Journal of Engineering Issues and Solutions

A Semantic Retrieval and Generation Framework for Alumni Intelligence Using LLaMA 3

Bipin Acharya[1*], Sudip Koirala[2], Rajina Chaudhari[3], Krishna Gautam[4], Bishwas Pokharel[5]

[1, 2, 3, 4] *Kathford International College Of Engineering and Management*
[5] *Government Of Nepal*
**Corresponding email: bipinacharya284@gmail.com*
Submitted: April 13, 2025; Revised: June 15, 2025; Accepted: June 21, 2025

https://doi.org/10.3126/joeis.v4i1.81598

Abstract

The study focused on the development of the *Alumni Intelligence System*, a scalable and intelligent solution designed to preserve and enable access to alumni project data within an academic institution. The system addresses significant challenges for limited and outdated historical records, with 155 usable records by integrating LLaMA 3, a large language model, with Retrieval-Augmented Generation (RAG). A vector database is utilized to hold the vector embeddings created from structured alumni project data. Upon receiving a query, the system performs similarity search to retrieve relevant contexts, which are then used to generate accurate and context-aware responses via LLaMA 3. Performance evaluation using metrics such as cosine similarity (0.78), precision (0.79), and BLEU score (0.64) indicates the system's effectiveness in retrieving and generating relevant academic information. The proposed system demonstrates the potential of LLM-powered architectures in enhancing academic knowledge retention, research support, and alumni data utilization.

Keywords: *Academic Knowledge Management, Alumni Intelligence System, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Semantic Search, Vector Embeddings*

1. Introduction

Since alumni provide current students with opportunity, networking, and mentorship, they are essential to the development and standing of educational institutions. However, maintaining meaningful engagement with alumni remains a challenge, as traditional communication methods are often ineffective and impersonal. While institutions possess valuable data on alumni, such as academic backgrounds, research work, and career paths, the information is frequently underutilized. To address the issues the study focuses on, the Alumni Intelligence System, which leverages Llama [4], an open-source large language model developed by Meta, integrated with Retrieval-Augmented Generation (RAG). Large language models (LLMs) have become widely used as a result of recent developments in generative artificial intelligence across various

domains, with models like ChatGPT, DALL·E, and Midjourney showcasing the capabilities of text and image generation. Generative AI models are evolving toward multimodal foundation models capable of learning from multiple data types simultaneously. Among open-source LLMs, Meta's Llama and Falcon have gained attention due to their efficient performance and scalability, emphasizing learning volume over sheer model size [1]. Despite challenges such as hallucinations and prompt sensitivity, LLMs have proven effective for diverse tasks including writing, debugging, and reasoning [2]. Tools like LangChain have simplified the development of LLM-based applications, particularly for domain-specific use cases [3]. Llama models utilize dense transformer architectures, SwiGLU activation, rotary positional embeddings, and have been trained on curated datasets of up to 2 trillion tokens from sources like CommonCrawl, C4, Wikipedia, and StackExchange. These models achieve strong benchmark results and, in many cases, outperform larger models like GPT-3 and PaLM while being more resource-efficient [4].

Reinforcement learning from human feedback (RLHF) has been used to improve Llama 2-Chat, optimized for dialogue tasks and surpasses many open-source and proprietary models in reasoning and knowledge-intensive benchmarks [5]. Studies show that Llama 2 and 3 models perform competitively in high-performance computing code generation tasks but may require prompt tuning for optimal results [6]. Llama 3 offers enhanced adaptability and nuanced responses compared to GPT-4o and PaLM 2, making it particularly suitable for personalized guidance [7], [8]. When integrated with Retrieval-Augmented Generation (RAG), Llama models significantly improve answer precision and factual consistency even in sparse data environments, sometimes outperforming larger models like GPT-4 without RAG [9], [10]. Llama 3.1 405B, the first open model with 405 billion parameters, rivals top AI models in knowledge, math, tool use, and multilingual translation, leading the chart to become a futuristic model of LLM [11]. The positional information is efficiently utilized by the Rotary Position Embedding (RoPE) technique. In particular, the suggested RoPE uses a rotation matrix to record the absolute location while also explicitly incorporating the relative position dependency into the formulation of self-attention.[12]. This study evaluates pruning techniques on the Llama-3 8B model for on-device AI applications, identifying a 53% sparsity threshold that preserves output quality with improved accuracy over Llama-2 7B, and proposes strategies to enhance performance in sparse language models[13]. The all-MiniLM-L6-v2 model is the backbone of the system. The all-MiniLM-L6-v2 model features 6 transformer encoder layers with fewer hidden units than BERT-base, resulting in a lightweight, faster model optimized for multilingual NLP tasks [14]. LLaMA3 demonstrates performance comparable to leading models like GPT-4 across various benchmarks and includes publicly available pre-trained and post-trained versions, along with the Llama Guard 3 for safety [15]. Llama 3 was instructed to communicate with users in a particular person's language style without changing its fundamental settings, resulting in a text-based conversational AI that mimics the person's language style [16]. With one key and value head for each subgroup of query heads, we suggest grouped-query attention (GQA), an interpolation between multi-head and multi-query attention [17]. While pretrained models can be modified for a range of natural language generation tasks, LLaMA2 tuned models are designed for assistant-like chat. Meanwhile, LLaMA3 instruction tuned models are designed for assistant-like chat, and developers are free to modify LLaMA 3 models for languages other than English as long as they abide by the Acceptable Use Policy and the LLaMA 3 community License [18],[19]. By identifying and isolating outlier weights that cause particularly large quantization errors and storing them in higher precision while compressing all other weights to 3–4 bits, Sparse-Quantized Representation (SpQR), a new compressed format and quantization technique, allows for the first time near-lossless compression of LLMs across model scales. For highly-accurate LLaMA and Falcon LLMs, this technique achieves relative accuracy losses of less than 1% in perplexity[20]. Additionally, the Llama 3.1 model collection facilitates the use of model outputs to enhance other models, such as distillation and synthetic data generation[21].

