EPiC Series in Computing

Volume 109, 2026, Pages 184–195

Proceedings of EUNIS 2026 Annual Congress

Andarax: Channeling the Flow of Knowledge Through a Secure Distributed LLM Architecture

Arun Sharma, Joaquim Campuzano, and Juan Antonio Martínez-Carrascal

Universitat Autònoma de Barcelona, Barcelona, Spain
arun.sharma@autonoma.cat, joaquim.campuzano@uab.cat, juanan.martinez@uab.cat

Abstract

We present **Andarax**, a Retrieval-Augmented Generation (RAG) system designed and deployed at the Universitat Autònoma de Barcelona (UAB) to serve as the foundation for the implementation of AI initiatives within the university. This paper describes the system architecture, the RAG pipeline, the infrastructure challenges encountered during development, and the practical lessons learned from building an LLM-based system within the constraints of a university environment, offering guidance for other institutions seeking to develop similar systems.

Andarax is built on a distributed architecture of seven virtual machines and supports dual LLM inference (Qwen2.5-32B and Mistral-7B, both AWQ-quantized and deployed on NVIDIA L40S GPUs), multilingual output in Catalan, Spanish, and English, and real-time streaming responses. As an initial use case, currently deployed for internal use and undergoing evaluation, the system has been configured as an AI teaching assistant for university courses. It provides students and teaching staff with accurate, citation-backed answers derived from course materials (lecture slides, PDFs, and notes), while operating entirely on university-owned infrastructure to ensure data sovereignty and GDPR compliance.

Keywords: Generative AI, Retrieval-Augmented Generation (RAG), Large Language Models, Educational Technology, On-Premise AI, Multilingual NLP, data sovereignty

1 Introduction

Large Language Models (LLMs) have demonstrated strong capabilities in natural language understanding and generation [1, 2]. In educational settings, students are increasingly using general-purpose tools such as ChatGPT for academic support. However, these tools use LLMs that present relevant concerns and limitations in this context: they lack access to institution-specific course materials, they can produce incorrect information, and they offer no traceability to source documents [3].

Retrieval-Augmented Generation (RAG) [4] addresses these issues by grounding LLM responses in retrieved documents. As a design consideration, the system design considers the support for different languages (for instance, in our case, Spanish, Catalan, and English). On-premise installation was chosen. From a practical perspective, this requires coordination efforts with university IT administrators who have their own priorities and timelines. Although EU

data protection regulations such as the GDPR (GDPR [5]) do not require on-premise deployment, on-premise infrastructures can reduce certain compliance and data sovereignty risks by providing greater control over data location, access, and processing activities.

In this scenario, we present **Andarax**[1]. Andarax defines a scalable architecture for developing potential AI-based systems in higher education environments. It is a production RAG system, that has been deployed at UAB and:

- Provides course-specific answers with verifiable citations to source documents and page numbers.
- Runs entirely on university-owned infrastructure (seven VMs) with no external API dependencies.
- Supports dual-model LLM inference (32B for quality, 7B for speed) using AWQ quantization. In our case, it runs on NVIDIA L40S GPUs with 48 GB VRAM.
- Handles multilingual queries and responses in Catalan, Spanish, and English.
- Is designed to evolve beyond course assistance into a broader university service platform.

The system is currently deployed as a testbed for UAB staff. Future plans include its extension and broader use after the formal evaluation phase. This paper describes the architecture, implementation, challenges, and lessons learned during the development of Andarax.

2 State of the Art

2.1 Large Language Models in 2025–2026

The LLM landscape has matured significantly since the release of GPT-3 [1]. Open-weight models now rival proprietary systems across many benchmarks. The Qwen2.5 family [6] achieves competitive performance with Llama 3.1 and GPT-4 on reasoning and multilingual tasks, while being available for on-premise deployment. Mistral-7B [7] showed that smaller models with careful training can outperform much larger predecessors, making local GPU inference practical on academic hardware.

