Santhosh Kammari

+91-7674874563 santhoshkammari1999@gmail.com LinkedIn GitHub

Summary

GenAI Engineer with **2+ years** in applied AI/ML with **production deployments**, specializing in **LLM-powered document intelligence**, hybrid retrieval systems, and **agentic pipelines** for financial/regulatory document processing. Built end-to-end systems deployed at **RBI** and banking clients — spanning multi-stage PDF ingestion, hybrid vector+SQL query engines, and UCP 600 rule validation. Delivered **3+ production systems** from scratch with 103+ commits across GenAI R&D components including multi-agent frameworks, RAPTOR-based multi-hop QA, and embedding model fine-tuning.

Education

Indian Institute of Information Technology and Management, Gwalior **Aug 2018 – June 2023**
Integrated Dual Degree (BTech + MTech), Information Technology *Gwalior, MP*

Technical Skills

Languages: Python 3, SQL
Python Packages: PyTorch, Hugging Face, Scikit-learn, NumPy, Pandas, OpenCV, NLTK, Pydantic
NLP & LLMs: Information Extraction, RAG, Multi-hop QA, Prompt Engineering, Embedding Fine-tuning, Text2SQL, Qwen3, LLaMA, Ollama, vLLM, BitsAndBytes (QLoRA)
LLM Frameworks: LangChain, LangGraph, LlamaIndex, DSPy, Autogen, OpenAI client
Databases & Vector Stores: Milvus, FAISS, SQLite, MongoDB, MySQL
Engineering: FastAPI, Gradio, Git, Langfuse, Tesseract OCR, Docker, Uvicorn
Communication: English (Professional), Hindi (Fluent), Telugu (Native)

Experience

Newgen Software – Number Theory Group **July 2023 – Present**
Data Scientist *Gurugram, Haryana*

- **Sole architect and sole developer** of RBI's AI-powered Decision Support System – **114 commits, 57 Python modules, ∼13,872 LOC** delivered in **3 weeks**; covered ingestion pipeline, hybrid query engine, similarity rules engine, 5 FastAPI GPU microservices, and Gradio UIs – zero handoff gaps.
- **8-stage GPU-orchestrated PDF ingestion pipeline**: DOTS OCR → Qwen3-14B text extraction → PDF-to-images → Qwen3-VL vision extraction → LLM ensemble merge → Milvus chunking → GTE+BGE-M3 embedding insert → SQLite insert. Each GPU step **starts and kills its own model server** to prevent OOM on shared GPU hardware; pipeline is **resume-safe** (already-processed files skipped) and fully concurrent via ThreadPoolExecutor.
- **Dual-model ensemble extraction**: Qwen3-14B (text) and Qwen3-VL (vision) extract fields independently from the same document; a **third LLM call arbitrates** both outputs using Pydantic-enforced JSON schema – handles stamps, handwriting, and tables that OCR alone misses. Prompts evolved through 4 generations to a single **unified appointment+remuneration prompt** per doc type.
- **Hybrid RAG + Text2SQL query engine**: LLM classifier routes per query → **GTE-large-en-v1.5 (1024-dim dense) + BGE-M3 sparse** Milvus search → Qwen3-Reranker-0.6B re-scoring → LLM answer synthesis; Text2SQL uses **majority-vote generation** (N SQL candidates from Snowflake Arctic-7B, `Counter.most_common()` consensus, second LLM pass for syntax fixes); **automatic SQL-to-vector fallback** on empty results – zero dead ends.
- **Multi-stage stateful chat memory system** – the most architecturally complex component: (1) **History-intent pre-classifier** (1 LLM call) short-circuits "summarize above"-type queries directly to history, **saving 9+ LLM calls** per turn; (2) context built from **lastK recency + relevantK semantic retrieval** (dense search + keyword/text-match reranked) over a per-session Milvus `CHAT_HISTORY` collection; (3) **4-variant query refinement** resolves pronouns/references using history context; (4) **sufficiency classifier** (majority vote, 4 calls) decides if history alone answers the query or pipeline answer is needed; (5) final **LLM merge** of history answer + pipeline answer when both are relevant. Total: **8–14 LLM calls per turn** on normal path vs 2 on history-only path.