The model used in this study aims to enhance response accuracy even with limited data. This study aims to preserve alumni project records, provide intelligent access to past contributions, and support students in selecting and analyzing potential projects. By creating a structured and accessible database, the system not only strengthens alumni relations but also enhances academic collaboration and institutional knowledge sharing.

2. Materials and Methods

2.1 Data Collection

The Alumni Intelligence System comprises several integrated components that enable the collection, processing, and retrieval of alumni project data through natural language queries. The process begins with data acquisition from multiple sources, including college records, library archives, and Google Forms, focusing on key elements such as student names, project titles, abstracts, and conclusions. The data collection focused on compiling a structured dataset containing information about past student or alumni projects. A total of 155 project records were gathered from the college's archives, primarily sourced through departmental documentation and alumni outreach efforts. The dataset was organized in a comma-separated values (CSV) format, allowing for easy processing and integration into machine learning pipelines. Each row in the dataset corresponds to a single project and includes the following attributes: names of up to four project members (with the fourth member's data optional and missing in some entries), project title, project abstract, and project conclusion. The Project Member Name 1–4 fields identify the contributors; Project Title gives the name of the work; Project Abstract offers a summary of the project's purpose, methodology, and scope; and Project Conclusion highlights the outcomes and significance of the work. These attributes collectively provide a comprehensive view of each project, which is essential for generating meaningful responses using the LLM and for preserving valuable academic contributions.

2.2 Block Diagram

After the collection of the data, it is then structured into nodes, where each node contains both text and associated metadata, enabling efficient categorization and semantic retrieval. The structured text is converted into vector embeddings using the MiniLM-L6-v2 model, a lightweight and high-performing transformer-based language model optimized for semantic similarity tasks. These embeddings are stored in a FAISS vector database, which allows for rapid and context-aware retrieval of relevant data based on vector similarity rather than exact keyword matching. When a user submits a query, it is embedded and compared against stored vectors to retrieve the most semantically relevant content. The system then generates a coherent, context-sensitive response that directly answers the user's question. This architecture ensures fast, accurate, and scalable performance, providing an effective solution for intelligent academic data management and institutional knowledge retention.

