LLaMA meets RAG: Concept-Aware Summarization of Academic Papers

Shribhakti S Vibhuti[1,*], Snehal V Devasthale[1], Shruti Sutar[1], Bhoomika Marigoudar[1], Saakshi Lokhande[1], and Dr. Uday Kulkarni[1]

Department of Computer Science and Engineering (AI)
KLE Technological University
Hubballi, India
*Corresponding authors' E-mail: 01fe22bci048@kletech.ac.in;
Contributing authors' E-mail(s):01fe22bci037@kletech.ac.in;
01fe22bci052@kletech.ac.in; 01fe22bci035@kletech.ac.in;
01fe22bci002@kletech.ac.in; uday_kulkarni@kletech.ac.in

Abstract. The exponential growth of scientific literature, particularly on platforms such as arXiv, presents a significant challenge for researchers to efficiently locate and grasp relevant content. This paper proposes a concept-aware summarization pipeline that combines Retrieval-Augmented Generation (RAG) with semantic search and fine-tuned language models. The system employs a transformer-based concept selector to identify key concepts, enhancing retrieval relevance and summary consistency. Summary generation is performed using a lightweight, Low-Rank Adaptation (LoRA)-optimized Large Language Model Meta AI(LLaMA) 3.2 (1B) model, enabling efficient, domain-aware summarization. Semantic search is facilitated through Facebook AI Similarity Search (FAISS), supporting query-based summarization workflows. The pipeline includes a Gradio-based user interface for accessible interaction. The proposed approach addresses the limitations of generic Large Language Models(LLMs) in handling long, domain-specific documents and aims to assist researchers in accessing concise, meaningful insights from scientific papers. The system demonstrates strong performance, achieving a ROUGE-L score of 0.73 and a BERTScore F1 of 0.95.

Keywords: Summarization,Concept Extraction,RAG, LLaMA, FAISS, Semantic Search,arXiv, LoRA,Research Paper Analysis

1 Introduction

In the domain of academic research, especially in fast-evolving fields like computer science and artificial intelligence, summarization plays a vital role in transforming lengthy, complex text into accessible formats that aid comprehension and discovery [1]. With researchers producing vast volumes of text, well-structured summaries have become indispensable tools for accelerating understanding and enabling knowledge dissemination [2].

Traditional methods of manual summarization, however, are often time-consuming and inefficient, especially when applied to dense scientific literature. The advent of Machine Learning (ML) and Deep Learning has significantly enhanced the ability of systems to understand and generate coherent summaries, with models now capable of preserving context and intent across various types of content [3].

At the forefront of these advancements are LLMs[4], such as GPT-3 and GPT-4 [5][6], as well as open-source alternatives like LLaMA [7]. These models have redefined the landscape of automatic summarization by generating human-like, well-organized summaries across multiple domains [8]. LLMs possess the ability to compress and rephrase large volumes of information into concise formats. Nevertheless, they encounter notable limitations when summarizing long documents like research papers, where context length exceeds the model's window, often resulting in lost details or inconsistencies between sections [9]. Another challenge is domain-specific understanding. General LLMs, trained on broad data, often struggle with technical terms and scientific accuracy—making them less reliable for tasks like summarizing research papers, where precision and domain knowledge matter most.

To overcome these barriers, RAG has emerged as a compelling solution [10][11]. RAG-enhanced systems first retrieve and rank relevant chunks of a document before passing the most pertinent passages to a generation model. This improves coherence and factual grounding while enabling effective summarization of longer texts. It also supports concept-aware structuring of information, aligning well with the needs of scholarly communication.

