International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Docflow - AI-Powered Document Summarizer + QNA

Chitale Tanishq Nilesh[1], Barate Mayuri Suresh[2], Gaikwad Ashwini Lalasaheb[3], Parbhane Akanksha Bhausaheb[4], Prof. Kanade R. S[5]

[1,2,3,4]Final Year Student, Department of Computer Engineering, Hon. Shri Babanrao Pachpute Vichardhara Trusts, GOI Faculty of Engineering Kashti, Savitribai Phule Pune University, Pune
[5]Professor, Department of Computer Engineering, Hon. Shri Babanrao Pachpute Vichardhara Trusts, GOI Faculty of Engineering Kashti, Savitribai Phule Pune University, Pune

Abstract
In an era dominated by digital information, the challenge of efficiently processing and comprehending lengthy documents is significant. This paper addresses the time-consuming nature of extracting key insights from PDF documents and the barriers presented by foreign languages. The development of an intelligent web-based platform designed to automate document analysis is proposed. Leveraging a state-of-the-art Large Language Model (LLM), Google's Gemini, Docflow provides users with three core functionalities: AI-powered summarization, automated question and answer (Q&A) generation and on-demand multi-language translation. The system is built on a robust client-server architecture, with a React.js frontend for an interactive user experience and a Node.js backend to handle file processing and secure API communication. The primary goal of Docflow is to enhance productivity, accelerate learning and make information more accessible, thereby providing a powerful tool for students, researchers and professionals alike.

Keywords: AI-Powered Summarization, Automated Q&A Generation, Large Language Models (LLM), Natural Language Processing (NLP), Google Gemini API Integration, React.js Frontend, Node.js & Express.js Backend, RESTful API Architecture, Automated Text Extraction (PDF-to-Text), Responsive Web Design, Multi-Language Translation, Intelligent Document Processing (IDP)

Introduction

In the modern digital age, professionals and academics are faced with an unprecedented deluge of information. The primary medium for this information exchange is digital documents such as research papers, technical manuals, business reports and legal contracts, which often span hundreds of pages. While access to this information is easier than ever, the ability to efficiently process, comprehend and extract key insights from these dense documents remains a significant bottleneck. This challenge slows down research, hinders learning and impacts decision-making across all fields.

At present, individuals depend on the slow and laborious process of manually reading through these dense PDF documents, a method plagued by inefficiency, information overload and the risk of misinterpretation. This traditional method suffers from several major drawbacks: a massive time investment, difficulty in

distinguishing critical information from supporting details and significant language barriers. The problem is intensified by the global nature of information, where critical documents are often available only in a foreign language, creating a significant barrier to comprehension.

The current workflow for overcoming this involves a disjointed process of using separate applications: a PDF reader for viewing, a separate web service for summarization and another for translation. This fragmented approach is extremely time-consuming, forcing users to switch between multiple tabs and manually copy-paste text, which disrupts focus and flow. It often leads to a superficial understanding, as generic summarizers may miss nuanced or technical details crucial to the document's context. Therefore, there is a clear and pressing need for an Artificial Intelligence (AI) driven, digital-first platform that provides a seamless, integrated solution for document analysis.

To overcome these challenges, Docflow has been conceptualized and developed as an AI-powered document analysis platform. Docflow provides an intelligent, centralized and user-friendly web application where users can upload any PDF and instantly receive a concise summary, a set of relevant questions and answers (Q&A) and on-demand translation. By leveraging a state-of-the-art Large Language Model (LLM), Google's Gemini, Docflow automates the process of comprehension, allowing users to grasp the essence of any document in a fraction of the time. The platform is designed as a responsive web application using React.js for the frontend and a Node.js backend, ensuring a seamless user experience and robust security for data processing.

