AIMResearcher - AI Research Assistant

An AIM Lab Project
June - August’25

Team Members
Aliza Saadi
Rameen Babar
Mehrunisa Nasir

What is AIMResearcher?

AIMResearcher is an open-source pipeline that ingests the flood of AI/ML literature, pinpoints only the papers truly relevant to a researcher's query, and distils each paper's scope, limitations, and datasets into structured, machine-readable outputs — all delivered alongside a narrated micro-summary.

Key value propositions:

1. **Relevance, not noise** – Hybrid dense–sparse retrieval cuts false positives that plague keyword search.

2. **Instant scope mapping** – JSON output of contributions, limitations, and future-work lets researchers spot gaps in seconds.

3. **Dataset registry on the fly** – Extracts benchmark names verbatim for hassle-free replication planning.

4. **Shareable insights** – Generates a 40-second narrated video script so findings travel beyond the lab.

5. **100 % open source** – Built on LangChain, LangGraph, Qdrant, and Mistral-7B; no proprietary lock-in.

Problem Statement

Every week, more than 5,000 new AI/ML papers hit arXiv alone. You fire up Google Scholar, type "reinforcement learning for robot navigation," and - boom - 150 PDFs stare back. You start reading one that *looks* promising, burn 30 minutes wrestling with its algebra, and then discover the authors are optimising Atari games, not robots. Multiply that false start by a semester and you've spent days on papers that are tangential - or totally off-topic.

The core pain points:

1. **Finding truly relevant work** - Keyword search returns near matches and outdated citations; you still have to skim dozens of abstracts.

2. **Grasping each paper's *scope* quickly** - Limitations and future-work sections are buried near the end, written in dense prose.
3. **Tracking datasets** - Identifying which benchmarks a study uses (and whether they're public) is tedious manual extraction.
4. **Summarising outcomes** - Turning insights into a short briefing - or a shareable video - requires even more time.

Researchers need an assistant that pinpoints only the papers that matter and surfaces their scope, datasets, and a crisp summary without the rabbit-holes.

Solution Overview

We built an end-to-end pipeline powered entirely by open-source components that:

1. **Ingests** AI/ML papers automatically from arXiv.
2. **Stores** semantics using both dense and sparse embeddings in a Qdrant hybrid index.
3. **Retrieves** the five most relevant papers for any natural-language query.
4. **Extracts** each paper's *scope / limitations* and *datasets* in structured form.
5. **Generates** a narrated slide-style video summarising each paper.

The entire stack is open-source: LangChain + LangGraph for orchestration, Qdrant for storage, BGE & BM25 for embeddings, Mistral-7B for reasoning, and MoviePy / gTTS for video.

Methodology & Key Design Choices

Component	Rationale	Highlights
LangChain + LangGraph	Provide plug-and-play primitives (loaders, embeddings, vector store) and a state-graph to link them, so our ingestion → retrieval → extraction pipeline is built from clear nodes we can swap or extend in a single line.	Rapid prototyping, strict typed-state validation, and easy future branching/parallelism.
Qdrant Hybrid Index	Combines dense semantic similarity with BM25 keyword exactness in one call.	GRPC for speed; scales to millions of chunks.
Dense Embedding: BAAI/bge-base-en-v1.5	Strong semantic recall, 768-dim, CPU-friendly; no proprietary license.	Loaded on CPU in production, CUDA optional during ingestion.
Sparse Embedding: BM25 via FastEmbedSparse	Retains exact keyword matches, boosting recall for niche terms.	Works natively inside hybrid retrieval.
Semantic Gradient Chunking	Keeps paragraphs intact while reducing prompt size by > 70 %, lowering LLM latency.	20 chunks per paper on average.
LLM: Mistral-7B-Instruct via Ollama	Best balance between hallucination rate and CPU latency among tested models.	Temperature 0 to maximise determinism.
Prompt Design	Separate prompts for query rewriting, scope extraction, dataset extraction, and narration.	Few-shot examples live in dedicated prompt files.
Video Composer	Converts summary to audio with gTTS, combines PDF page images via PyMuPDF + OpenCV, stitches with MoviePy.	30–50 s narrated video per paper.