Platforms like arXiv, with their vast collection of research papers, pose challenges of content overload and domain complexity. Locating relevant literature, understanding methods, and identifying key contributions can be time-consuming. Existing summarization tools often lack technical depth, overlook nuances, or fail to adapt to user-specific needs—highlighting the need for domain-aware systems that offer more than surface-level summaries. To address this, we propose a fully automated summarization pipeline tailored for scientific literature. As shown in fig.1, the system combines RAG with key concept extraction to produce structured, accurate summaries. A transformer-based concept selector identifies semantically important ideas, while a LoRA-optimized LLaMA 3.2 (1B) model generates concise, focused summaries [12].

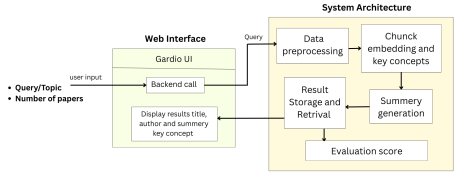

Fig. 1. Proposed Architecture for Concept-Aware Summarization

To enable efficient retrieval, the system uses FAISS[13][14] for semantic search and query-based summarization. It achieves strong results—ROUGE-1 (0.739), ROUGE-2 (0.736), ROUGE-L (0.739), and BERTScore F1 of 0.955—showing a 20% improvement over non-LoRA models in preserving content quality. Users can enter custom queries to obtain targeted summaries that correspond to the content of arXiv papers. A Gradio-based interface allows easy interaction , making the tool widely accessible. This study provides a scalable system for summarizing technical literature, facilitating deeper understanding and quicker discovery by combining domain-specific retrieval with improved generation and concept identification.

The study is segmented into various sections for better understanding. Section 2 is background study for better understanding of research. In addition, Section 3 provides information about the proposed pipeline followed by Section 4 is the results, and Section 5, the conclusions, and future work.

2 Literature Review

Recent progress in NLP has merged retrieval methods with LLMs, advancing automatic summarization of complex content like scientific papers. However, challenges remain, especially in ensuring factual accuracy, domain relevance, and handling lengthy texts. Models such as GPT-3 and GPT-4 have set new standards in language generation, producing coherent summaries even for technical queries, as noted in the GPT-4 Technical Report [6]. However, their limited context windows restrict their ability to process full research papers, often causing missing details, section inconsistencies, or hallucinations, highlighting the need for more robust long-context summarization solutions.

To overcome these issues, new paradigms, such as RAG, have emerged. RAG improves the performance of generative models by retrieving the relevant context from external knowledge sources before generating outputs. Notably, the Self-RAG framework builds on this idea by incorporating a feedback loop: it not only retrieves and generates, but also critiques its own responses. This self-reflective mechanism has been shown to significantly improve the factual precision and contextual coherence of generated summaries [15]. Such innovations mark a pivotal step towards making AI-generated summaries more trustworthy and evidence-based.

Recent work has shifted toward optimizing retrieval itself. Traditional RAG models rely on surface-level matching, whereas newer approaches, such as concept-aware retrieval, focus on deeper semantic understanding. Our pipeline adopts this by using a transformer-based concept selector to retrieve only meaningful, relevant content. This is supported by FAISS, which enables fast, scalable vector-based retrieval in high-dimensional semantic spaces [14]. FAISS plays a key role in ensuring real-time responsive retrieval, making it a vital component in modern RAG-based summarization systems.

The choice of generative model is crucial in summarization pipelines. Despite GPT-4's strength, lighter models, such as LLaMA 3.2–1B, perform well when

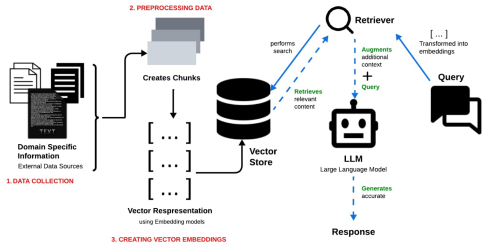

Fig. 2. RAG architecture

supported with efficient retrieval. Meta created LLaMA models to achieve the balance between accuracy and efficiency to increase the accessibility of high-quality LLMs [16]. Fine-tuned with LoRA, the 3.2–1B variant excels in focused tasks like summarization. Within a RAG setup, external retrieval offsets its smaller size, enabling competitive results in knowledge-heavy scenarios.

