DOI: 10.17148/IJARCCE.2025.14371

ShodhX: Efficient Document Analysis and Interaction Using Large Language Models

Mrs. Swati Chiplunkar[1], Ms. Thanisha Belchada[2], Mr. Aditya Joshi[3],

Mr. Vinaykumar Choursiya[4]

Assistant Professor, Department of IT, Thakur College of Engineering and Technology, Maharashtra, India[1]

Student, Department of IT, Thakur College of Engineering and Technology, Maharashtra, India[2,3,4]

Abstract: This research paper explores the development of an application leveraging fine-tuned Large Language Models (LLMs) for advanced interaction with research papers and GitHub repositories. Traditional methods often involve manual effort or siloed tools, lacking integrated insights across textual and code-based resources. Our system bridges this gap, enabling developers and researchers to analyze papers, understand implementations, and create adaptations seamlessly. By fine-tuning a pre-trained LLM on research papers and source code, and integrating Retrieval-Augmented Generation (RAG) techniques, the system delivers contextual and dynamic interactions. Unlike previous approaches focusing on isolated analysis or summarization, our unified platform combines academic and practical insights, helping users navigate complex topics, validate claims, and prototype ideas efficiently. This work addresses key challenges in integrating text and code for knowledge discovery, setting the stage for enhanced research and development workflows.

Keywords: Large Language Models, AI-Powered Code Understanding, Codebase Analysis, Intelligent Document Processing, Automated Literature Review, Contextual Code Retrieval.

I. INTRODUCTION

The rapid evolution of Large Language Models (LLMs) has revolutionized the field of natural language processing (NLP), enabling models to perform complex tasks such as text generation, summarization, and question-answering with unprecedented accuracy [1]. Despite these advancements, existing applications often fall short in integrating diverse resources, such as textual documents and codebases, into a unified framework. Researchers and developers frequently encounter challenges when attempting to bridge the gap between theoretical knowledge presented in research papers and its practical implementations in repositories [2]. Traditional approaches to research analysis involve manual reading and interpretation, which can be time-consuming and inefficient. Similarly, exploring GitHub repositories for corresponding implementations requires significant effort, often leading to fragmented insights [3].

While some tools offer partial solutions, such as code summarization or document search, they lack the contextual and dynamic interaction necessary for a seamless user experience. Our proposed system addresses these limitations by leveraging fine-tuned LLMs and Retrieval-Augmented Generation (RAG) techniques [4]. By combining the semantic understanding of research papers with the technical details of GitHub repositories, the system enables users to interact with both resources cohesively. For instance, users can query the system to explain specific concepts from a paper while simultaneously exploring relevant code snippets from repositories, facilitating a deeper understanding and quicker prototyping. This paper outlines the methodology for fine-tuning LLMs on a custom dataset comprising research papers and source code, as well as the integration of RAG to enhance user interactions. We also discuss the challenges and innovations associated with aligning diverse data formats and creating a responsive, intuitive interface for end-users. By addressing these challenges, our system paves the way for streamlined research and development workflows, bridging the gap between theory and practice in an unprecedented manner.

II. LITERATURE SURVEY

The integration of Large Language Models (LLMs) in research and software development has been explored extensively, with prior work focusing on improving document retrieval, summarization, and code generation. Vaswani et al. [1] introduced the Transformer model, which laid the foundation for modern LLMs by utilizing self-attention mechanisms to improve contextual understanding. However, while Transformers significantly enhanced language modeling, they were not designed for domain-specific integration with research and code, requiring extensive fine-tuning for specialized use cases. Devlin et al. [2] expanded on this with BERT, which demonstrated improvements in bidirectional language modeling and contextual embeddings.