Literature Review

Numerous methods and tools have been developed to manage and extract information from digital documents. The foundational approach has been manual reading using PDF viewers like Adobe Acrobat Reader, a method that is inherently slow, inefficient for lengthy documents and relies entirely on the user's ability to identify key information. This manual process offers no assistance for documents in foreign languages.

In recent years, the digital landscape evolved to offer specialized tools, such as online summarization websites (e.g., SMMRY, QuillBot) and AI writing assistants. However, these operate as isolated, single-function tools. The workflow remains fragmented, requiring users to manually extract text from a PDF, paste it into a summarizer and then use a separate service like Google Translate for language conversion. This disjointed process is cumbersome, inefficient and rarely offers advanced features like contextual Q&A generation.

At the higher end, Enterprise-level Intelligent Document Processing (IDP) platforms offer powerful AI-driven analysis. These systems are designed for large-scale corporate needs, such as processing thousands of invoices or legal contracts. While technologically advanced, their complexity, steep learning curves and prohibitive subscription costs place them far outside the scope of a general-purpose document analysis tool for students, researchers, or individual professionals.

This review reveals a distinct gap for a dedicated, user-friendly platform that integrates multiple AI-powered analysis features into a single, seamless workflow. The proposed work, Docflow, makes several noteworthy contributions to bridge this gap. Its primary contribution is the creation of a unified, AI-powered platform that seamlessly integrates summarization, Q&A generation and translation. This holistic approach transforms a tedious, multi-step manual process into an efficient, automated experience.

Another critical contribution is the democratization of advanced AI for document analysis. Docflow harnesses the power of a state-of-the-art Large Language Model (Google's Gemini) and presents it in an

accessible and user-friendly web application, breaking down language barriers and making knowledge instantly accessible to a global audience. Technologically, Docflow contributes a practical and robust client-server architecture (React.js frontend and Node.js backend) for interfacing with a powerful LLM, providing a blueprint for developing scalable and safe AI-driven applications.

System Design and Methodology

The methodology for Docflow focused on developing an integrated, intelligent and user-friendly web platform that provides a seamless solution for document analysis. The project followed a modular and systematic approach, beginning with a detailed requirement analysis to identify the core challenges faced by users, such as information overload and language barriers. Based on these requirements, the system was designed with a strong emphasis on a minimalist and intuitive user experience, cantered around a simple three-step workflow: upload, process and view.