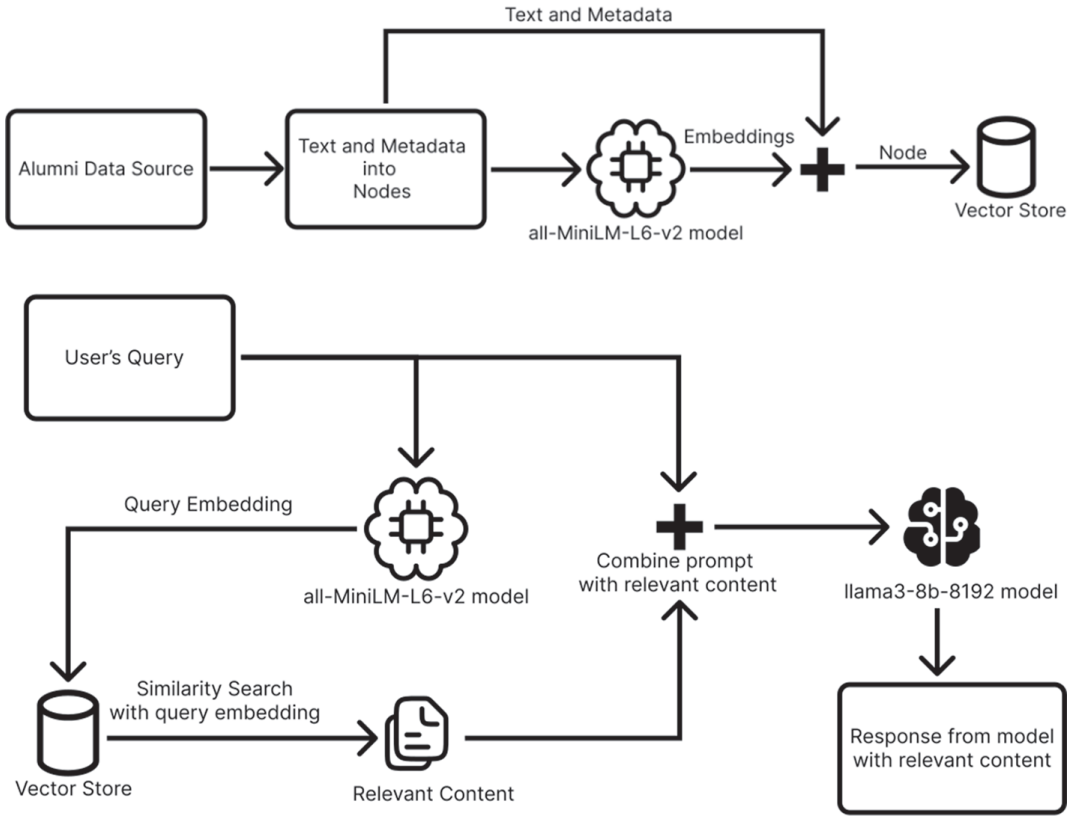

Figure 1. System Block Diagram of LLaMA

2.3 Llama Architecture

The LLaMA (Large Language Model Meta AI) architecture is utilized for generating responses to user queries based on contextually relevant information retrieved from the vector store. Developed by Meta AI, LLaMA is a highly efficient transformer-based language model optimized for high-quality text generation with reduced computational overhead compared to larger models like GPT-4. It incorporates several advanced features such as Grouped Multi-Query Attention (GQA), RMS Normalization, and SwiGLU activation to enhance scalability and performance. The embedding process transforms words into high-dimensional vectors, enabling the model to grasp semantic relationships and interpret user queries with contextual understanding rather than mere keyword matching. RMS Normalization ensures balanced input by preventing dominant words from skewing the model's internal computations, thereby maintaining consistent quality in output. LLaMA's self-attention mechanism, enhanced by a Key-Value (KV) Cache, allows it to process words in parallel, store prior context efficiently, and accelerate response generation by avoiding redundant computations. Additionally, the Feed Forward SwiGLU activation function enables the model to capture complex patterns by filtering irrelevant information and retaining crucial features, ensuring that the final responses are both accurate and contextually meaningful. This architecture enables LLaMA to deliver accurate and contextually relevant responses, providing users with high-quality information based on their queries. LLaMA is an effective tool for natural language processing problems because it combines sophisticated normalization algorithms, self-attention mechanisms, and effective embeddings. The methodology employed in developing the Alumni Intelligence System follows a systematic approach encompassing data collection, system design, embedding generation, and response processing. Each component—from sourcing and structuring alumni project

data to transforming it into vector embeddings and retrieving relevant content—contributes to the overall performance of the system. Leveraging the strengths of both LLaMA and MiniLM-L6-v2, the system establishes a robust, scalable, and intelligent framework capable of delivering timely and accurate responses, thus supporting effective knowledge management and information retrieval in academic environments.

Here, Swish function is represented mathematically as:

$$\text{Swish}(x) = x * \sigma(\beta x) \quad (1)$$

where:

• $\sigma(\beta x)$ represents sigmoid function applied to βx and β is a learnable parameter[12].

Here, GLU function can be represented mathematically as:

$$\text{GLU}(x) = (W x + b) \otimes \sigma (V x + c)$$

• w, b, v, c are a learnable parameter.

Combinely, Feed Forward swiGLU can be represented mathematically as:

$$\text{SwiGLU}(x) = (W x + b) \cdot \text{Swish}(V x + c) \quad (2)$$

where $\text{Swish}(x) = x \cdot \sigma(x)$, $\sigma(x) = \frac{1}{1+e^{-x}}$[12]

Finally, the **Linear Layer** and **Softmax Function** are applied to convert the processed numerical data into human-readable text. The **Softmax** function selects the most probable next word based on the context, ensuring that the generated response is both coherent and contextually appropriate.

The Softmax function is defined as:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{Z_j}} \quad (3)$$

Where[12];

- Z_i represents the logit for the ith class
- K represents the total number of classes
- e^{z_i} represents the exponential of logit
- $\sum_{j=1}^{K} e^{Z_j}$ represents the sum of exponentials across all classes

Self-Attention (Scaled Dot Product)

Self-Attention, specifically the Scaled Dot-Product Attention, is a fundamental mechanism in transformer models. Three input matrices are used in its operation: the Value matrix (V), the Key matrix (K), and the Query matrix (Q). All of these matrices are produced using learnt linear transformations from the input sequence. The numerical description of the attention mechanism is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T / \sqrt{dk}\right) \cdot V \quad (4)$$

Here, Q and K are matrices with the shape (sequence_length × d_k), while V has the shape (sequence_length ×

d_v). The dot product QK^T measures the similarity between the query and key vectors. To avoid having too big values that can result in tiny gradients after using the softmax function, this result is then scaled by the square root of the key size dk[12]. These scaled dot products are then transformed into a probability distribution via the softmax function, which essentially allocates attention weights. The output of the attention mechanism is then generated by computing a weighted sum of the value vectors using this distribution.

Multi-Head Attention

Multi-Head Attention is an extension of the self-attention mechanism that allows a model to jointly attend to information from different representation subspaces at different positions. Multi-Head Attention executes several attention operations, or "heads," concurrently as opposed to executing a single attention function. The input queries (Q), keys (K), and values (V) are transformed by each head using a unique set of learnt projection weights. The following is a mathematical description of the process as:

$$MultiHead(Q,K,V) = Concat(head1, \dots\dots\dots, headh)W^o \quad (5)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (6)$$

In this formulation, each headi is an individual attention head that performs scaled dot-product attention using linearly projected versions of Q, K, and V. The projections are done using learnable weight matrices W_i^Q, W_i^K and W_i^V for queries, keys, and values, respectively. Once all attention heads are computed, their outputs are concatenated along the feature dimension using Concat(), and the result is linearly transformed with the output projection matrix W^O[12]. This mechanism allows the model to capture various types of relationships and interactions in the data more effectively than using a single attention head.