Quantization techniques, particularly AWQ (Activation-aware Weight Quantization) [8], have enabled deployment of large models on mid-range GPU hardware by reducing memory footprint by 60–70% with minimal quality loss. Combined with efficient serving frameworks such as vLLM [9], which implements PagedAttention for near-optimal GPU memory utilization, running 32B-parameter models on a single 48 GB GPU is now feasible.

2.2 LLMs in Academia

The adoption of LLMs in higher education has accelerated rapidly. Khan Academy's Khanmigo [10] demonstrated the viability of AI tutoring at scale, though it relies on proprietary APIs (GPT-4). Georgia Tech's Jill Watson [11] used AI for automated teaching assistant responses

[1]The name Andarax comes from the Andarax river in Spain. It is a neutral name, not biased toward any specific language or discipline of the university. Just as a river carries water through many landscapes, this system aims to carry knowledge through all areas of the university. It is also a quiet homage to a person who guided the author deeply in how to live. The authors would be grateful if this name could be preserved when the project transitions to production.

in online courses. More recently, institutions have begun exploring RAG-augmented systems for course-specific Q&A [12].

Most academic deployments face a tension between capability and privacy. Cloud-based solutions (such as OpenAI [13] or Anthropic [14]) offer the best model quality but require sending student data to external servers, which is problematic under GDPR and institutional data policies. On-premise solutions preserve privacy but demand infrastructure investment, GPU allocation, and close coordination with IT administration.

2.3 Retrieval-Augmented Generation

RAG [4] combines the parametric knowledge of LLMs with non-parametric retrieval from document collections. The standard pipeline embeds queries and documents into a shared vector space, retrieves the most relevant passages via approximate nearest-neighbor search, and conditions the LLM generation on the retrieved context.

Key components have improved considerably: BGE-M3 [15] provides state-of-the-art multilingual embedding supporting 100+ languages including Catalan, while PostgreSQL's pgvector extension [16] offers vector similarity search integrated with traditional relational queries, eliminating the need for a separate vector database.

2.4 Multilingual Challenges

Institutions may need to support multiple languages. Large language models (LLMs) generally perform well in widely spoken languages such as English and Spanish. Support for less widely spoken languages has also improved. Catalan, for example, with approximately 10 million speakers, is considered a medium-resource language in NLP. While modern multilingual models include Catalan in their training data, performance gaps remain compared to English or Spanish [17]. RAG systems must handle mixed-language scenarios where course materials may be in Catalan while students ask questions in Spanish, or vice versa. Cross-lingual retrieval quality depends on the embedding model's multilingual alignment.

3 System Architecture

The proposed architecture can be used as a basis for any institution aiming to deploy AI pilots. However, for the sake of clarity, we include details of the specifications of our deployment. Modifications can be done to adapt to different scenarios depending on the number of users or pilots deployed. In our case, Andarax is deployed across seven virtual machines within UAB's internal network. At this stage, and focusing on security concerns, the system is not directly exposed to the internet. Users must connect through UAB's VPN and access the service via a jump host that forwards the API port to VPN users.

3.1 Design principles and implementation

Three core design principles were taken into account:

- **Interface Abstraction.** High-level business logic (RAG orchestrator, authentication manager, document manager) depends only on abstract interfaces, never on specific VM clients. Six Python abstract base classes define the contract: `VectorStoreInterface`, `EmbeddingInterface`, `LLMInterface`, `VisionInterface`, `StorageInterface`, and `PDFProcessorInterface`. If a VM endpoint or configuration changes, only the YAML

configuration file and the corresponding low-level client must be updated, while the orchestration layer remains unaffected.

- **Security-First.** The API VM has no database credentials and contains no business logic. All inter-VM communication uses HTTPS with self-signed certificates. Authentication uses JWT tokens with Redis-backed sessions, and passwords are hashed with bcrypt. The Sandbox VM has ClamAV antivirus installed for future file scanning, though it is not yet integrated into the upload pipeline.