Nevertheless, BERT's static pretraining limited its ability to dynamically retrieve and interpret research-related queries, particularly when combined with evolving repositories of code. Lewis et al. [4] proposed Retrieval-Augmented Generation (RAG), an approach that combines retrieval mechanisms with generative models to improve knowledge-intensive tasks. While RAG significantly improved contextual responses in open-domain question answering, its application to integrating research papers and GitHub repositories remains largely unexplored. Chen et al. [3] examined LLMs for code synthesis and comprehension, which enhanced code generation but did not establish a clear link between theoretical knowledge from academic papers and practical implementation.

Existing solutions, such as OpenAI's Codex and GitHub Copilot, have demonstrated substantial advancements in AI-assisted coding. However, these systems primarily focus on code completion and generation rather than an interactive research-driven approach. Our work builds upon these prior studies by integrating an LLM fine-tuned for research comprehension and codebase analysis. Unlike prior approaches that handle text and code independently, our system unifies both domains through Retrieval-Augmented Generation, allowing users to derive structured insights from research papers while simultaneously exploring related implementations in GitHub repositories. By addressing the gaps in contextual code retrieval and academic research interpretation, our approach facilitates a more comprehensive and interactive workflow for researchers and developers alike.

III. METHODOLOGY

Our methodology is designed to seamlessly integrate research paper analysis with codebase understanding through a fine-tuned Large Language Model (LLM) and Retrieval-Augmented Generation (RAG). The system aims to bridge the gap between theoretical knowledge and practical implementation, enabling researchers and developers to derive meaningful insights. The core components of our approach include data acquisition, model fine-tuning, retrieval mechanisms, query processing, response generation, and continuous improvement.

A. Data Collection and Preprocessing

The first step in our methodology involves data acquisition, ensuring that research papers and corresponding source code repositories are readily available for retrieval and analysis. To achieve this, we curate a domain-specific dataset, sourcing research papers from open-access journals, preprint servers, and institutional repositories while acquiring relevant codebases from public repositories like GitHub. Once collected, research papers undergo preprocessing, which includes tokenization (breaking text into structured units), section segmentation (dividing content into abstracts, introductions, methodologies, results, and discussions), and metadata extraction (capturing citations, author details, keywords, and references). Similarly, source code is parsed through functional unit segmentation (dividing code into functions, classes, and modules), inline documentation extraction (analyzing comments and docstrings for context), and dependency analysis (understanding package imports and their interactions). This structured preprocessing maintains the contextual integrity of both textual and code-based data, ensuring efficient retrieval and interpretation.

B. Fine-Tuning the Large Language Model (LLM)

To enhance research comprehension and programming understanding, we adopt a two-stage fine-tuning process. The first stage, domain-adaptive pretraining, involves additional pretraining of the base transformer model on our curated dataset. This allows the model to specialize in academic writing and software engineering concepts. In the second stage, task-specific fine-tuning, the model undergoes supervised learning to improve its information retrieval and summarization tasks, particularly aligning research findings with corresponding implementations in source code. This hierarchical approach ensures that the LLM is well-equipped to analyze research literature and its real-world applications.

C. Retrieval-Augmented Generation (RAG)

To facilitate precise knowledge extraction, we integrate a RAG framework that dynamically retrieves relevant content from research papers and code repositories. The RAG pipeline involves three key steps: embedding and indexing, query processing, and contextual response generation. Research papers and code snippets are converted into dense vector representations using transformer-based embeddings, which are then stored in a scalable vector database for efficient retrieval. When a user submits a query, the system transforms it into vector representations, searching for semantically similar content within the database. The retrieved content is then appended to the query, enabling the fine-tuned LLM to generate responses that integrate both theoretical insights and practical implementations.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2025
DOI: 10.17148/IJARCCE.2025.14371