Workflow Summary

1. **Data ingestion**
   • Query arXiv for “Artificial Intelligence and Machine Learning” papers.
   • Enrich metadata with stable URLs.
   • Split each paper into ~20 semantic chunks.
   • Embed every chunk with BGE-base (dense) and BM25 (sparse) and upsert into Qdrant (“axRiv_research_papers”).
2. **Retrieval-Augmented Generation (RAG) pipeline**
   • Query transformation node rewrites the user’s question for clarity (e.g., expands acronyms).
   • A hybrid search (dense + BM25) retrieves the top-k chunks, their metadata, and similarity scores.
   • Those chunks form the external knowledge context that is injected into Mistral, enabling the LLM to generate an answer grounded in the retrieved evidence rather than model priors.
3. **Scope extraction**
   • The same retrieved context feeds a dedicated “scope” prompt that asks for contributions, limitations, and future-work.
   • Output is parsed to JSON for downstream automation.
4. **Dataset extraction**
   • Another prompt asks the model to list dataset names exactly as mentioned in the paper.
   • Output is validated and stored as a Python-list string.
5. **Video Generation**
   • Summary → conversational narration via gTTS (English TTS).
   • Corresponding PDF converted to high-resolution page images (PyMuPDF).
   • OpenCV repeats each page frame to match narration timing; MoviePy merges audio & frames.
   • Output: `final_video.mp4` — a ready-to-share 1080p explainer.

LLM Comparative Analysis

Model	Avg CPU Latency (min)	Accuracy (manual)	Hallucinations	Verdict
gemma3	3.5	★☆☆☆☆	High	Fast but unreliable.
Mistral	25 - 35	★★★★☆	Low	**Chosen;** accuracy > latency
DeepSeek-R1	41+	★★☆☆☆	Medium	Too slow, fabricates data.

Mistral's < 5 % hallucination rate outweighs its 7 × latency penalty for research-grade outputs.

Trade-Off Discussion

Latency vs. Accuracy:

Gemma responded in under four minutes but misidentified scopes or invented datasets about 20 % of the time. DeepSeek was both slower and less truthful. Mistral's 5 % hallucination rate justified its higher latency for our use case, especially when run asynchronously.

Hybrid Retrieval:

Pure dense embedding missed exact-term queries; pure BM25 ignored synonyms. The hybrid approach improved NDCG@5 by 0.12 on a 50-query dev set.

Chunking Strategy:

Reducing average prompt size from roughly 7 100 to 2 000 tokens shaved several minutes off Mistral's generation time.

Possible Future Improvements

1. **GPU accelerate Mistral**: Run the 7 B model on an A100 or Apple-silicon GPU via vLLM/llama.cpp to cut inference from ~25 min to ≤ 6 min per query while keeping cost under $0.15.

2. **Parallelise scope + dataset extraction**: Split the LangGraph after retrieval, run both prompts concurrently, then merge - targeting a 30 % drop in end-to-end time.

3. **Stream answers live:** Enable token streaming in the Ollama client so users see the first words within two seconds; add a “stop / refine” control in the UI.

4. **Automate hourly arXiv ingestion:** Use GitHub Actions to pull new cs.AI/cs.LG/cs.CV papers, deduplicate by arXiv ID, chunk, embed, and upsert; trigger a Slack alert on any ingestion lag > 60 min.

5. **Auto-publish narrated videos:** After rendering, upload the MP4 plus scope and dataset JSON to the internal portal, tagged by arXiv ID, within ten minutes of video completion.

6. **Scalability:** Containerise services and deploy on Kubernetes; shard Qdrant and front-end everything with a message queue. Autoscale CPU ingestion pods by feed lag and spin up GPU inference pods on demand, allowing the system to index 1 M+ papers and serve 100 concurrent queries with sub-2 s retrieval latency.

AIMResearcher turns the relentless torrent of AI literature into precise, actionable insight - merging hybrid retrieval, RAG grounding, and automated video narration in a fully open-source, container-ready stack. It frees researchers from time-sink skimming, surfaces verifiable scopes and datasets in seconds, and scales seamlessly to millions of papers and hundreds of simultaneous queries. With AIMResearcher, the focus shifts from *finding* knowledge to *extending* it.