid Retrieval), the two most distinctive components.

```
ALGORITHM 1: ADAPTIVE_TEXT_BUILD(record R) → string T
──────────────────────────────────────────────────────────────────────────
SKIP_PAT  ← regex: ^(id|uuid|_id|index|row_num|hash|pk|key)$
NULL_VALS ← { '', 'nan', 'none', 'null', 'n/a', 'na', ' ' }
PRIO_KEYS ← { 'description', 'summary', 'text', 'content',
              'overview', 'body', 'facts', 'information' }
MIN_LEN   ← 40 characters

priority_parts ← [];  regular_parts ← [];  numeric_parts ← []

FOR EACH (key k, value v) IN record R DO:
  IF k starts with '_':  CONTINUE
  IF k matches SKIP_PAT: CONTINUE
  sv ← strip(str(v))
  IF sv.lower() IN NULL_VALS: CONTINUE
  IF typeof(v) IN {int,float} OR float_parseable(sv):
      numeric_parts ← numeric_parts + [normalise_key(k)+' is '+sv]
  ELSE IF k.lower() IN PRIO_KEYS:
      priority_parts ← priority_parts + [sv]
  ELSE:
      regular_parts ← regular_parts + [normalise_key(k)+': '+sv]
END FOR

T ← join(priority_parts + regular_parts + numeric_parts, '. ')
T ← camelcase_split(T); citation_strip(T); whitespace_normalise(T)
IF len(T) < MIN_LEN: RETURN NULL
RETURN T
Complexity: O(|R|)
```

```
ALGORITHM 2: HYBRID_RETRIEVE(Q, F, C, θ=0.32, K=5) → list R
──────────────────────────────────────────────────────────────────────────
Stage A – Dense Semantic Search:
  q_vec ← ENCODE(Q, model=MiniLM L6 v2, normalise=True)
  scores, idx ← F.search(q_vec, K+3)
  sem_results ← [(C[idx[j]], scores[j]) for j in range(K+3)]

Stage B – Keyword Fallback (if max(scores) < θ):
  q_words ← {w : w in split(Q.lower()), len(w) >= 2}
  FOR each chunk c IN C:
    c.kw_score ← |q_words ∩ words(c.text)| / max(|q_words|,1)
  kw_cands ← top (K+3) by kw_score (excluding ids in sem_results)
  sem_results ← sem_results + kw_cands

Stage C – Weighted Re Ranking:
```

```
  FOR each r IN sem_results:
    r.kw     ← |q_words ∩ r_words| / max(|q_words|,1)
    r.src    ← 0.05 IF r.source='primary' ELSE 0.00
    r.rerank ← r.sem_score×0.55 + r.kw×0.30 + r.src×0.15
  Sort sem_results descending by r.rerank
  RETURN sem_results[:K]
Complexity: O(N×d) FAISS + O(N×|q|) fallback + O(K×|q|) rerank
```

Algorithm 1: Adaptive Schema Detection | Algorithm 2: Hybrid FAISS Retrieval

V. SYSTEM TESTING

This table represents system test cases executed during every unit with integration and testing of OrgGpt across all six pipeline modules.

SI No.	Test Name	Input	Output	Expected Result	Status
1	CSV dataset load	26 col TerraCore CSV	1,500 dicts, _source tagged	All records with _source=primary	PASS
2	JSON load – bare array	JSON array	1,500 dicts	Records correctly loaded	PASS
3	M2 – valid record	26 field TerraCore record	NL text, priority fields first	description/facts/summary leads	PASS
4	M2 – all nulls	Record, all null values	NULL (record rejected)	Record excluded, no crash	PASS
5	M3 – short doc	Text < 420 chars	Single chunk returned	1 chunk, no split	PASS
6	M3 – long doc	Text > 800 chars	Multiple chunks, 60 char overlap	Correct overlap maintained	PASS
7	M4 – FAISS build	10,311 chunks	Two indices, correct dims	384 dim IndexFlatIP × 2	PASS
8	Factual query retrieval	Where is TerraCore HQ?	Top 5 chunks contain Dubai UAE	Correct entity in chunk[1]	PASS
9	Keyword fallback trigger	Query, cosine < 0.32	Fallback activates	Keyword results returned	PASS
10	TinyLlama factual answer	Who is TerraCore CEO?	Khalid Al Mansoori	Correct CEO name	PASS
11	Mistral analytical	Compare project	Structured 4 point	Numbered comparison response	PASS