Rotary Positional Embedding (RoPE)

Rotary Positional Encoding (RoPE) encodes position by rotating embedding vectors based on their position index instead of adding positional embeddings. It uses a frequency term:

$$\theta_i = 10000^{\frac{-2t}{d}} \quad (7)$$

where i is the dimension index and d is the vector's total dimensionality. The position-aware embedding is computed as:

$$RoPE(x, p) = rotate(x, \theta.p) \quad (8)$$

In this case, rotate(x, θ·p) applies a dimension-wise rotation, where x is the input vector and p is the token's location. RoPE enhances relative position handling and performs well in longer sequences[12].

RoPE with Linear Attention

One way to express self-attention is as follows:

$$Attention(Q,K,V)_m = \frac{\sum_{n=1}^{N} \sin(q_m,k_n)v_n}{\sum_{n=1}^{N} sim(q_m,k_n)} \quad (9)$$

In standard self-attention:

$$sim(q_m, k_m) = e^{(\frac{q_m k_m}{\sqrt{d}})} \quad (10)$$

This computation requires quadratic complexity $O(N^2)$ due to pairwise operations[12].

linear attention reformulates this as:

$$Attention(Q,K,V)_m = \frac{\sum_{n=1}^{N} \phi(k_n) v_n^T \phi(q_m)}{\sum_{n=1}^{N} \phi(k_n)^T \phi(q_m)} \quad (11)$$

Where ϕ (·) ϕ(·) is typically a non-negative function, e.g. ϕ(x)=ELU(x)+1

To incorporate RoPE into linear attention, we apply a rotation matrix R Θ , m R Θ ,m to encode positional information while preserving vector norms[12]. The RoPE-enhanced linear attention becomes:

$$Attention(Q,K,V)_m = \frac{\sum_{n=1}^{N} \left(R_{\theta,m}\phi(q_m)\right)^T (R_{\theta,m}\phi(k_n)) v_n}{\sum_{n=1}^{N} \phi(q_m)^T \phi(k_n)} \quad (12)$$

Layer Normalization

Layer Normalization (LayerNorm) standardizes the input tensor x across its features to stabilize and accelerate training. It is defined as:

$$LayerNorm(x) = \gamma . \frac{(x-\mu)}{\sqrt{\epsilon+\sigma^2}} + \beta \quad (13)$$

Here, μ is the mean and σ^2 is the variance of x. A small constant ε ensures numerical stability. The normalized output is then scaled and shifted using learnable parameters γ (scale) and β (bias). This helps the model adaptively adjust the normalized values during training (12)

Causal Language Modeling Loss

Causal Language Modeling Loss measures how well a model predicts the next token in a sequence, given all previous tokens. It is defined as:

$$L = -\sum_{t=1}^{T} logP(x_t | x_1, x_2, \dots\dots\dots, x_{t-1}) \quad (14)$$

Here, x is the token at position t, and the model is trained to maximize the probability of each token given its left context[12]. The loss L is the negative log-likelihood summed over all positions, encouraging the model to assign high probabilities to the correct next tokens. This setup ensures the model generates text in a left-to-right, causal manner.

Residual Block (Transformer Layer)

The Transformer block with residual connections is defined as:

x' = x + Block(x) (15)

where,

Block(x) = FFN(Layer Norm(Self Attention(Layer Norm(x))))

Here, x is the input. The input is normalized, passed through self-attention, normalized again, then through a feed-forward network[12]. The output is added back to the original input via a residual connection. This design improves training stability and gradient flow.

KV cache

Partial Attention Computation (Chunk-First Phase)

In this step, attention is calculated over several sequences over shared prefix chunks. The attention

computation works as follows given a query tensor slice Qi:j R(j-i) d, key and value tensors K(C), V(C) ∈ R(cd) for a chunk C:

1. Compute attention weights:

$$W^{(C)} = Q_{i:j}.(K^{(c)})^{T} \quad (16)$$

2. For numerical stability, compute the row-wise maximum:

$$m^{(c)} = \max(W^{(c)}, axis = 1) \quad (17)$$

3. Apply softmax scaling:

$$E^{(c)} = e^{(W^{(c)} - m^{(c)})} \quad (18)$$

4. Compute the normalization factor:

$$n^{(c)} = \sum E^{(c)}$$

5. Compute the output:

$$o^{(c)} = E^{(c)}.V^{(c)} \quad (19)$$

This chunk-first approach allows batch processing of shared prefixes, improving memory and computation reuse[12].