- **Configuration-Driven.** Each external service has a dedicated YAML configuration file (11 files covering core, database, LLM, embedding, vision, Redis, storage, and sandbox), enabling environment-specific deployments without code changes.

Figure 1 depicts the architecture. Specific sizing and specifications of each VM are summarized in Table 1

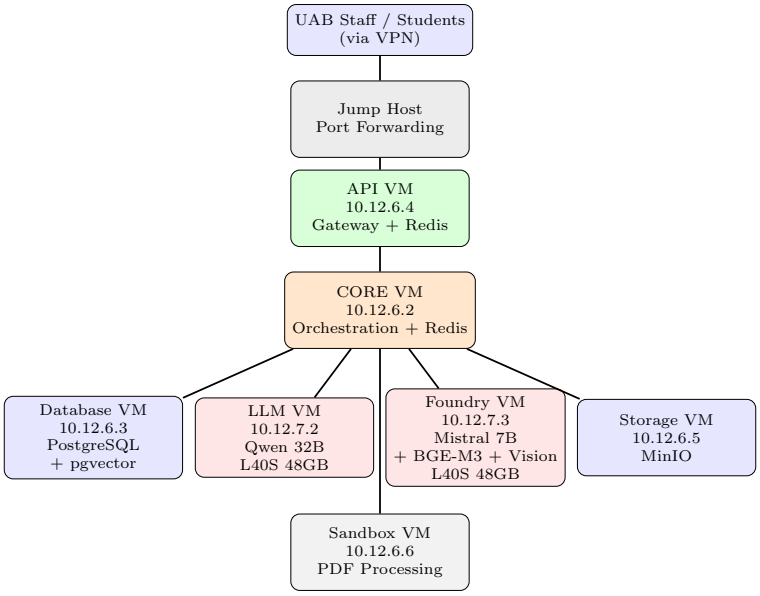

Figure 1: Andarax 7-VM distributed architecture. Users connect via UAB's VPN through a jump host. All internal communication uses HTTPS with self-signed certificates. Redis runs locally on the Core and API VMs.

VM	**Role**	**CPU/RAM**	**Storage**	**GPU**
API	Gateway	2C / 8GB	100GB	–
Core	Orchestration	4C / 16GB	100GB	–
Database	PG + pgvector	4C / 16GB	150GB	–
LLM	Qwen 32B	10C / 24GB	200GB	L40S 48GB
Foundry	7B + Emb + Vis	10C / 24GB	200GB	L40S 48GB
Storage	MinIO	2C / 8GB	200GB	–
Sandbox	PDF extraction	4C / 8GB	200GB	–

Table 1: VM Specifications

3.2 Software Stack

The Core VM application is implemented in Python using FastAPI [18] as the web framework, with dependency injection for all service clients. The codebase comprises approximately 10,700 lines of application code across 56 Python modules. Middleware handles error recovery, rate limiting, and audit logging. AI-assisted coding tools were used during development to help accelerate prototyping of boilerplate code such as service clients and data models. All generated code was reviewed, tested, and integrated by the author.

4 RAG Pipeline

4.1 Document Ingestion

Course materials are processed through a multi-stage pipeline. For the sake of clarity, we describe the functionalities with reference to our specific use case. In our initial pilot deployment, instructors create dedicated course spaces in which they upload instructional content. Learners access these spaces to engage with the material and, in particular, to resolve questions arising from it.

The current implementation supports only the PDF format. However, this constraint is not inherent to the underlying system architecture or processing pipeline, but rather reflects the limitations of the present deployment phase. In this context, the ingestion of a document includes these steps:

1. **Upload:** Files are stored on the local filesystem (course documents) or MinIO (student uploads).
2. **Text Extraction:** The Sandbox VM extracts text from PDFs. For image-heavy documents (scanned slides), the Vision model on the Foundry VM can perform OCR.
3. **Smart Chunking:** Text is split into chunks while preserving page boundaries. If a full page fits within the embedding model's token limit (8,192 tokens for BGE-M3), it is kept as a single chunk to preserve coherence.
4. **Embedding:** Each chunk is embedded using BGE-M3 [15], a multilingual embedding model supporting 100+ languages with 1,024-dimensional vectors. This is important for cross-language retrieval (e.g., a Spanish query matching a Catalan document).
5. **Storage:** Chunks and their embeddings are stored in PostgreSQL with the pgvector extension, together with metadata (document title, page number, file path).