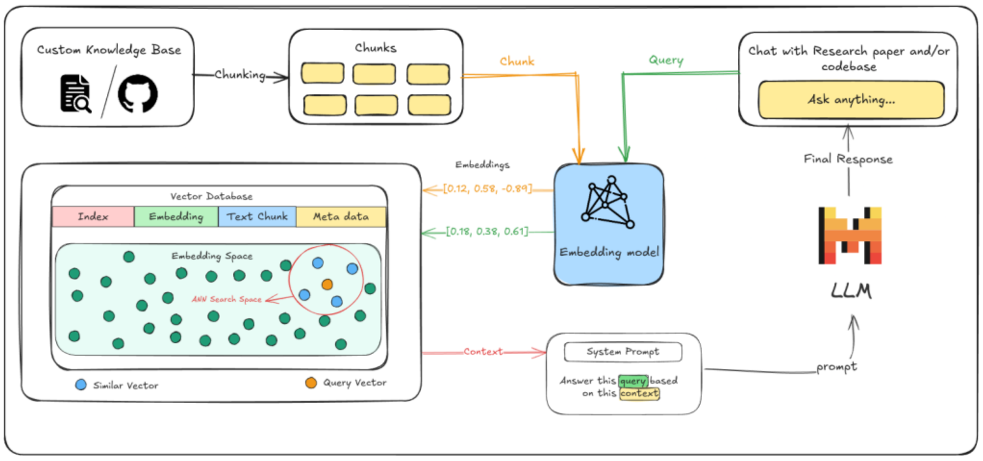


Figure 1: LLM-RAG Architecture – Query-driven retrieval with contextual response generation.

D. Hybrid Search and Adaptive Filtering

To improve retrieval accuracy, our system employs a hybrid search strategy that combines neural embeddings for semantic search with keyword-based indexing for precision. While embedding-based search ensures contextual similarity between queries and stored knowledge, keyword-based indexing refines results for enhanced specificity. Additionally, adaptive filtering mechanisms dynamically adjust ranking based on query complexity and authoritative sources, balancing precision and recall for high-quality search results.

E. System Deployment and User Interaction

Our trained model is deployed as an API, allowing users to interact with it via a web-based interface. The system follows a structured workflow: users submit queries, upload research papers, or provide GitHub repository links for code retrieval. Once a query is received, the system processes it and retrieves the most relevant content, presenting it in a clear, structured format. To further enhance usability, an interactive feedback loop is integrated, allowing users to rate response relevance and accuracy, thereby refining retrieval performance over time.

F. Continuous Learning and Model Refinement

To ensure that ShodhX remains accurate and relevant, we implement a continuous learning pipeline. This involves periodic model retraining with newly published research papers and updated GitHub repositories, ensuring the model stays up-to-date with the latest developments. Additionally, user interactions are logged and analyzed, helping identify patterns for improvement. Retrieval strategies are also continuously refined based on real-world user interactions, ensuring that query responses become more precise and insightful over time.

G. Evaluation and Performance Metrics

The effectiveness of our system is assessed through multiple evaluation metrics. Relevance and accuracy are measured using precision and recall, ensuring that retrieved documents and code snippets align with user queries. Semantic coherence is evaluated using NLP metrics such as BLEU and ROUGE scores, which quantify the quality of generated responses. Additionally, execution and usability testing is conducted, ensuring that code suggestions are functionally correct and aligned with the theoretical content. Lastly, qualitative feedback from researchers and developers helps improve overall system usability and efficiency.

H. Scalability and Future Enhancements

Our architecture is designed to support scalability, allowing distributed deployment and efficient parallel processing. Future improvements will include expanding multimodal capabilities by integrating additional data sources like datasets, graphs, and experimental logs. Additionally, we plan to enhance code analysis by implementing static and dynamic analysis techniques, enabling the system to assess code quality and suggest optimizations. Lastly, we aim to introduce personalized recommendations, where machine learning algorithms tailor suggestions based on user preferences and past interactions.