Merging Partial Attention Results (Sequence-First Phase)

In this phase, the partial attention outputs are merged for each sequence. Let $O^{(C)}$ be the output from chunk C, and m_i, n_i, o_i be the running maximum, normalization factor, and output for sequence i. The merged results are computed as:

6. Compute scaling factors for merging:

$$x^{(c)} = e^{(m^{(c)} - \max(m^{(c)}, m_i))} \quad (20)$$

$$y^{(c)} = e^{(m_i - \max(m^{(c)}, m_i))} \quad (21)$$

7. Merge the outputs:

$$O_i \leftarrow x^{(c)}.o^{(c)} + y^{(c)}.o_i \quad (22)$$

8. Merge the normalization factors:

$$m_i \leftarrow \max(m^{(c)}, m_i) \quad (23)$$

9. Update the maximum score:

$$m_i \leftarrow \max(m^{(c)}, m_i) \quad (24)$$

This two-phase design, chunk-first followed by sequence-first, enables efficient KV reuse and accurate softmax computation across shared and individual segments of sequences[12].

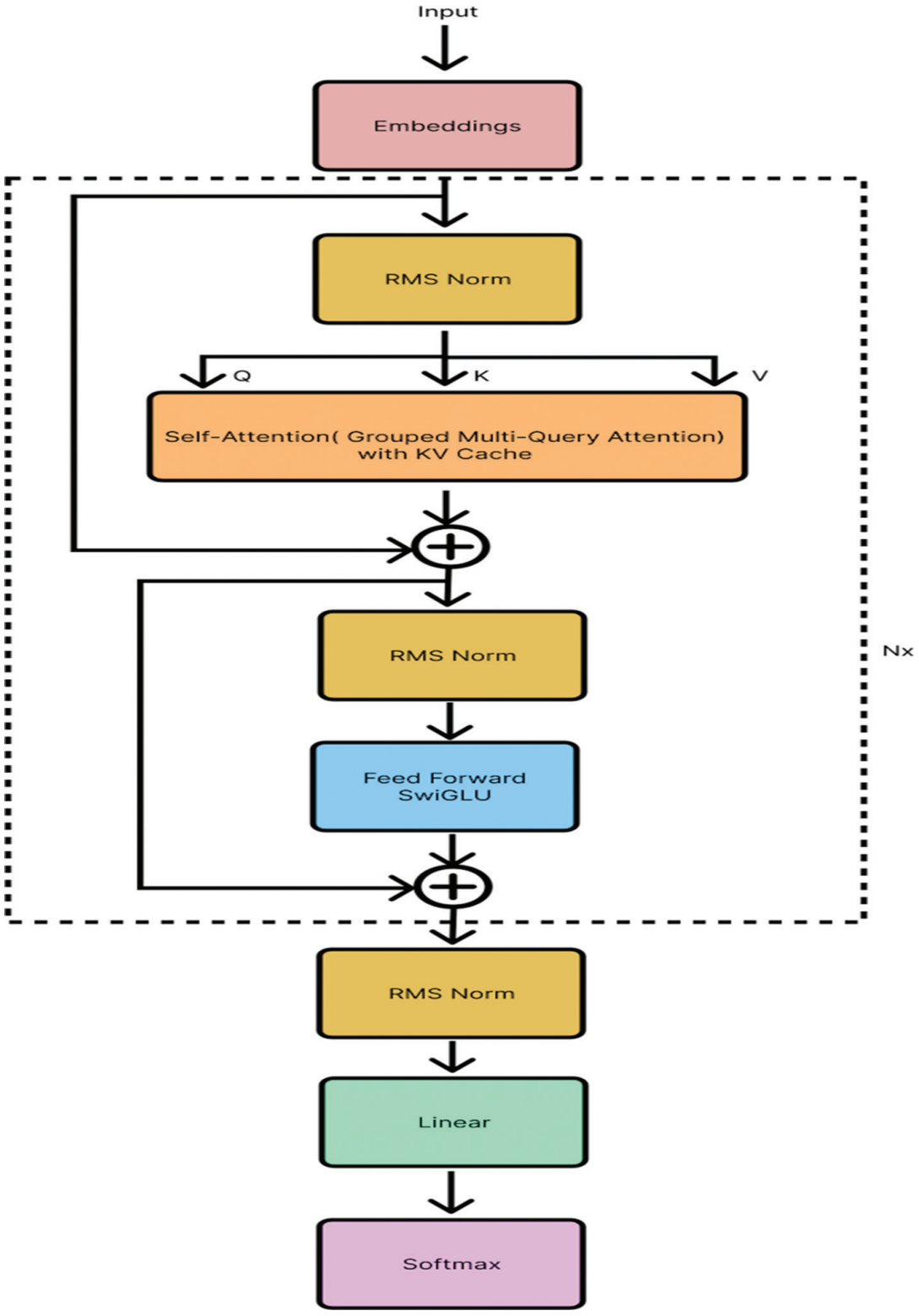

Figure 2. LLaMa Architecture

2.4 Tools and Applications

For the development of our Alumni Intelligence System, a comprehensive set of tools and technologies was employed across various stages of design, implementation, and deployment, ensuring functionality, scalability, and user experience. Python served as the primary programming language due to its simplicity and extensive library support, with TensorFlow and Keras facilitating deep learning tasks and NumPy enabling efficient numerical computations. The backend was built using FastAPI, a high-performance web framework that provided asynchronous capabilities, automatic request validation, and interactive API documentation via Swagger UI. On the frontend, Next.js was chosen for its React-based architecture, supporting server-side rendering and static site generation, which enhanced performance and responsiveness. Pandas was crucial for structured data management and preprocessing, while PyTorch supported deep learning experiments with dynamic computation graphs and GPU acceleration. The integration of Hugging Face's MiniLM-L6-v2 model allowed for high-quality embeddings and semantic search functionality, significantly improving the system's ability to interpret and respond to user queries. Deployment was streamlined through Vercel, offering seamless integration with Next.js for automated deployments and global content delivery. Additionally, Figma aided in designing user interfaces and system architecture prototypes, while GROQ was explored for structured data querying in content-rich applications. Collectively, this robust technology stack enabled the development of a responsive, intelligent, and scalable alumni information system.