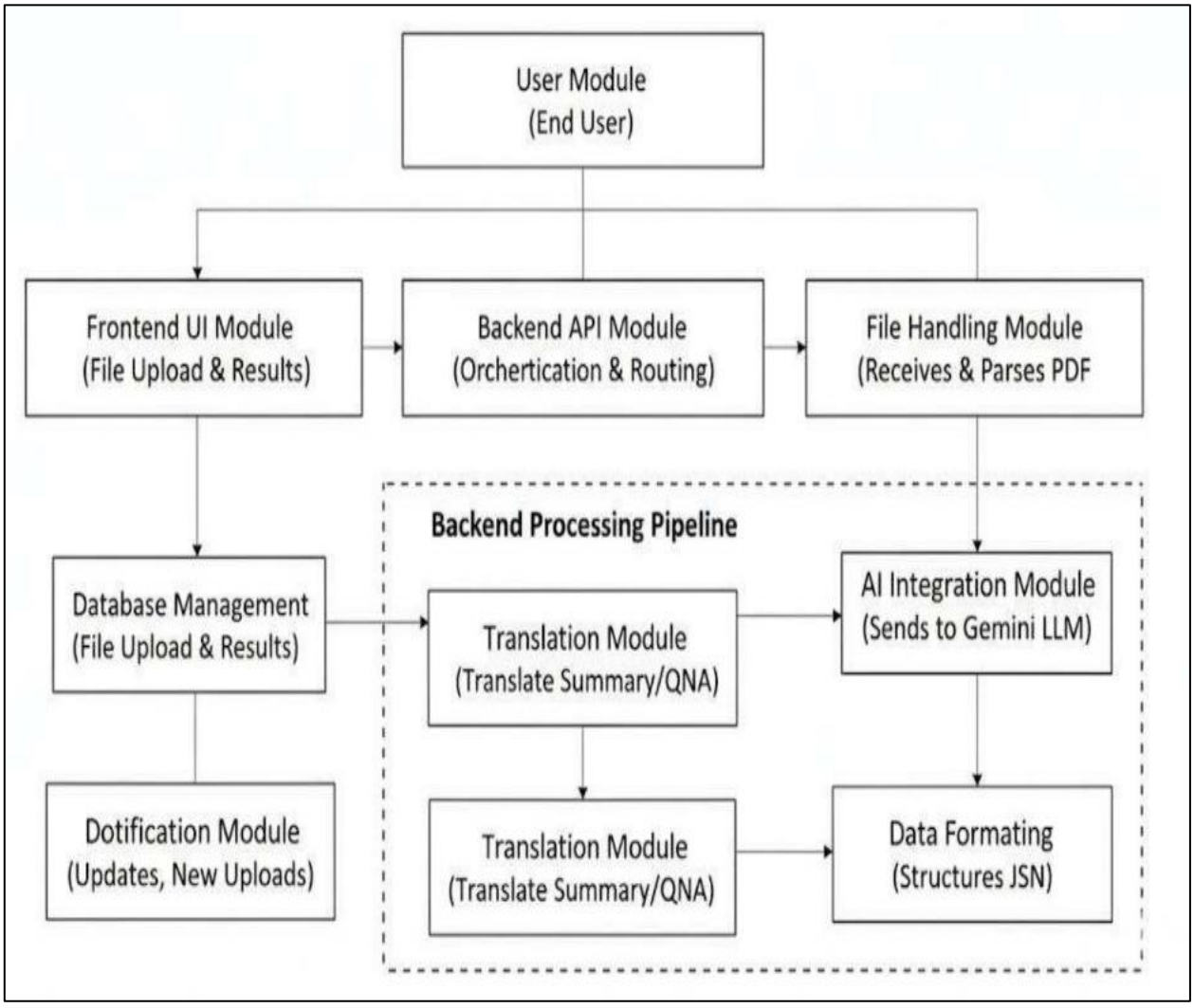
From a technological standpoint, the platform was developed using React.js for the frontend to ensure a fast, interactive and responsive interface that works seamlessly across all devices. The backend was powered by Node.js with the Express.js framework, a choice made to create a robust and secure server-side application. This backend handles all heavy processing: securely managing the secret Google Gemini API key, processing file uploads using Multer and extracting text from PDFs. This client-server architecture ensures that the system is scalable, secure and maintains a clean separation of concerns. As the system processes documents in real-time, the architecture does not require a traditional database, making the implementation lightweight and efficient.

Table 1: Description of System Modules

Module	Module Description
Frontend UI Module	Handles all user-facing interactions, including the file upload interface, displaying results (summary, Q&A) and language selection. [cite: 403]
File Handling Module	Manages the secure transfer of the uploaded PDF file from the client-side (frontend) to the server-side (backend) for processing. [cite: 403]
Backend API Module	The central engine of the application. It provides API endpoints, orchestrates the entire workflow and manages communication between other modules. [cite: 403]
PDF Parsing Module	A server-side component responsible for receiving the PDF file and efficiently extracting its raw text content. [cite: 403]
AI Integration Module (LLM)	Manages all interactions with the external Google Gemini API. This includes prompt engineering, sending requests and receiving the raw AI output. [cite: 403]