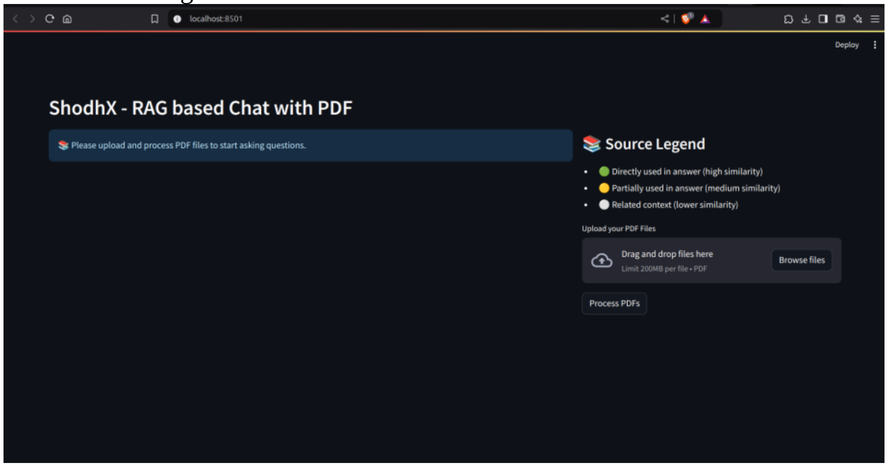
By adopting this comprehensive methodology, our system ensures a seamless, intelligent, and continuously improving interaction between theoretical research and practical implementation, thereby bridging the gap between academic knowledge and real-world applications.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 14, Issue 3, March 2025
DOI: 10.17148/IJARCCE.2025.14371

IV. RESULTS AND DISCUSSIONS

This section provides a candid overview of the prototype phase of ShodhX, a Retrieval-Augmented Generation (RAG)-based system designed for querying academic documents. As the system is still under development, this discussion focuses on its *architectural design*, *intended functionalities*, and *qualitative observations* from the development process. No empirical testing or user feedback has been conducted yet; instead, this analysis highlights the prototype's foundational framework, developmental challenges, and pathways for improvement.

A. Prototype Architecture and Design Goals



ShodhX - RAG based Chat with PDF

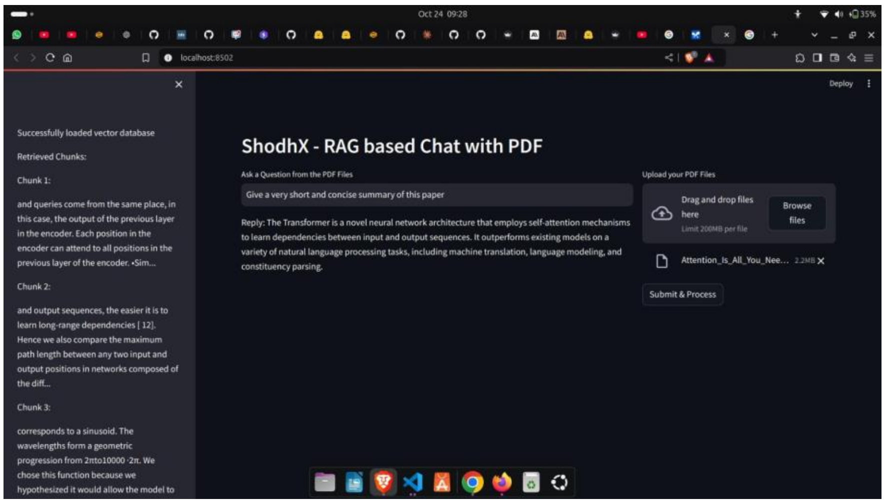
📚 Please upload and process PDF files to start asking questions.

📚 Source Legend

- 🟢 Directly used in answer (high similarity)
- 🟡 Partially used in answer (medium similarity)
- ⚪ Related context (lower similarity)

Upload your PDF Files
Drag and drop files here
Limit 200MB per file • PDF

Figure 2.1: ShodhX Home Screen



Successfully loaded vector database

Retrieved Chunks:

Chunk 1:

and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder. •Sim...