3. Results and Discussion

Python, Pandas, GROQ, and Next.js were used in the development of the planned Alumni Intelligence System, which also included a Retrieval-Augmented Generation (RAG) mechanism and the LLaMA3 language model. The dataset used to train the system included roughly 155 final-year student projects that were gathered mostly from the Computer and Electronics faculties via online surveys, college records, and library archives. The primary focus of the dataset was project-based data which formed the backbone for generating responses. The core objective was to enable the system to retrieve relevant information and generate coherent responses using stored alumni project data. This was accomplished using vector embeddings for efficient retrieval and the LLaMA3 model for language generation. The system successfully demonstrated its ability to generate alumni information in response to user queries based on the indexed project data.

To evaluate system performance, several retrieval and generation metrics were employed. The outcomes are summarized below:

Table 1: Metrics Comparison

Metric	Description	Average Value (Approx.)
Hit Rate@5	Measures the frequency of the correct result appearing in the top 5 retrieved items(Higher is better)	0.90
Avg Euclidean Distance	Indicates the semantic distance between query and retrieved vectors(Lower is better)	0.19
ROUGE-L	Evaluates text overlap based on longest common subsequence(Higher is better)	0.74
Cosine Similarity	Measures semantic similarity between query and retrieved items(Higher is better)	0.78
Mean Average Precision (MAP)	Aggregates precision across all relevant ranks(Higher is better)	0.73
Precision@5	Proportion of relevant items among the top 5 results. (Higher is better)	0.79
BLEU Score	Measures the quality of generated text based on n-gram overlap with reference text.	0.64