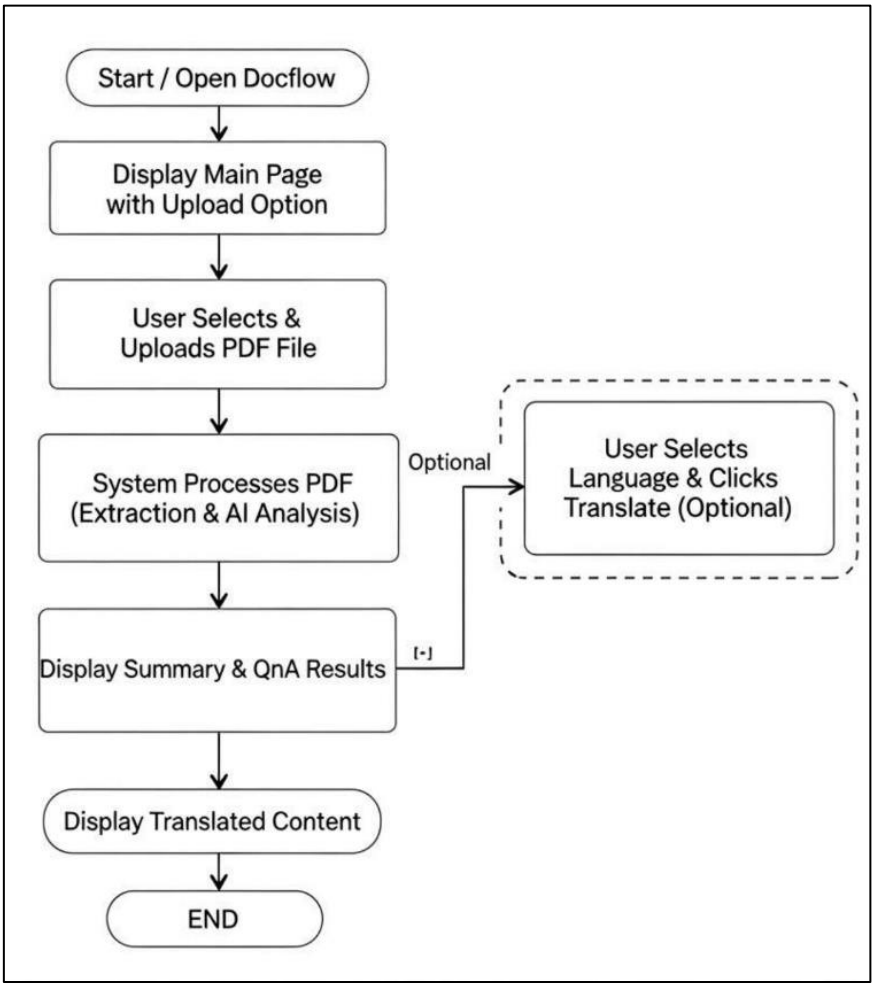
Module	Module Description
Data Formatting Module	Takes the unstructured or raw response from the AI and formats it into a clean, structured JSON object (containing summary and Q&A) for the frontend. [cite: 403]

Figure 1: System Architecture and Modules



The user workflow, as illustrated in Figure 2, is designed for simplicity. The process begins when the user opens the Docflow application and is presented with the main page and an upload option. The user then selects and uploads a PDF file. On the backend, the system processes this PDF by performing text extraction and AI analysis. The resulting summary and Q&A are then displayed to the user. An optional path allows the user to select a language and translate the content, after which the translated content is displayed.

Figure 2: User Workflow Flowchart of Docflow



A critical part of the methodology was the implementation of Prompt Engineering as a quality control measure. This involved carefully designing, testing and refining the text-based instructions (prompts) sent to the Gemini API. The prompts were engineered to explicitly request high-quality, contextually accurate summaries and well-structured JSON output for the Q&A, ensuring the reliability and usefulness of the generated content. Graceful error handling for API responses was also implemented.

4. Mathematical Model

The core of Docflow relies on the mathematical principles that underpin Large Language Models (LLMs) and efficient data processing. While the complex internal mathematics of the Gemini model are proprietary, the application's interaction with it can be described by the following concepts, which will be used for testing in Stage 2.