- **5-rule NBFC Similarity Engine** running all rules concurrently at application-receive time across **3 SQLite databases** (NBFC, Gov Banks, Gov Non-Banks): Rule 1/2 – vector COSINE search (GTE-large, group-by application_id) for current/proposed activity similarity with generic-proposal LLM filter; Rule 3 – SQL IN-match on shareholder `jurisdiction_country` (skips all-Indian cases, sorts by %); Rule 4 – NLTK keyword extraction + LIKE-match on group company activities with NLTK stop-word filtering; Rule 5 – **cross-DB exact name JOIN** (directors, shareholders, group companies matched against all 3 DBs including Gov Banks/Non-Banks association and experience tables, DIN-primary match) – zero result cap, every overlap surfaced for the dealing officer.
- Built **5 production FastAPI microservices** on GPU3 (192.168.170.76): DOTS OCR (port 8002), Qwen3-VL vision (port 8099), Snowflake Arctic Text2SQL (port 8004/8022), Qwen3-Reranker-0.6B (port 8024), GTE-large embedding (port 8025); all managed via shell lifecycle scripts with **GPU memory tuned to 40% utilization** per model; integrated **Langfuse** for LLM trace observability; built human annotation Gradio UI for reviewing/correcting extractions side-by-side with original PDF images.

Number Theory **July 2022 – June 2023**
AI/ML Engineer *Gurugram, Haryana*

- **Primary author of the Trade Finance Rule Engine** (deployed at banking clients) – **591-line rule orchestrator** (`rules.py`) + **3,358-line operations library** (`operations.py`) implementing **40+ operation types** across UCP 600, CSCH, and bank-custom rules; orchestrator dynamically routes each rule by ID prefix (before-extraction / after-extraction / consistency / src-dest) through a configurable schema dispatch layer – rule definitions loaded from MongoDB, results written back per LC number.
- **BERT-based signature verification** (`signed_by_named_party()`): bank documents have handwritten/stamped signatures in arbitrary image positions – solved by combining **Tesseract OCR coordinate extraction** to locate "on behalf of" signature regions, BERT cosine similarity (`sentence-transformers`) to match party names against expected signatories, fuzzywuzzy fallback for abbreviations/aliases, and a two-pass fallback (OCR region match → direct key presence check) – handles all real-world sign variants without manual template rules.
- **Cross-document numeric reconciliation** (`number_comparison()`): locale-aware parsing (`locale.atof`, `word2number`, regex digit extraction as 3-tier fallback) handles `1,000,000.53` and word-form amounts; configurable tolerance bands (`agg` field as % of source amount) for GTE/LTE compliance – exact diff amounts returned per rule for auditor review; handles currency normalization and nested key traversal across arbitrary extraction schemas.
- **Structured Document Extraction API** (primary author, 6 months, **3,652 lines across 6 modules**) – FastAPI service extracting structured key-value fields from 5 LC form types (LC Application, Export Letter, LC Amendment, Letter of Indemnity, Export Bill) deployed at banking clients. Core innovation: a **spatial coordinate engine** (`definitions.py`, 1845 lines) that locates any field on any bank's form variant without retraining – each field is defined by a **declarative geolocation schema** (right_of, left_of, bottom_of, top_of, location zone) resolved at runtime against live Tesseract OCR token coordinates.
- Coordinate resolution pipeline: (1) **fuzzy token matching** (`get_bbox_combined_geolocation`) scans OCR token stream line-by-line using `fuzz.ratio` + `partial_ratio` with special-char normalization to locate field label coordinates; (2) **parent-anchor scoping** – each field's value search is constrained to the spatial zone relative to its parent label's bounding box, eliminating false matches from other form sections; (3) **multi-line keyword handling** for labels split across OCR lines; (4) **document-level padding calibration** – median horizontal/vertical inter-token gap computed from full OCR stream and used as tolerance for spatial proximity checks. (5) **value zone extraction** (`get_value`) computes the bounding zone from geolocation constraints (xmax from left_of anchor, ymax from top_of anchor) and collects all OCR tokens inside – handles right-of, left-of, bottom-of multi-constraint intersections.
- Typed value pipeline: extracted raw text passed through **DataTypeHandler** (dispatch by field type: date, amount, alphanumeric, string) → **DataTypeValidator** for format verification → confidence score output. **CV2-based checkbox detection** (`checkbox_detection.py`, 590 lines): extracts ROI from image by field coordinates, applies adaptive thresholding + contour detection to determine checked/unchecked state. Two-pass extraction: semi-auto API result used if available, `field_not_in_semi_auto_api()` fallback for any field the upstream model missed – **zero fields left blank** regardless of template variation.
- **Perceptual hash bucketing for document deduplication** (new algorithm, Jan 2024): replaced $O(n^2)$ pairwise image comparison with **Neal-Krawetz dhash bucketing** – hash all pages O(n), compute Hamming