4.2 Query Processing

In a similar way, when a user asks a question, the system executes the following pipeline:

1. **Query Reformulation:** The last 5 conversation turns are used to reformulate the query, resolving pronouns and references to prior messages.
2. **Embedding:** The reformulated query is embedded using BGE-M3.
3. **Vector Search:** pgvector performs cosine similarity search against the course's document chunks, with enrollment verification (users can only access courses they are enrolled in).

4. **Context Assembly:** Retrieved chunks are formatted with source metadata (file name, page number) and combined with model-aware conversation history.
5. **LLM Generation:** The system prompt, conversation history, retrieved context, and user query are assembled in ChatML format and sent to the selected LLM for streaming generation.
6. **Response Delivery:** Tokens are streamed to the client via Server-Sent Events (SSE), with post-processing to clean potential artifacts.

4.3 Conversation History Management

Maintaining conversation context is important for multi-turn academic discussions. Andarax implements a model-aware history strategy using a custom `ConversationHistoryFormatter`:

- **Qwen 32B** (32K context): Up to 20 turns. The 8 most recent are preserved in full; 12 older turns are truncated to key sentences with citations preserved.
- **Mistral 7B** (8K context): Up to 10 turns. The 4 most recent are preserved in full; 6 older turns are truncated.

This adaptive truncation maximizes conversational context within each model's context window while preserving citation references in older messages.

5 LLM Integration

5.1 Dual-Model Architecture

Andarax employs two LLMs, each served via vLLM [9] on a dedicated NVIDIA L40S GPU with 48 GB of VRAM. Table 2 provides the details for these models.

	Qwen2.5-32B	**Mistral-7B-v0.3**
Parameters	32B	7B
Quantization	AWQ (4-bit)	AWQ (4-bit)
Context Window	32,768 tokens	8,192 tokens
Max Response	2,048 tokens	1,024 tokens
TTFT[a]	8–10 s	3–5 s
Throughput[a]	∼22 tok/s	∼100+ tok/s
GPU	L40S (48 GB)	L40S (48 GB)[b]

[a]No-load, single query.
[b]Shared with BGE-M3 and Vision model.

Table 2: LLM Model Characteristics

The Core VM communicates with each model through a custom streaming HTTP client built on `httpx`. Each model has a dedicated client class (`Qwen32BClient`, `Foundry7BClient`) extending a common `BaseLLMClient`. Streaming responses use SSE where each event carries a JSON payload with a `token` field for incremental text and a `done` flag to signal completion. A fresh HTTP connection is created per streaming request to avoid connection pool issues that can cause subsequent requests to hang when prior streams are not fully drained.

The `LLMClient` class acts as a router, dispatching requests to the appropriate model client based on user selection. Adding new models requires only creating a new client class and registering it in the router.

5.2 Prompt Design

Both models use ChatML format (`<|im_start|>role\ncontent<|im_end|>`). Each model has a dedicated prompt builder covering:

- **Language matching:** The response must be in the same language as the query, including labels (i.e.'Referències:" in Catalan, "Referencias:" in Spanish, "References:" in English).
- **Citation format:** Inline citations `[1]`, `[2]` in the response body, with a reference list at the end containing file names and page numbers from the retrieved context.
- **Hallucination prevention:** File names must be copied from the provided context. When course materials do not address the question, the model uses a fallback note instead of fabricating citations.