Another crucial area of focus in the literature has been the interface between users and AI systems. A growing body of research emphasizes that user interface design can significantly influence transparency, interpretability, and overall trust in AI-driven tools. This is especially true for applications like document summarization, where users benefit from real-time feedback and intuitive interactions. Gradio, an open source Python library, serves this purpose well by offering a simple yet versatile UI framework to deploy ML models, supporting multimodal input/output, and enabling seamless interaction, even for users without programming background [17][18].

Lastly, insights from adjacent fields such as code question answering further highlight the limitations of generic LLMs in technical domains. As demonstrated by Andryushchenko et al., without domain-specific tuning or retrieval support, LLMs often struggle to maintain logical structure or technical fidelity in their outputs [19]. These findings align with the broader consensus that concept-driven, retrieval-augmented, and lightweight summarization pipelines are essential for producing high-quality, context-aware summaries of complex scientific material.

3 Proposed Work

This work proposes a RAG pipeline enhanced with a concept-driven summarization strategy, using a lightweight LLM-LLaMA 3.2-1B-fine-tuned using LoRA, and a web-based user interface for accessibility.

3.1 System Architecture

The system integrates semantic search, key concept extraction, and LLM-based summarization to generate structured and informative summaries from arXiv research articles as shown in fig.3. It comprises the following components:

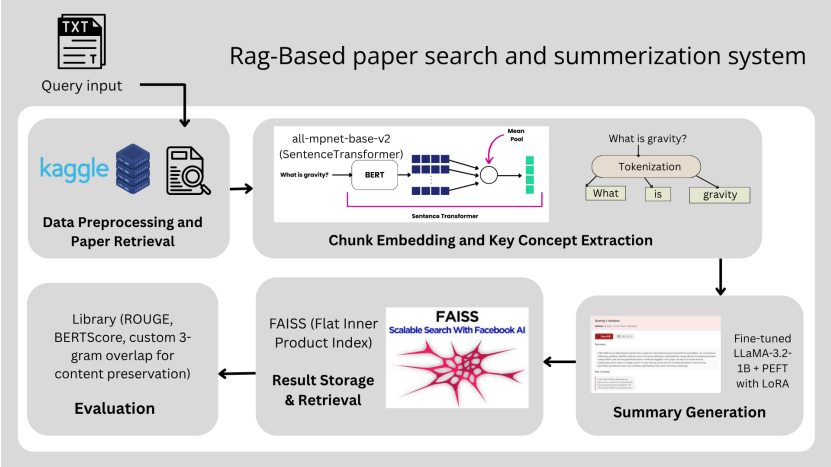

Fig. 3. Pipeline for RAG-Based paper search and summarization system

1. Data Representation and Paper Retrieval

Papers are retrieved using the arXiv API based on user-defined queries. Each retrieved paper is represented as a structured set:
P = {title, authors, abstract, published_date, updated_date, arxiv_id, pdf_url}

Here, the title refers to the name of the research paper, while authors is a list of contributors. The abstract provides a concise summary of the research. The published_date and updated_date denote the original publication date and the latest revision date, respectively. The arxiv_id is the unique identifier assigned by arXiv, and the pdf_url gives direct access to the paper in PDF format.

To search for papers, the system constructs a query string as defined in equation 1

$$\text{search\_query} = \text{"ti:"} + Q \tag{1}$$

where search_query is the final string sent to the arXiv API. The prefix `"ti:"` ensures the search is limited to paper titles, and Q represents the user-provided keyword or phrase (e.g., machine learning).