Chunk 2:

and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the diff...

Chunk 3:

corresponds to a sinusoid. The wavelengths form a geometric progression from 2πto10000 ·2π. We chose this function because we hypothesized it would allow the model to

ShodhX - RAG based Chat with PDF

Ask a Question from the PDF Files
Give a very short and concise summary of this paper

Reply: The Transformer is a novel neural network architecture that employs self-attention mechanisms to learn dependencies between input and output sequences. It outperforms existing models on a variety of natural language processing tasks, including machine translation, language modeling, and constituency parsing.

Upload your PDF Files
Drag and drop files here
Limit 200MB per file
Attention_Is_All_You_Nee... 2.2MB

Figure 2.2: Response from ShodhX

The current version of ShodhX is structured around three core components: the document processing pipeline, the query-handling workflow, and the user interface. The document processing pipeline follows a structured approach where documents are split into semantically coherent segments using rule-based thresholds such as fixed token limits. These segmented chunks are then converted into vector representations using a pre-trained language model, which enables efficient retrieval. A lightweight vector database stores these embeddings, allowing similarity-based searches.
For query handling, the system employs a retrieval module that uses cosine similarity to fetch relevant chunks based on user queries. The retrieved content is then synthesized into coherent answers by the language model. A basic front-end interface facilitates PDF uploads and displays responses alongside the retrieved chunks, ensuring transparency in the querying process.
The key design goals of ShodhX emphasize clarity in retrieved chunks to help users trace responses back to the source material while also balancing computational efficiency with response quality, making it suitable for academic use cases.

B. Qualitative Observations from Development

An analysis of the current prototype reveals several strengths and challenges. One of the key strengths is modularity, as

the decoupled design of retrieval and generation allows iterative improvements to individual components. Additionally, transparency is ensured by displaying retrieved chunks alongside the generated responses, helping users verify the credibility of the output. Early implementation results suggest foundational efficiency, as the pipeline can handle moderate-sized documents, typically between 20 to 30 pages, without significant latency.
However, certain limitations have been observed during development. In terms of chunking, the fixed token-based splitting approach occasionally disrupts semantic continuity by breaking text mid-paragraph. Additionally, non-textual elements such as tables, figures, and equations are excluded from processing, limiting the system's ability to provide a comprehensive analysis. Retrieval limitations have also been identified, as the system currently struggles with conceptual synthesis across multiple documents. For instance, queries requiring comparative analysis, such as "Compare methodologies in Papers A and B," are unsupported due to the system’s single-document focus. Moreover, ambiguous queries, such as "Explain the framework," may retrieve irrelevant sections if multiple frameworks are discussed within the document. The response generation process also presents constraints, as generated responses tend to mirror the structure of the source text rather than adapting to diverse query formats. Additionally, there is no mechanism to handle follow-up questions or contextual references, such as "Expand on the previous point."

C. Critical Limitations as a Prototype

Several scope restrictions limit the prototype’s current capabilities. The system is designed for single-document focus, meaning cross-referencing or multi-paper analysis is not yet feasible. Furthermore, it does not support non-textual elements such as diagrams and equations or multilingual documents. The contextual understanding of the model remains a challenge, as it retrieves textually similar chunks but struggles with implicit context. For example, a query like "Why was approach X chosen?" may retrieve sections mentioning approach X but fail to infer the rationale behind its selection. Lastly, scalability concerns remain unverified. While the system functions efficiently for small documents, its ability to process larger corpora exceeding 100 pages has yet to be tested.

D. Pathways for Iterative Refinement

Based on the challenges observed, several priorities have been identified for future refinement. Technical improvements will focus on enhancing chunking, retrieval, and response generation. Implementing adaptive chunking using semantic-aware splitting methods, such as leveraging section headings or topic modeling, can help preserve context. To enhance retrieval, integrating metadata such as section titles and keywords could improve chunk prioritization, while a hybrid approach combining lexical and semantic search may help address ambiguities. Improvements in response generation will involve fine-tuning the language model on academic text to enhance paraphrasing and conceptual synthesis, along with implementing post-processing techniques to filter redundant or low-confidence chunks.