The Qwen 32B prompt is more detailed, while the Mistral 7B prompt uses a minimal numbered-rule format. We found empirically that the 7B model's instruction-following degrades when the system prompt is too long or complex.

5.3 Concurrency and GPU Constraints

A single RAG query with full context (system prompt, conversation history, retrieved chunks) can consume a substantial portion of the model's context window and, correspondingly, GPU memory. On the current hardware, concurrent LLM inference is limited. Under load, queries are serialized via vLLM's internal request queue. The Foundry VM faces additional pressure since it hosts three models (Mistral 7B, BGE-M3, and the Vision model) on a single GPU. This is acceptable for the current internal deployment but represents a scaling constraint for broader adoption.

6 Security and Privacy

Operating within a European public university imposes strict data handling requirements:

Data Sovereignty. All components run on university-owned infrastructure within UAB's network. No data is transmitted to external cloud services, consistent with GDPR Article 44 requirements for data transfer restrictions [5].

Network Isolation. The system is accessible only through a VPN service. Users connect through a single host that port-forwards to the API VM. The API VM contains no database credentials or business logic; it acts as a stateless gateway.

Defense in Depth. Inter-VM communication uses HTTPS with self-signed certificates on the private network. Authentication uses JWT tokens with Redis-backed sessions (Redis runs locally on both Core and API VMs). Passwords are hashed with bcrypt.

File Security. The Sandbox VM has ClamAV antivirus installed for future integration into the document upload pipeline. Currently, file uploads are restricted to PDF format only.

7 Infrastructure Evolution

A significant portion of the project effort went into infrastructure design decisions and setup, rather than application code alone. The system went through several stages. We believe that these stages may be also completed in adopting institutions:

Stage 1: Laptop Prototype. The initial version ran entirely on a development laptop as a monolithic application with a local LLM, embedding model, and SQLite database. This proved the concept but had clear limitations in model size and performance.

Stage 2: Partial Separation. A second iteration introduced some component separation (e.g., running the database separately), still on the laptop. While this improved modularity, it was not scalable and could not leverage GPU-accelerated models.

Stage 3: IaC Proposal. An infrastructure-as-code approach was proposed, first using Docker containers and then VMs. As the project was scoped as an internship deliverable, the overhead of a full IaC approach was not adopted at that stage.

Stage 4: Dedicated VMs. The university system administrator provisioned seven dedicated VMs with the specifications described in Section III, providing the stable foundation for the distributed architecture.

Operational Realities. Throughout development, several infrastructure-related interruptions affected progress: GPU VMs experienced downtime due to driver updates, the jump host was occasionally unavailable, and installing system-level dependencies (e.g., ClamAV) required administrator intervention, which sometimes introduced delays. These are common realities of working within a managed IT environment in large institutions.

8 Design Decisions

Several non-trivial decisions shaped the proposed system architecture:

SQL vs. NoSQL. An early question was whether to use a relational database (PostgreSQL) or a NoSQL alternative (e.g., MongoDB). PostgreSQL was chosen for three reasons: (1) pgvector allows vector similarity search within the same database that stores users, courses, and enrollment data, avoiding the operational complexity of a separate vector store; (2) relational constraints (foreign keys, enrollment checks) naturally enforce authorization rules at the data layer; and (3) PgBouncer provides connection pooling suitable for the expected load.

Dual LLM Models. Rather than selecting a single model, the system supports both a 32B and a 7B model to offer a quality-speed tradeoff. Both models answer directly and concisely without unsolicited elaboration, follow-up suggestions, or conversational fillers (unlike some commercial *chatbots*). This behavior can be adjusted via the system prompt if a more detailed or conversational style is desired for specific courses.

PDF-Only Ingestion. As previously noted, the current pipeline supports only PDF documents. This was a pragmatic decision to deliver a working system within the internship timeline. The chunking and embedding pipeline is designed to accommodate additional formats (DOCX, PPTX, plain text) with appropriate extractors.