2. Chunk Embedding and Key Concept Extraction

Document processing begins by segmenting long texts into semantically coherent chunks using NLTK's sentence tokenizer [20]. Given a document T consisting of sentences $S = \{s_1, s_2, \ldots, s_n\}$, the system constructs a sequence of chunks $C = \{c_1, c_2, \ldots, c_m\}$, where each chunk $c_i = \{s_j, s_{j+1}, \ldots, s_{j+k}\}$ comprises consecutive sentences whose total token length approximates a predefined threshold τ, as defined in equation 2:

$$c_i = \{s_j, s_{j+1}, \ldots, s_{j+k}\} \quad \text{such that} \quad \sum_{l=j}^{j+k} |s_l| \approx \tau \tag{2}$$

where T is the input document, S represents its sentences, C contains the generated chunks, $|s_l|$ denotes token count of sentence s_l, and τ is the target chunk size.

Each chunk $c \in C$ is embedded into a dense vector representation using the SentenceTransformer model *all-mpnet-base-v2*. This yields embeddings $E_c \in \mathbb{R}^d$, where $d = 768$ denotes the embedding dimension, as defined in equation 3:

$$E_c = \text{SentenceEncoder}(c) \quad \forall c \in C \tag{3}$$

where E_c is the chunk embedding vector, d is the fixed dimension size (768), and SentenceEncoder is the pretrained transformer model.

To extract key concepts, the embeddings $E = [e_1, e_2, \ldots, e_n]$ are passed through a multi-layer Transformer encoder, generating contextually enriched representations $E' \in \mathbb{R}^{n \times d}$, as defined in equation 4:

$$E' = \text{TransformerEncoder}(E) \tag{4}$$

where E is the sequence of chunk embeddings and E' contains their contextualized representations with the same dimensions.

A multi-head self-attention mechanism is then applied over E' to compute inter-chunk dependencies, producing an attention matrix A, as shown in equation 5:

$$A_{,=} \text{MultiHeadAttention}(E', E', E') \tag{5}$$

where MultiHeadAttention computes relationships between chunks using queries, keys and values all set to E', outputting attention weights A.

The attention matrix is normalized and passed through a learned scoring network to generate importance scores $S \in \mathbb{R}^n$, represented in equation 6:

$$S = \text{Scorer}(\text{LayerNorm}(A)) \tag{6}$$

where LayerNorm applies normalization to A, and Scorer is a trainable network producing importance scores S for each chunk.

These scores are converted into probabilities using a sigmoid activation function, given by equation 7:

$$P = \sigma(S) \tag{7}$$

where σ is the sigmoid function converting scores S to probabilities P in [0,1] range.

The number of key concepts k is dynamically selected, constrained between a minimum of 3 and a maximum $K_{\max} = 7$, based on the number of chunks exceeding a 0.65 probability threshold, this is represented in equation 8:

$$k = \min\left(\max\left(|\{i : P_i > 0.65\}|, 3\right), K_{\max}\right) \tag{8}$$

where k is the final concept count, bounded by minimum 3 and maximum 7 concepts, selected from chunks with $P_i > 0.65$.

The TopK function is then used to select the indices K of the top k highest-scoring concepts, is given by equation 9:

$$K = \text{TopK}(S, k) \tag{9}$$

where TopK selects indices K of the k most important chunks based on their scores S, representing the final key concepts.

This pipeline effectively transforms raw documents into semantically segmented chunks, encodes them into vector representations, and identifies salient concepts using attention-based scoring mechanisms.

3. Summary Generation with LoRA-Adapted LLaMA
The summary generation process uses a lightweight large language model, LLaMA-3.2-1B, which is fine-tuned using PEFT with LoRA. A structured prompt for the key concepts that were the highest ranked extracted from the input document and sent to the model. These concepts guide the generation process by providing semantically rich context, allowing the model to focus on the most informative content. The LLaMA model, adapted with LoRA to reduce computational overhead and memory footprint, produces a logically and technically sound summary. This summary is organized into four logically connected sections: the problem statement, the methodology employed in the research, the key findings, and the implications and suggestions for future work. This structured generation ensures that the output is both concise and comprehensive, making it suitable for quick academic consumption and downstream research analysis.