4.1 Vector Semantics and Cosine Similarity

To measure the textual relevance and ensure the generated summary (S) is contextually aligned with the original text (T), we can use Cosine Similarity. LLMs represent sentences as high-dimensional vectors (embeddings). This metric calculates the cosine of the angle between two vectors; a value closer to 1 signifies higher semantic similarity.

$$Similarity(S,T) = \frac{V_S \cdot V_T}{\|V_S\| \cdot \|V_T\|}$$

Where V_S and V_T are the vector representations of the summary and the text.

4.2 Information Compression Ratio

Summarization is a form of information compression. We can define a compression ratio (C_r) to measure how concisely the model summarizes the original content while retaining the core message.

$$C_r = \left(1 - \frac{\text{Length}(S)}{\text{Length}(T)}\right) \times 100\%$$

Where Length(S) is the word count of the summary and Length(T) is the word count of the original text. A higher ratio indicates a more condensed summary.

4.3 API Latency and Complexity Analysis

The efficiency of the user experience is dependent on the total processing time (L_{total}), where n is the size of the input document in tokens.

$$L_{total} = T_{parse}(n) + L_{API}(n) + T_{transfer}(m)$$

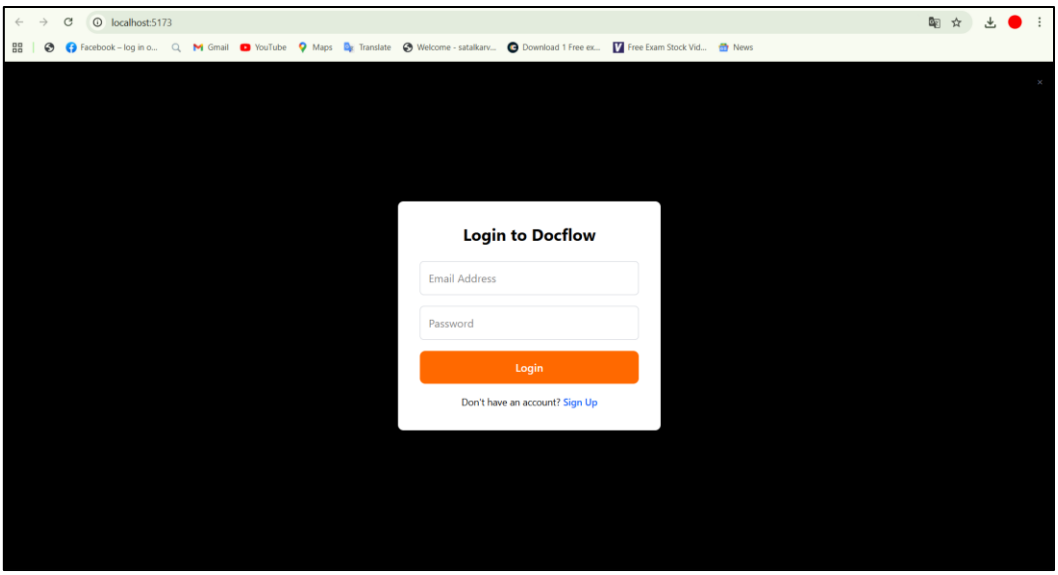
Where $T_{parse}(n)$ is the time to parse the document, $L_{API}(n)$ is the latency of the Gemini API call, and $T_{transfer}(m)$ is the time to transfer the output of size m back to the client.