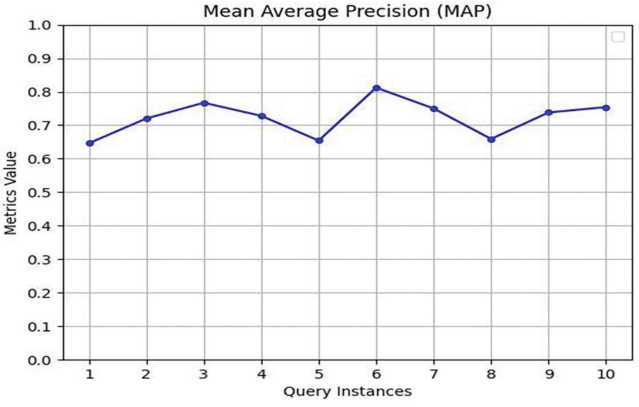

Figure 3: Cosine Similarity between query and top result vectors

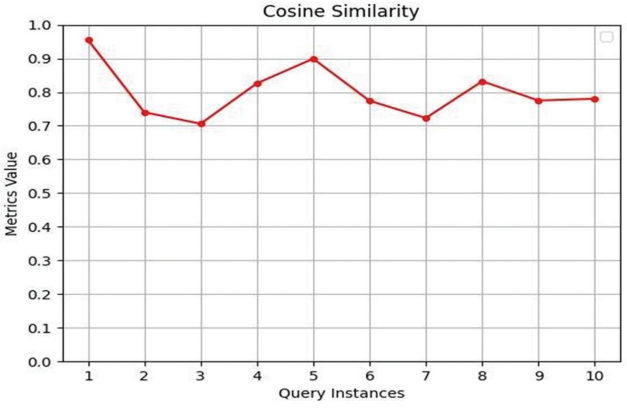

Figure 4: MAP over all queries

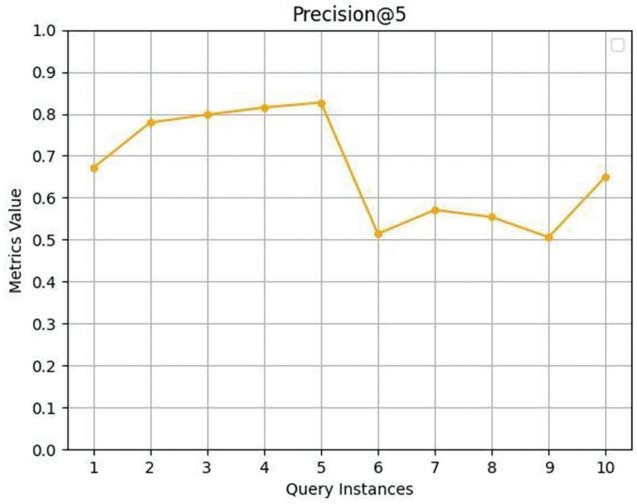

Figure 5: Precision for retrieved results per query

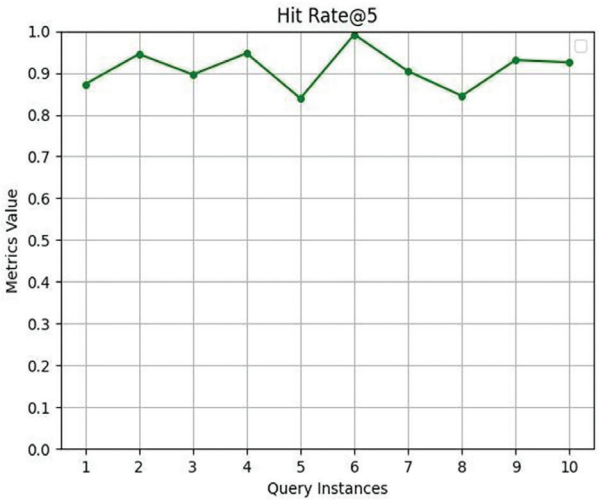

Figure 6: Hit Rate Of Queries

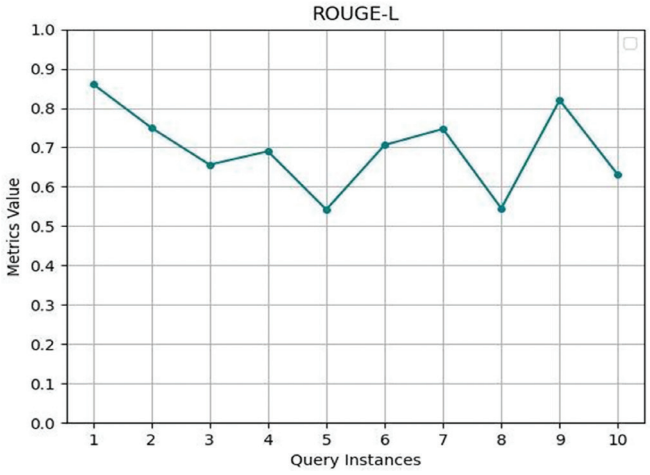

Figure 7: Average Euclidean Distance between query and retrieved vectors

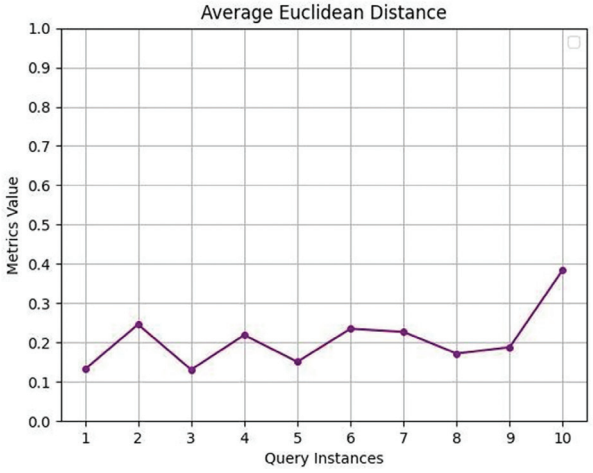

Figure 8: ROGUE-L score between generated and reference texts

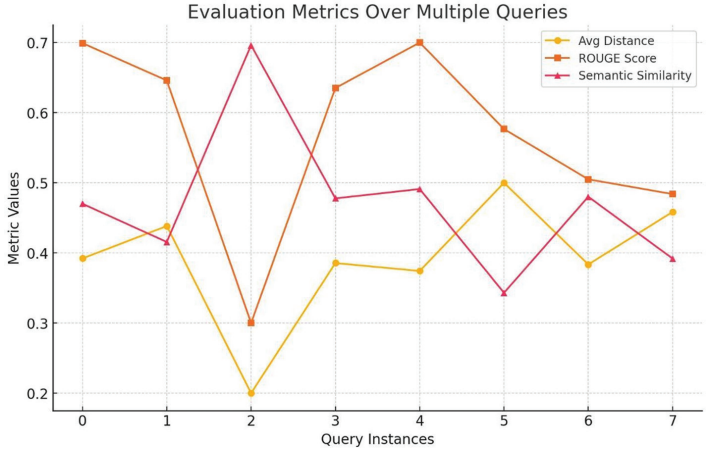

Figure 9: Evaluation metrics

These evaluation results reveal strong system performance across all dimensions. The Cosine Similarity score of 0.78 indicates high semantic alignment between queries and retrieved embeddings. A MAP value of 0.73 confirms consistent relevance across search results. The Precision@5 of 0.79 demonstrates that the majority of the top five results are contextually appropriate. The Precision@5 of 0.79 demonstrates that the majority of the top five results are contextually appropriate. The Hit Rate@5 of 0.90 suggests that in 90% of cases, at least one relevant document was found among the top five results, providing users with a high likelihood of finding useful information quickly. Meanwhile, the Average Euclidean Distance of 0.19 further confirms the semantic closeness of retrieved vectors, supporting the effectiveness of the embedding approach. In terms of response quality, the system achieved a ROUGE-L score of 0.74, indicating significant content overlap between the generated response and reference documents. The BLEU Score of 0.64, while slightly lower, still reflects a reasonable degree of fluency and accuracy in text generation, with minor opportunities for improvement in stylistic diversity and language variability. Overall, the findings support the viability and efficiency of integrating big language models with vector retrieval for the management of alumni data. The system meets its goal of providing intelligent, accurate, and context-aware responses to alumni-related queries, proving its potential for integration into educational or institutional knowledge systems.