4. Result Storage and Retrieval

We use FAISS to create a searchable index of paper embeddings. For each paper, we store its summary, key concepts, and original text. When searching, we find similar papers by calculating how closely the embeddings align with the query embedding.

5. Evaluation Metrics
We evaluate our system using standard metrics for both summarization and retrieval. Summarization quality is assessed with ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L), BERTScore for semantic similarity, and a content preservation measure. We also compare the efficiency of our LoRA-adapted LLaMA-3.2-1B model against full fine-tuning approaches, focusing on inference time and memory usage.

The algorithm used in system architecture is shown in algorithm.1.

3.2 Web Interface Implementation

The Gradio-powered web User Interface(UI) is deployed on edge-capable hardware with CPU/GPU support, enabling low-latency, real-time interaction for research paper summarization. Users can input queries, choose how many papers

Algorithm 1 RAG-Based Scientific Paper Search and Summarization Pipeline

Require: User query Q
Ensure: Ranked summaries with key concepts and evaluation metrics

Stage 1: Paper Retrieval
papers $\leftarrow$ ArXivSearch(config).search_papers(Q)
for each paper p in papers **do**
  text $\leftarrow$ FetchPaperText(p.arxiv_id)
  if text is None **then**
    text $\leftarrow$ p.abstract
  end if
  p.full_text $\leftarrow$ text
end for
Stage 2: Document Processing
for each paper p in papers **do**
  chunks $\leftarrow$ ChunkDocument(p.full_text)
  $E \leftarrow$ EncodeChunks(chunks)
  $H \leftarrow$ TransformerEncoder(E)
  $S \leftarrow$ ConceptSelector(H)
  key_concepts $\leftarrow$ TopK(chunks, S)
  doc_embedding $\leftarrow$ MeanPooling(H)
end for
Stage 3: Summary Generation
for each paper p in papers **do**
  prompt $\leftarrow$ StructurePrompt(key_concepts)
  summary $\leftarrow$ LLaMA_Generate(prompt)
end for
Stage 4: Storage and Retrieval
for each paper p in papers **do**
  FAISS.add(doc_embedding, {summary, key_concepts, p.full_text})
end for
$q \leftarrow$ Encode(Q)
results $\leftarrow$ FAISS.top_k(q)
Stage 5: Evaluation
for each result r in results **do**
  r.metrics $\leftarrow$ EvaluateSummary(r.summary, r.full_text)
end for
return ranked results with summaries, key concepts, and evaluation metrics
=0

to process, and receive responsive cards featuring titles, authors, summaries, key concepts, and PDF links. The app uses a lightweight RAG pipeline and offloads preprocessing to local threads. A reactive UI pattern shows loading animations before output, with dynamic HTML and pastel-themed cards rendered using Gradio's abstraction layer. The design cleanly separates backend logic from presentation, ensuring mobile responsiveness and robust error handling.

4 Results and Analysis

The proposed RAG-based pipeline was evaluated across all modules—retrieval, chunking, embedding, summarization, and user interaction—using speed, accuracy, and quality metrics. The following subsections detail the performance of each component.

4.1 System Architecture

The effectiveness of the system architecture was assessed across its key architectural components, as detailed below.

1. Data Representation and Retrieval

The arXiv API served as the primary retrieval source, with title-based queries ensuring relevance. Fallback to the Kaggle arXiv dataset guaranteed robustness when API responses were incomplete, as shown in Table 1.