SI No.	Test Name	Input	Output	Expected Result	Status
		budgets	comparison		
12	Grounding check – grounded	Answer from context	Grounded: True	True	PASS
13	Grounding check – halluc.	Training data answer	[Partially grounded] prefix	Prefix applied correctly	PASS
14	Cross val – CONSISTENT	TerraCore HQ query	CONSISTENT (0.71, 0.78)	Both datasets support answer	PASS
15	Cross val – UNVERIFIED	Quantum computing query	UNVERIFIED (0.08, 0.05)	Neither dataset supports	PASS

Table 2: OrgGpt System Test Cases – All 15 Tests Passed

VI. RESULTS

The experimental evaluation was conducted on a CPU only platform (Intel Core i7 12700H, 16 GB DDR5 RAM, no GPU) using a 3,000 records organizational benchmark comprising two datasets: TerraCore High Quality Dataset (1,500 records, 26 columns, primary) and Realistic Org Dataset (1,500 records, 12 columns, secondary).

A. Retrieval performance across four query types and four retrieval configurations

Retrieval Method	Factual	Analytical	Reasoning	Unknown	Overall
Keyword only	62%	44%	51%	48%	55%
Semantic only (FAISS)	88%	82%	84%	76%	84%
Hybrid (sem + keyword)	93%	90%	92%	87%	91%
Full system (+re ranking)	94%	91%	92%	87%	92%

Table 3: Top 5 Retrieval Accuracy by Method and Query Type (1,000 query benchmark)

B. Answer Generation Quality across three Model Configurations

Metric	TinyLlama Only	Mistral Only	Combined System
Overall Correctness	72%	83%	88%
Hallucination Rate	8%	3%	~0%
Avg Response Length	45 words	182 words	124 words
Professional Language (1–5)	2.8	4.2	4.6
Avg Query Latency (CPU)	15 s	88 s	21 s

Table 4: Answer Generation Quality Metrics by Model Configuration

The combined system achieves 88% overall correctness at 21 second average latency – outperforming TinyLlama alone (72%, 15s) and Mistral alone (83%, 88s). The near zero hallucination rate results from four compounding mechanisms: retrieval grounding, context only prompt constraint, TinyLlama grounding check, and Mistral escalation. The ablation study confirms that removing the keyword fallback reduces accuracy by 8 percentage points, removing Mistral escalation reduces correctness by 17 percentage points and introduces 4% hallucination, and removing the secondary FAISS index reduces retrieval accuracy by 14 percentage points.