distances only within buckets; **50-page bundle: 1225 comparisons** → ∼**25**; handles N-way duplicates (same document submitted 3×) with minimum-Hamming-distance assignment to resolve multi-ownership conflicts; supports 5 hash methods (CNN, WHash, AHash, PHash, DHash) with configurable threshold.

- Co-authored the **Bundle API** `/omni` **endpoint** orchestrating **8 downstream microservices** sequentially (pre-validation → classification → LC extraction → structured doc extraction → document indexing → rule engine → group API → pre-calculation); fixed a **race condition** where document indexing was called before extraction completed – resolved with explicit result-validation gates and per-service configurable timeouts via `post_with_retry()`; added JWT auth, rate limiting (30 req/10s), and per-workitem structured logging throughout.

Projects & POCs

MeantFor – RBI Regulatory Entity Classifier (POC → Production) | *Python, NLTK, Pydantic, vLLM, MLflow* **2025**

- **Sole author** of a production NLP pre-routing module in RBI DSS: query → **conflict-aware NLTK abbreviation expansion** (ALL-CAPS-only matching for conflicting words like "ARE"/"FOR" to prevent false expansions; clean abbreviations matched case-insensitively) → **lemma-sequence department tagging** (multi-word department names matched via WordNetLemmatizer with `max_gap=2` non-stopword tolerance, position-offset tracking for in-place tag insertion) → optional **token-overlap tree pruning** (`relevant_leaves`: keeps only leaves with ≥1 query token overlap, then `build_pruned_tree` reconstructs minimal hierarchy – reduces LLM context for large trees) → LLM classifier on XML-serialized tree.
- LLM receives entity tree as **structured XML context** + tag-enriched query; outputs Pydantic-enforced JSON `{including, not_applicable}` mapped to a **4-root multi-label hierarchy** (banks, nbfc, all_india_financial_institutions, asset_reconstruction_company) with both root-category and low-level leaf resolution; **regex JSON salvage** and Qwen3 `<think>...</think>` stripping before parse. Department processor supports two modes: `tag_value` (readable enrichment) and `metadata` (emits Milvus filter expressions like `metadata["department"] like "%DBR%"`). Pipeline additionally supports **date-range filtering** (`filter_by_date` on `date_of_issue`) and **RBI reference-number filtering** (`filter_by_reference_number` for circular code patterns like `RBI/2024-25/123`). Prompt iterated across 8 experiment runs from **80%** → **93.99% high-level / 92.90% low-level accuracy** on **183 eval + 240 synthetic** samples.
- **Full MLflow eval pipeline**: parallel `ThreadPoolExecutor` inference → JSONL streaming → multi-label sklearn metrics (exact-match accuracy, micro/macro/weighted F1, per-class P/R/F1) → multilabel confusion matrix artifacts per level → **interactive failure tables** (high-fail / low-fail / both-fail splits logged to MLflow UI) – every run fully reproducible by run ID; `-think` flag enables chain-of-thought for ablation; `-no-abbrv`/`-no-dept` for component ablation.
- Refactored from monolithic POC into a **production Python package** (`meant_for/`: separate modules for `core/`, `filtering/`, `tree/`, `llm/`, `preprocessing/`); `MeantForFilter` supports `warmup` parameter (pre-warms vLLM KV cache on init) and per-call `optimize` flag (enables token-overlap tree pruning); deployed as FastAPI flag (`meant_for_filter: bool`) gating which of the **4 root Milvus collections** are searched per query.