Table 1. Paper Retrieval Performance and Corpus Composition

Metric	**Value**
Query source	arXiv API (title-restricted)
Total papers available	2.4 million (1991–present)
Technical papers	1.92 million (80%)
Non-technical content	480,000 (20%)
Avg. response time	<1 second
Fallback data source	Kaggle arXiv dataset (2.4M records)
Text availability	Abstracts + Metadata (100%)
Full-text availability	680,000 (28.3%)

2. Chunk Embedding and Key Concept Extraction

Chunking leveraged NLTK's sentence tokenizer for high accuracy, with optimal performance at 768 tokens per chunk. FAISS enabled low-latency embedding searches, though recall rates indicated room for improvement in semantic matching, as detailed in Table 2.

Table 2. Chunking and Embedding Performance Metrics

Component	**Value**
Tokenizer	NLTK Sentence Tokenizer
Tokenization accuracy	100%
Optimal chunk size	768 tokens
Chunking optimization metric	ROUGE-L
Embedding model	all-mpnet-base-v2
Search latency (FAISS)	0.12 ms
Recall@5	40%
Content preservation (manual evaluation)	57.7%

3. Summary Generation with LoRA-Adapted LLaMA

The LLaMA 3.2-1B model, fine-tuned via LoRA (configuration in Table 3), generated structured summaries using the decoding parameters from Table 4.

Table 3. LoRA Fine-Tuning Configuration

Hyperparameter	**Value**
Base model	LLaMA 3.2-1B
PEFT method	LoRA
Rank	8
Scaling factor	16
Dropout	0.1

4. Result Storage and Retrieval (FAISS)

FAISS demonstrated linear scalability with document volume (Table 5), maintaining sub-millisecond latency while minimizing memory overhead.

Table 4. Summary Decoding Hyperparameters

Parameter	**Value**
Max new tokens	250
Temperature	0.5
Top-p (nucleus sampling)	0.95
Beam size	4
Repetition penalty	1.15

Table 5. FAISS Retrieval Evaluation

Metric	**Value**
Scalability	Linear with indexed documents
Latency	0.12 ms
Memory overhead	Minimal (metadata + embeddings)

5. Evaluation Metrics

As quantified in Table 6, fine-tuning with LoRA yields substantial improvements over the baseline across all metrics. The ROUGE and BERTScore results confirm stronger alignment with reference summaries (e.g., +0.177 ROUGE-1, +0.046 BERTScore F1). Despite these gains, manual analysis reveals persistent opportunities to improve factual consistency and detail retention.

Table 6. Summary Evaluation: Baseline vs. LoRA Fine-Tuned Model

Metric	**Baseline**	**LoRA Fine-Tuned**
ROUGE-1	0.562	0.739
ROUGE-2	0.558	0.736
ROUGE-L	0.562	0.739
BERTScore F1	0.909	0.955
Content Preservation	0.385	0.577

The consistent gains across n-gram overlap (ROUGE), semantic similarity (BERTScore), and human evaluation suggest that fine-tuning enhances both surface-level and conceptual summary quality.

4.2 Web Interface Implementation

The Gradio-based UI (as shown in Fig. 4) offers an intuitive interface for users to search research papers by entering a query and selecting the desired number of results using a slider. The output includes essential metadata such as the title, authors, and publication date, along with a direct link to the PDF, an