4. Conclusions

In conclusion, the development and evaluation of the Alumni Intelligence System have demonstrated the effectiveness of integrating modern natural language processing techniques, specifically the LLaMA3 language model and Retrieval-Augmented Generation (RAG), for alumni data management. The system effectively uses vector embeddings to enable efficient retrieval and context-aware text generation, giving users accurate, pertinent, and cohesive answers to alumni-related queries. The system was trained using a dataset of 155 final-year student projects. The evaluation metrics highlight the system's strong performance, with particularly high results in Cosine Similarity (0.78), Precision@5 (0.79), and Hit Rate@5 (0.90), validating the system's ability to retrieve and generate meaningful information. While there is room for improvement in response fluency, as indicated by the BLEU score of 0.64, the overall results underscore the potential of the system for scalable, intelligent knowledge management in academic institutions. This research showcases the power of combining vector-based retrieval with large language models to enhance the accessibility and usability of alumni project data, making it a valuable tool for academic and institutional environments.

References

Jeong, C. (2023). A study on the implementation of generative ai services using an enterprise data-based llm application architecture. *arXiv preprint arXiv:2309.01105*.

Topsakal, O., & Akinci, T. C. (2023). Creating large language model applications Utilizing LangChain: A primer on developing LLM apps fast. *International Conference on Applied Engineering and Natural Sciences*, *1*(1), 1050–1056. https://doi.org/10.59287/icaens.1127

Chataut, S., Do, T., Gurung, B. D. S., Aryal, S., Khanal, A., Lushbough, C., & Gnimpieba, E. (2024). Comparative study of domain driven terms extraction using large language models. *arXiv preprint arXiv:2404.02330*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Valero-Lara, P., Huante, A., Lail, M. A., Godoy, W. F., Teranishi, K., Balaprakash, P., & Vetter, J. S. (2023). Comparing Llama-2 and GPT-3 LLMs for HPC kernels generation. *arXiv preprint arXiv:2309.07103*.

Luo, X., O'Connell, S., & Mithun, S. (2025, March). Assessing Personalized AI Mentoring with Large Language Models in the Computing Field. In *2025 IEEE Symposium on Computational Intelligence in Natural Language Processing and Social Media (CI-NLPSoMe Companion)* (pp. 1-5). IEEE.

Liu, Tiedong, and Bryan Kian Hsiang Low. "Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks." *arXiv preprint arXiv:2305.14201* (2023).

Q. Nguyen, D.-A. Nguyen, K. Dang, S. Liu, K. Nguyen, S. Y. Wang, W. Woof, P. Thomas, P. J. Patel, K. Balaskas, J. H. Thygesen, H. Wu, and N. Pontikos, "Advancing question-answering in ophthalmology with retrieval-augmented generation (rag): Benchmarking open-source and proprietary large language models," medRxiv,2024.[Online]. Available:https://www.medrxiv.org/content/early/2024/11/19/2024.11.18.24317510

Chen, J., Chen, S., Cao, J., Shen, J., & Cheung, S. C. (2025). When LLMs Meet API Documentation: Can Retrieval Augmentation Aid Code Generation Just as It Helps Developers?. *arXiv preprint arXiv:2503.15231*.

Sam, Kira & Vavekanand, Raja. (2024). Llama 3.1: An In-Depth Analysis of the Next Generation Large Language Model. 10.13140/RG.2.2.10628.74882.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, *568*, 127063.

Liu, Bo. (2024). Comparing sparse Llama 3 and Llama 2 models for on-device AI assistants. 10.21203/rs.3.rs-4927672/v2.

C. Yin and Z. Zhang, "A study of sentence similarity based on the all-minilm-l6-v2 model with 'same semantics, different structure' after fine-tuning," in Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024). Atlantis Press, 2024, pp. 677–684. [Online]. Available: https://doi.org/10.2991/978-94-6463-540-969

Grattafiori, Aaron, et al. "The llama 3 herd of models." *arXiv preprint arXiv:2407.21783* (2024).

Chen, Ziyang & Moscholios, Stylios. (2024). Using Prompts to Guide Large Language Models in Imitating a Real Person's Language Style. 10.48550/arXiv.2410.03848.

Ainslie, Joshua, et al. "Gqa: Training generalized multi-query transformer models from multi-head checkpoints." *arXiv preprint arXiv:2305.13245* (2023).

Huggingface. Meta-llama-2-7b, 2023. https://huggingface.co/meta-llama/ Llama-2-7b-hf, Accessed: 2024-07-31.2

Huggingface. Meta-llama-3-8b, 2024. https://huggingface.co/meta-llama/ Meta-Llama-3-8B, Accessed: 2024-07-31. 2

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. arXiv preprint arXiv:2306.03078, 2023

Huggingface. Meta-llama-3.1-8b-instruct, 2024. https://huggingface.co/meta-llama/ Meta-Llama-3.1-8B-Instruct, Accessed: 2024-08-02