3-Tier Priority Retrieval & Reranking Engine | *FastAPI, Milvus, Qwen3-Reranker-0.6B, vLLM, pymilvus* **2025**

- **Designed and implemented a production 3-tier retrieval architecture** over **5 Milvus collections** on Azure (`10.2.2.22`, database `RBI_2026`): **Alpha** (RBI_Master_Direction_Root + RBI_Master_Circular_Root + RBI_Circular_Root – highest authority) parallel-searched 20 chunks each → reranked → top-20; **Beta** (RBI_Notification_Root + Legal_Framework_RBI FAQs) and **Gamma** (Legal_Framework_RBI Acts/Regulations/Rules, same collection different metadata filters) searched in parallel → combined → top-5; **final 25 chunks** = alpha-20 + beta/gamma-5 always surface highest-authority documents first.
- **Reranker**: `Qwen3-Reranker-0.6B` via vLLM (`gpu_memory_utilization=0.1`, `kv_cache_dtype=fp8`, `max_model_len=8192`, tokens truncated to 6144 before generation, `enforce_eager=True`); scoring: **yes/no token logit extraction** – `exp(logprob_yes) / (exp(logprob_yes) + exp(logprob_no))` per chunk, sorted descending; dense search uses **IP metric,** `anns_field="dense"`**,** `consistency_level="Strong"`; deduplication by **(filename, page_number) composite key** before reranking.

- FastAPI `/query` endpoint: accepts `query_decomposition`, `thinking_mode`, `meant_for_filter`, `sources` flags; per-query **debug folder** generated with timestamped JSON snapshots at each step (00_query_input, 01_decomposed_queries, 02_subquery_results, 99_final_response); sub-queries executed in **parallel ThreadPoolExecutor** across sources; final answer synthesized by a **comprehensive answer LLM** from all sub-query answers.

Multi-Agent Deep Research System | *Python, LangGraph, Custom Agent Framework* **2025**

- Built a multi-agent deep research system operating over **1000+ web pages** with a MarkdownAgent and context management layer that scales to **100 parallel subagents**.
- Benchmarked on **BrowseComp** and achieved **85% accuracy** – competitive with frontier research agents.

DocVeda – Enterprise Document Intelligence Platform | *FastAPI, Milvus, doclayout-YOLO, Qwen3* **2025**

- Designed and built a **modular production RAG platform**: OmniDocs fetch → DOTS OCR → **doclayout-YOLO layout parsing** → semantic chunking → GTE dense embedding → Milvus insert; query pipeline: **LLM metadata filter expression generation** (LLM reads Milvus scalar field schema, outputs filter on the fly) → dense vector search → **(filename, page_number) deduplication** → cross-encoder reranking → optional LLM synthesis; full per-stage timing telemetry.
- Architected with **CUDA 12.6 + ONNX Runtime** backend; extensible embedding/reranker plugins; **LLM Guard** integration for prompt injection safety; `doclayout-YOLO` for layout-aware chunk boundaries (tables, figures, headers separated from body text).