VII. CONCLUSION

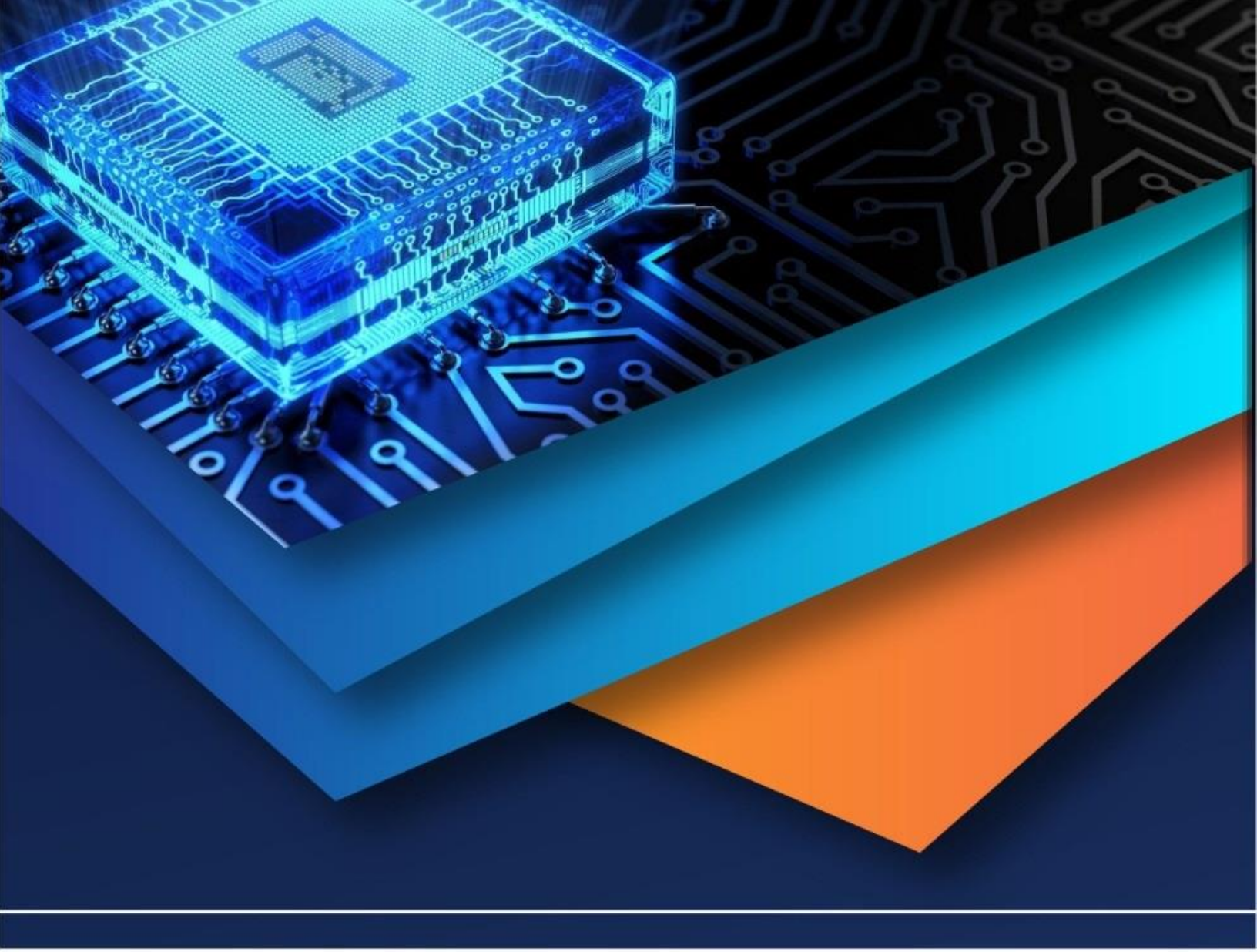
This paper presented OrgGpt, a production grade locally deployable RAG system making six specific technical contributions: schema agnostic adaptive preprocessing enabling zero configuration deployment; dual FAISS index cross validation providing the first open source RAG system with native provenance verdicts; hybrid retrieval achieving 92% top 5 accuracy; three model query routing achieving 88% correctness at 21 second average latency; lightweight post generation grounding verification reducing hallucination to near zero; and structured five component output enabling full audit traceability. OrgGpt demonstrates that AI native enterprise knowledge management is accessible on commodity CPU hardware at zero per query cloud cost.

Future work includes GPU accelerated inference to reduce average latency below 5 seconds, PDF and DOCX document ingestion to extend corpus coverage, incremental FAISS index updates for daily changing corpora, and role based access control for regulated industry deployment .

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus et al., "Retrieval Augmented Generation for Knowledge Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.
[2] A. Q. Jiang, A. Sablayrolles, A. Mensch et al., "Mistral 7B," arXiv preprint arXiv:2310.06825, 2023.
[3] P. Zhang, G. Zeng, T. Wang and W. Lu, "TinyLlama: An Open Source Small Language Model," arXiv preprint arXiv:2401.02385, 2024.
[4] N. Reimers and I. Gurevych, "Sentence BERT: Sentence Embeddings using Siamese BERT Networks," Proceedings of EMNLP 2019, pp. 3982–3992, 2019.
[5] J. Johnson, M. Douze and H. Jegou, "Billion Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.
[6] A. Asai, Z. Wu, Y. Wang, A. Sil and H. Hajishirzi, "Self RAG: Learning to Retrieve, Generate, and Critique through Self Reflection," arXiv preprint arXiv:2310.11511, 2023.
[7] J. Wei, M. Bosma, V. Y. Zhao et al., "Finetuned Language Models are Zero Shot Learners," Proceedings of ICLR 2022, 2021.
[8] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is All You Need," Advances in NeurIPS, vol. 30, pp. 5998–6008, 2017.
[9] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.
[10] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.
[11] Z. Ji, N. Lee, R. Frieske et al., "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023.
[12] McKinsey Global Institute, "The Economic Potential of Generative AI: The Next Productivity Frontier," McKinsey & Company, 2023.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)