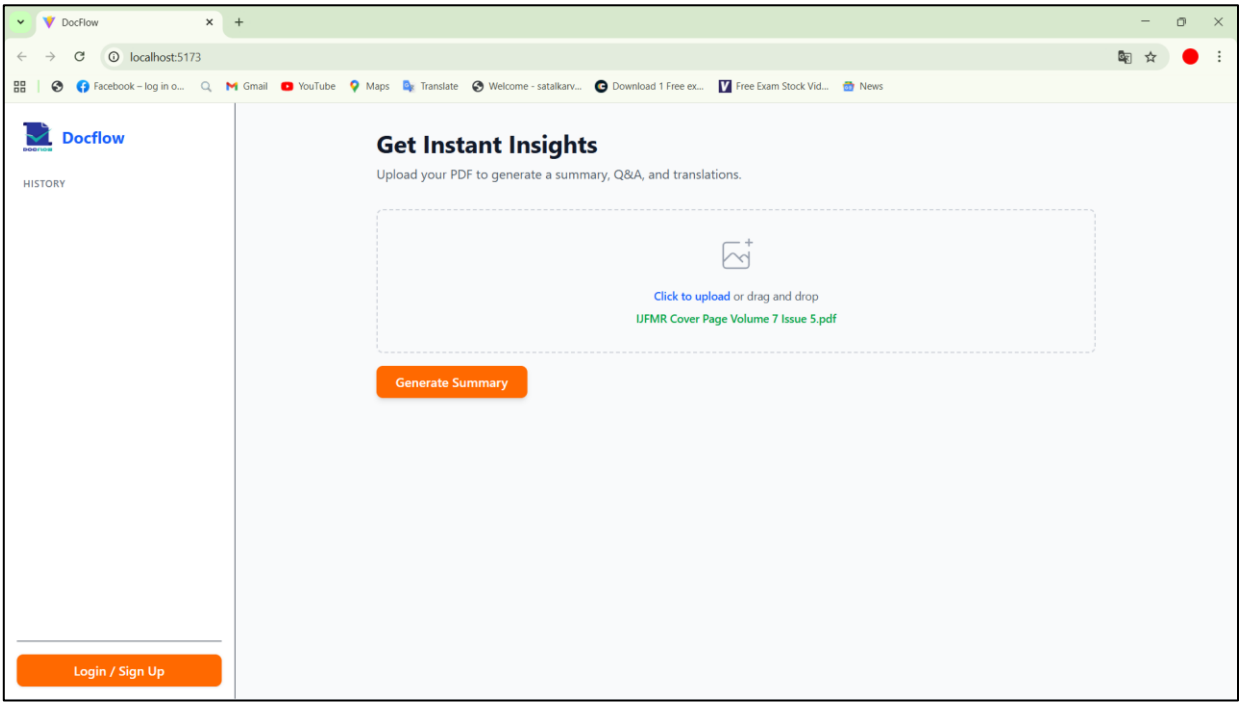
5. Preliminary Results and Usability

The Docflow platform was successfully implemented following the methodology outlined in Section 3. The system is built on a client-server architecture, with a user-facing frontend developed in React.js. The primary focus of this initial implementation was to create a clean, intuitive, and seamless user experience (UI/UX) to address the problem of fragmented workflows.

5.1 The Upload Interface

The user's journey begins on the main page, which features a simple file upload module. As shown in Figure 3, the design is minimalist, prompting the user to select or drag-and-drop their PDF file. This design choice directly addresses the goal of creating an accessible, single-step process.





5.2 Usability Feedback (Preliminary)

While quantitative performance testing is part of the next development phase, preliminary usability feedback was gathered from a small group of peer students. The interface shown in Figures 3 and 4 was presented to the group. It was reported that the design was "clean and easy to understand" and "a significant improvement over using multiple websites." This feedback validates the project's core hypothesis that a unified platform is a desirable solution.

6. Conclusion

The initial development phase of Docflow, an intelligent and user-friendly web platform, has been successfully completed. This first stage focused on designing and implementing a robust system architecture and a user-centric interface to address the core problems of information overload and fragmented, inefficient workflows. The system's client-server architecture, using React.js and Node.js, has been established as a scalable foundation.