Hybrid Sparse+Dense+Reranker Retrieval Benchmarks | *Milvus, BGE-M3, BM25, DeepSeek, pymilvus* **2024–2025**

- Ran **systematic end-to-end benchmarks** across sparse (BM25), dense (GTE, BGE-M3), weight-based hybrid fusion, and **RRF (Reciprocal Rank Fusion)** over 929 TradeFinance + 719 Email queries at chunk-level and document-level; **BGE-M3 + RRF**: Precision@1=70.18%, MAP@10=71.9%, NDCG@10=82.14% on TradeFinance; used **DeepSeek as LLM evaluator** against human-annotated ground truth; benchmarked across both chunk-level and document-level relevance.
- Quantified hybrid+RRF outperforms all single-modality baselines; findings directly drove the **GTE dense + BM25 sparse dual-embedding** architecture in the RBI production system and selection of Qwen3-Reranker for the tier-merge pipeline.

RBI Multi-Hop & LLM Metadata Filter POC | *Milvus, NLTK, Pydantic, SQLite* **2024–2025**

- Designed a **multi-hop retrieval pipeline** for 14K RBI PDFs: entity extraction → entity expansion → multi-turn Milvus searches with **proof-of-document citation chaining**; evaluated across 100+ annotated multi-hop queries tracking answer completeness and citation accuracy.
- Built **LLM-based Milvus filter expression generator**: LLM reads scalar field schema and outputs valid pymilvus filter expressions (`metadata["field"] == "value"` syntax) on the fly; systematic issue tracking of ambiguous/missing metadata fields across 14K docs resolved before production ingestion.

Text2SQL Evaluation & Query Decomposition Engine | *FastAPI, SQLite, Snowflake Arctic-7B, vLLM* **2025**

- Benchmarked **Snowflake Arctic-7B vs. baselines** on RBI's 7-table SQLite schema (100+ queries); metrics: execution accuracy, schema recall, SQL syntax validity; validated **majority-vote SQL** (N candidates → `Counter.most_common()` → second LLM syntax-fix pass) as the production strategy.
- Built **query decomposition module** (production FastAPI service): complex queries decomposed via LLM (`/no_think` suffix for speed); `requires_decomposition` flag check avoids unnecessary splits; sub-queries executed in **parallel ThreadPoolExecutor** across sources; answers synthesized by a **comprehensive-final-answer LLM** – all wired into the tier-merge retrieval API.

Vector DB Quantization & GPU/CPU Cost Benchmark | *Milvus, llama.cpp, Q4KM, FP16* **2025**

- Benchmarked **LLM-generated metadata filter queries** (llama3_8b Q4KM) on CPU vs GPU: improved prompt raised **execution accuracy from 48% → 96% on CPU** and 40% → 96% on GPU (same accuracy, both platforms); CPU avg 69.2s vs GPU 0.81s (85× faster on GPU); **llama3_3b Q4KM CPU**: 0% → 100% with 57% time reduction – informed which model/platform to use for metadata filter generation in production.
- Separate **VDB retrieval quality benchmarks**: compared llama8b/llama3b across CPU/GPU configurations; **average 55.77% time reduction** and **80.8% accuracy gain** across all models after prompt optimization – documented Pareto analysis across model size, quantization, and hardware for infrastructure decisions.

Qwen3 Domain Fine-Tuning on RBI Corpus | *HuggingFace, QLoRA, Qwen3, SFT* **2025**

- Prepared **domain-specific instruction-tuning dataset** from RBI regulatory PDFs for Qwen3 fine-tuning; built metadata extraction, preprocessing, and SFT data pipeline from raw HTML and PDF sources; tracked experiment results in structured eval spreadsheets.
- Targeted improvement in **structured field extraction accuracy** for NBFC/banking regulatory documents – complementing the vLLM-served inference path in production with a lighter fine-tuned model alternative.