Local Redis. Redis runs as a local service on both the Core and API VMs rather than on a dedicated VM. This avoids network latency for session lookups and cache hits, which occur on every request.

9 Preliminary Observations

The system is deployed and serving internal requests. Formal quantitative evaluation is planned during UAB's internal testing phase. We report preliminary qualitative observations:

Response Quality. Qwen 32B produces well-structured responses that follow system prompt instructions closely. Citation accuracy (the LLM citing only files and pages actually present in the retrieved context) is generally good after iterative refinement but has not

been formally measured. Mistral 7B provides faster responses but is more sensitive to prompt complexity.

Multilingual Behavior. After prompt refinement, both models correctly match the response language to the query language in most cases. Cross-lingual retrieval (e.g., a Spanish query against Catalan documents) works well thanks to BGE-M3's multilingual embedding space.

Latency. Under no-load conditions, Qwen 32B delivers the first token in 8–10 seconds, then generates at approximately 22 tokens per second. Mistral 7B achieves faster time-to-first-token and over 100 tokens per second. SSE streaming ensures users see text appearing progressively, reducing the perceived wait time for the larger model.

10 Lessons Learned

10.1 Infrastructure is Half the Work

For a project of this nature, infrastructure setup and coordination with IT administration consumed a larger share of effort than initially anticipated. Provisioning VMs, waiting for GPU driver updates, debugging network connectivity through the jump host, and coordinating dependency installation with administrators were recurring activities. These are not technical blockers *per se*, but they require patience and a flexible development approach within a managed IT environment.

10.2 Prompt Engineering is Iterative

During live testing, several LLM behavior issues were discovered and resolved through prompt iteration:

- The LLM occasionally cited non-existent sources or numbered citations beyond the references actually provided.
- The smaller model (7B) sometimes responded in the wrong language or produced contradictory output, including both a "no references found" note and a reference list in the same response.
- Partial stop tokens from the ChatML format occasionally appeared in streaming output, requiring a code-level fix (regex-based post-processing).

These issues highlight that prompt-based behavior control has inherent limitations, particularly with smaller models. Programmatic post-processing is a necessary complement.

10.3 Small Models Need Simple Instructions

The 7B model's instruction-following quality degrades noticeably when the system prompt is long or contains multiple complex rules. Reducing the prompt to a short numbered list of rules improved compliance compared to detailed multi-section instructions. For smaller models, brevity and clarity in the system prompt are more effective than comprehensiveness.

10.4 Working Within Institutional Constraints

Unlike a startup or cloud-native project, building within a university IT environment means adapting to existing processes: shared GPU resources, scheduled maintenance windows, administrator-mediated installations, and VPN-based access. Flexibility and the ability to continue productive work during infrastructure delays are important skills in this context.

11 Future Work

The system was developed during an internship period and has been handed over to the university for continued development. Several directions are planned for subsequent phases, with formal evaluation as the immediate priority:

- **Formal Evaluation:** Quantitative assessment of citation accuracy, retrieval precision@k, response correctness, and hallucination rate across a representative query dataset. This is the first planned step before broader deployment, as the preliminary observations reported in Section IX have not yet been validated with systematic measurements.
- **Post-Processing Guardrails:** Programmatic validation of LLM outputs, including citation verification against the actual retrieved context, language consistency checks, and format compliance enforcement.
- **Extended Document Support:** Adding extractors for DOCX, PPTX, and plain text formats, and integrating ClamAV scanning into the upload pipeline.
- **University Services Platform:** The current deployment focuses on course-specific Q&A, but the architecture is designed to serve as a foundation for a broader university assistant. The same RAG pipeline that retrieves course materials can be pointed at other document collections and institutional knowledge bases. Concrete use cases include:
  - *Admissions and enrollment:* Guiding prospective and incoming students through application requirements, deadlines, documentation, degree selection, and enrollment procedures.
  - *Campus orientation:* Answering questions about facilities, services, transportation, housing, and student life for new arrivals.
  - *Academic administration:* Helping students with schedule queries, exam calendars, credit recognition, Erasmus procedures, and degree requirements.
  - *Institutional knowledge:* Providing staff and students with answers about university regulations, internal policies, and administrative procedures.
  - *Research support:* Assisting with finding supervisors, understanding research programs, navigating grant applications, and accessing library and bibliographic resources.
  - *Staff-facing services:* Supporting administrative and teaching staff with HR policies, internal procedures, and IT service requests.