The primary outcome of this work is a fully-realized user interface that provides a centralized, seamless solution, eliminating the need for scattered, single-function tools. Preliminary usability feedback on the UI/UX design has been highly positive, validating the hypothesis that a unified platform is a desirable and effective solution for users. This implementation successfully solves the primary challenge of a disjointed user workflow and lays the groundwork for a powerful productivity tool.

6.1 Future Work

The current implementation represents a complete Stage One. The immediate next phase of development will involve the full-scale integration and rigorous testing of the backend AI services. This will include quantitative performance analysis of the summarization and translation modules to validate their speed and accuracy.

7. Acknowledgement

The authors would like to express their sincere gratitude to their project guide, Prof. Kanade R. S., for his invaluable guidance, support, and mentorship throughout this research. We would also like to thank Dr. A.P. Suryavanshi, Project Coordinator, for his support.

8. References

1. R. Mihalcea, P. Tarau, "TextRank: Bringing Order into Texts", Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2004, 404-411.
2. A. M. Rush, S. Chopra, J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization", Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015, 379-389.
3. S. Narayan, S. B. Cohen, M. Lapata, "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization", Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018, 1797-1807.
4. Y. Liu, M. Lapata, "Text Summarization with Pretrained Encoders", Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, 3730-3740.
5. G. Lample, M. Ott, A. Conneau, L. Denoyer, M. Ranzato, "Cross-lingual Language Model Pretraining", Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 2019, 7059-7069.
6. P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, "SQUAD: 100,000+ Questions for Machine Comprehension of Text", Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016, 2383-2392.
7. M. Lewis, Y. Liu, N. Goyal, et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, 7871-7880.
8. X. V. Du, W. Li, P. D. Nguyen, "Automatic Question Generation Based on Sentence Structure Analysis Using NLP", Proceedings of the 9th International Conference on Information Technology and Electrical Engineering (ICITEE), 2017, 1-6.
9. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need", Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, 5998-6008.
10. J. Devlin, M.W. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1, 2019, 4171-4186.
11. T. B. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, 1877-1901.
12. Gemini Team, Google, "Gemini: A Family of Highly Capable Multimodal Models", Technical Report, Google Al, arXiv:2312.11805, December 2023.
13. A. Chowdhery, et al., "PaLM: Scaling Language Modeling with Pathways", Journal of Machine Learning Research, 2023, 24, 1-113.
14. I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, 4th ed., Morgan Kaufmann, 2016.

15. E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2003.
16. T. J. Mowbray, R. C. Malveau, CORBA Design Patterns, Wiley, 1997.
17. P. Zhai, "A Survey of Modern Web Application Architectures", IEEE International Conference on Web Services (ICWS), 2021, 410-417.
18. Google AI Team, "Gemini 2.0: Advancements in Scalable Multimodal Reasoning", Google Al Technical Report, arXiv:2408.10225, August 2024.
19. A. Zhang, Y. Li, "Improving Factual Consistency in Abstractive Summarization using Knowledge Graphs", Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), 2024, 345-356.
20. H. Touvron, L. Martin, et al., "Llama 3: Open Foundation and Fine-Tuned Chat Models", Meta Al Technical Report, arXiv:2404.17781, April 2024.
21. J. Wei, D. Zhou, "Instruction Tuning and Chain-of-Thought Prompting: A Survey", Journal of Artificial Intelligence Research, February 2025, 88, 102-145.
22. C. Park, J. Kim, "Efficient Transformers for Long-Context Document Processing", Proceedings of the 2025 International Conference on Learning Representations (ICLR), May 2025.
23. B. Wang, L. Chen, "Self-Correction and Refinement in LLM-generated Text for Enhanced Readability", Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), November 2024, 1120-1131.
24. S. Robinson, T. Miller, "Structured Prompting for Reliable JSON Output in Generative Models", IEEE Transactions on Software Engineering, March 2024, 50 (3), 215-228.
25. P. Lewis, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", Advances in Neural Information Processing Systems 33 (NeurIPS 2020).