Multi-Agent Collaboration & Trust-Learning System (Research) | *Python, AutoGen, Multi-LLM* **2024**

- Designed and documented a **heuristic multi-agent collaboration framework**: planner decomposes tasks → agent pool votes on assignment → agents operate in **independent / collaborative / transfer** modes; per-agent **skill confidence scores** (0.0–1.0 per capability) gate handoff thresholds (independent if score >0.8, collaborative if 0.5–0.8, transfer if <0.5); **trust scores** updated from past task outcomes influence future delegation.
- Evaluated **AutoGen** (dev wheel `autogen_agentchat-0.4.0.dev7`) as a framework alternative; adapted design to **software company simulation** (UI, DS, Backend, QA teams); documented agent skill profiling, knowledge-memory vs environment-state separation, and MAQL/DQN-inspired delegation strategies – published internal research writeups and a medium article series.

Multi-Agent Framework from Scratch | *Python, Ollama, HuggingFace* **2024**

- Built a **multi-agent orchestration framework** before LangGraph gained adoption – agent pool, communication hub, dynamic step generator that re-plans mid-execution based on intermediate agent outputs.
- Implemented **multi-LLM backend routing** (Ollama local + HuggingFace remote) with task decomposition, environment simulation, and real-world scenarios (climate reporter, data science pipeline, search mixture).

RAPTOR Multi-Hop Question Answering | *FAISS, SentenceTransformers, LlamaIndex* **2024**

- Implemented **RAPTOR hierarchical retrieval** – cluster → summarize → retrieve tree built over document corpus; custom embedding and summarization models wired to FAISS retriever for multi-document reasoning.
- Built GPU-batched Wikipedia corpus loader and **evaluation harness** with multi-hop answer quality metrics, outperforming flat top-k retrieval on cross-document questions.

Embedding Model Fine-Tuning + VDB Benchmarking | *Milvus, SentenceTransformers, llama.cpp* **2024**

- Fine-tuned domain embedding model using **synthetically generated QA pairs** (8-bit quantized LLM as zero-label data generator); automated SFT training → HuggingFace Hub publish pipeline via SentenceTransformersFinetuneEngine.
- Benchmarked **8 embedding models** (gte-large-en-v1.5, all-MiniLM-L6-v2, gte-modernbert-base, nomic-embed-text-v2-moe, stella_en_400M_v5, e5-large-v2, gte-base-en-v1.5, and fine-tuned variant) across Precision@k, MAP@k, MRR@k, NDCG@k (20 metrics); **finetuned_gte_1: Precision@1=0.796 vs base gte-large 0.764** (+4.2%); annotator evaluation confirmed retrieval precision improvement before deployment.

Query Decomposition Engine | *LangChain, Pydantic, OpenAI* **2024**

- Built an LLM-powered engine decomposing complex NL questions into **N independent sub-queries** with **chain-of-thought prompting (V_2_0_COT)** and Pydantic-constrained structured output parsing; evaluated via weighted scoring harness on Wikipedia-derived synthetic Q&A.

LLM Fine-Tuning for Key-Value Extraction | *HuggingFace, QLoRA, PEFT* **2024**

- Fine-tuned open-source LLM for structured key-value extraction from regulatory documents; improved extraction accuracy from **50% to 90%** via domain-specific instruction tuning with QLoRA.

Fraud Detection on Bank Payments using ML | *Python, Scikit-learn, NumPy, Pandas* **2022**

- BTech thesis: ML classification on Banksim dataset achieving **96.64% accuracy**; published in **IEEE** – International Conference for Advancement in Technology (DOI: 10.1109/ICAT54021.2022.9726104).

Publications

Fraud Detection on Bank Payments using Machine Learning ***Santhosh Kammari*** Jan 2022 **DOI**
International Conference for Advancement in Technology (IEEE) **Certificate**