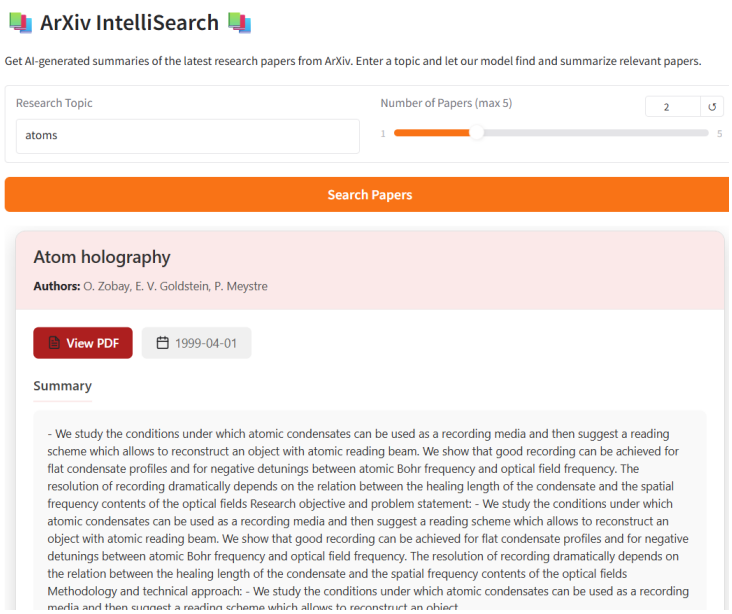

Fig. 4. Web Interface for Paper Search and Summarization

AI-generated summary, and extracted key concepts. Built with Gradio, the interface ensures a user-friendly and efficient experience for accessing summarized scientific content.

5 Conclusion and Future Work

The proposed concept-aware summarization pipeline, integrating RAG with a LoRA-optimized LLaMA 3.2–1B model, demonstrates strong performance (ROUGE-L: 0.739, BERTScore: 0.955) in generating structured scientific summaries while addressing key limitations of generic LLMs. Future work will explore full-text integration, cross-domain generalization, and multilingual support to enhance the system's robustness and applicability. The interactive Gradio interface provides a practical foundation for these extensions, enabling real-time researcher engagement with large academic repositories.

References

1. A. Nenkova and K. McKeown, *A Survey of Text Summarization Techniques*. Boston, MA: Springer US, 2012, pp. 43–76. [Online]. Available: https://doi.org/10.1007/978-1-4614-3223-4_3
2. R. Gandhi. (2024) A comprehensive dive into research paper summaries. Accessed: 2025-04-30. [Online]. Available: https://centilio.com/resources/a-comprehensive-dive-into-research-paper-summaries/

3. A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *CoRR*, vol. abs/1704.04368, 2017. [Online]. Available: http://arxiv.org/abs/1704.04368
4. S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," 2025. [Online]. Available: https://arxiv.org/abs/2402.06196
5. G. Yenduri, R. M, C. S. G, S. Y, G. Srivastava, P. K. R. Maddikunta, D. R. G, R. H. Jhaveri, P. B, W. Wang, A. V. Vasilakos, and T. R. Gadekallu, "Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions," 2023. [Online]. Available: https://arxiv.org/abs/2305.10435
6. OpenAI and J. A. et al., "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774
7. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971
8. J. A. Baktash and M. Dawodi, "Gpt-4: A review on advancements and opportunities in natural language processing," 2023. [Online]. Available: https://arxiv.org/abs/2305.03195
9. N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," 2023. [Online]. Available: https://arxiv.org/abs/2211.08411
10. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2312.10997
11. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401
12. L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," 2023. [Online]. Available: https://arxiv.org/abs/2308.03303
13. H. Jégou, M. Douze, and J. Johnson. (2017) Faiss: A library for efficient similarity search. Accessed: 2025-04-30. [Online]. Available: https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/
14. M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2025. [Online]. Available: https://arxiv.org/abs/2401.08281
15. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," 2023. [Online]. Available: https://arxiv.org/abs/2310.11511
16. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux *et al.*, "Llama 3: Open foundation and instruction-tuned language models," https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md, 2024.
17. A. Abid, A. Abdalla, J. Zhang, and J. Zou, "Gradio: Hassle-free sharing and testing of ml models in the wild," *arXiv preprint arXiv:1906.02569*, 2019.
18. S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, M. Cakmak, E. Kamar, and E. Horvitz, "Guidelines for human-ai interaction," *arXiv preprint arXiv:1906.02569*, 2019.

19. G. Andryushchenko, V. Ivanov, V. Makharev, E. Tukhtina, and A. Valeev, "Leveraging large language models in code question answering: Baselines and issues," 2024. [Online]. Available: https://arxiv.org/abs/2411.03012
20. S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.