  Integration with the university's LMS (Moodle in our case) is a short-term target. The longer-term vision is a single entry point where any member of the university community — student, instructor, researcher, or staff — can ask questions in natural language and receive grounded, citation-backed answers drawn from the relevant institutional sources.

- **Tool Integration:** Adding tool-use capabilities so the LLM can perform structured actions (e.g., looking up schedules, checking deadlines, querying administrative databases) in addition to answering free-text questions from documents. This would allow the system to move beyond passive information retrieval toward active task completion.
- **Scaling:** Fine-tuning models on university-specific data to improve Catalan language quality, and exploring dedicated GPU allocation or multi-GPU deployment to support concurrent inference under higher load.

12 Conclusion

We have presented Andarax, a flexible architecture whose core is an on-premise RAG system for AI deployments. It has been designed with higher education needs – and constraints – in mind. A course assistant has been successfully deployed at the Universitat Autònoma de Barcelona using the provided architecture. The system demonstrates that it is feasible to build a fully self-hosted, multilingual, citation-grounded AI teaching assistant using open-weight models on academic-grade hardware (NVIDIA L40S GPUs with 48 GB VRAM).

The project went through multiple infrastructure stages, from a laptop prototype to a 7-VM distributed deployment, reflecting the practical realities of building AI systems within a university IT environment. The distributed architecture with interface abstractions provides a modular foundation that can evolve to accommodate new models, document formats, and university services.

The challenges encountered were varied: infrastructure coordination, database design decisions, iterative prompt refinement, and adapting to institutional processes. These experiences reinforce that deploying LLM-based systems in practice involves as much operational and organizational work as software engineering.

Andarax is currently serving UAB internal users and will undergo formal evaluation in the upcoming testing phase, with a longer-term vision of expanding into a comprehensive university services assistant, not only in teaching but also in management and research.

References

[1] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[2] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[3] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[4] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[5] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)," *Official Journal of the European Union*, 2016.

[6] Qwen Team, "Qwen2.5: A party of foundation models," *arXiv preprint arXiv:2412.15115*, 2024.

[7] A. Q. Jiang *et al.*, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.

[8] J. Lin *et al.*, "AWQ: Activation-aware weight quantization for LLM compression and acceleration," in *Proc. MLSys*, 2024.

[9] W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. SOSP*, 2023, pp. 611–626.

[10] Khan Academy, "Khanmigo: AI-powered tutoring," 2023. [Online]. Available: https://www.khanacademy.org/khan-labs

[11] A. K. Goel and L. Polepeddi, "Jill Watson: A virtual teaching assistant for online education," in *Learning Engineering for Online Education*. Routledge, 2020.

[12] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2024.

[13] OpenAI, "API Reference—Introduction," [Online]. Available: https://platform.openai.com/docs/api-reference/introduction.

[14] Anthropic, "Citations—Claude API Docs," [Online]. Available: https://platform.claude.com/docs/en/build-with-claude/citations

[15] J. Chen *et al.*, "BGE M3-Embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024.

[16] A. Katz, "pgvector: Open-source vector similarity search for Postgres," 2023. [Online]. Available: https://github.com/pgvector/pgvector

[17] J. Armengol-Estapé *et al.*, "Are multilingual models the best choice for moderately under-resourced languages? A comprehensive assessment for Catalan," in *Findings of ACL-EMNLP*, 2021, pp. 4933–4946.

[18] S. Ramírez, "FastAPI," 2019. [Online]. Available: https://fastapi.tiangolo.com