Beyond technical refinements, feature expansion will aim to enhance system capabilities. Introducing multi-document support through a corpus-level vector index would allow users to query multiple academic papers simultaneously. Additionally, offering user customization options, such as adjustable chunk sizes and response verbosity, would improve usability for diverse academic needs. Incorporating context persistence through session memory would enable follow-up questions, allowing users to maintain context across multiple queries.
To ensure reliability, validation strategies will be implemented, including benchmarking against academic QA datasets such as SciQ to evaluate retrieval and generation accuracy. Conducting user testing in collaboration with researchers will help assess usability, response relevance, and potential pain points, further guiding iterative improvements.
In conclusion, ShodhX, in its current prototype form, demonstrates a functional RAG framework for academic document querying, with a modular design that permits incremental upgrades. While the system lacks empirical validation and advanced features, its architecture effectively addresses key requirements such as transparency, efficiency, and adaptability. However, challenges related to contextual gaps, scalability uncertainties, and rigid chunking highlight areas that require further refinement. Moving forward, development efforts will focus on enhancing semantic understanding, expanding scope, and preparing for rigorous testing. This phase serves as a foundational step toward building a robust tool that accelerates academic research.

V. FUTURE SCOPE

While the current prototype of ShodhX demonstrates promising capabilities in integrating research paper analysis with codebase retrieval via Retrieval-Augmented Generation (RAG), several areas remain ripe for further development. The future scope of this work is directed toward addressing the identified limitations while enhancing the application’s functionality and user experience.Future developments of ShodhX will focus on enhancing multimodal integration by incorporating images, tables, and diagrams through OCR and image embedding to provide a holistic understanding of research content. Advanced contextualization will be achieved by refining query disambiguation and implementing hierarchical response generation for more precise and structured answers. Scalability and real-time performance will be improved using vector indexing, distributed computing, and incremental learning for continuous model updates.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2025

DOI: 10.17148/IJARCCE.2025.14371

Personalized user experiences will be prioritized by introducing user profiling and interactive feedback loops to enhance retrieval relevance. Transitioning to a SaaS model will enable multi-user collaboration with cloud integration and API-driven workflows, making ShodhX more versatile and accessible. Additionally, identified research gaps will be addressed by domain-specific fine-tuning and expanding datasets to improve contextual retrieval and cross-modal knowledge transfer, bridging theoretical insights with practical applications. By focusing on these key areas, ShodhX aims to evolve into a comprehensive, interactive platform that not only overcomes existing limitations but also sets new standards for integrating academic research with practical implementation. These advancements will empower users to navigate the complexities of modern research and development with unprecedented efficiency, accessibility, and accuracy.

REFERENCES

[1]. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017.
[2]. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.
[3]. M. Chen, A. Radford, J. Wu, H. Jun, D. Luan, and I. Sutskever, “Evaluating Large Language Models Trained on Code,” OpenAI, 2021. [Online]. Available: https://openai.com/research/evaluating-large-language-models-trained-on-code
[4]. P. Lewis, B. Oguz, S. Riedel, H. Schwenk, and V. Karpukhin, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
[5]. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, and L. Zettlemoyer, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” arXiv preprint arXiv:1907.11692, 2019.
[6]. F. Feng, X. Wang, and H. Zhao, “CodeBERT: A Pre-trained Model for Programming and Natural Languages,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
[7]. T. B. Brown, B. Mann, N. Ryder, et al., “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, 2020.
[8]. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” arXiv preprint arXiv:1910.10683, 2019.
[9]. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” OpenAI, 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
[10]. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized Autoregressive Pretraining for Language Understanding,” in Proceedings of the 33rd International Conference on Machine Learning (ICML), 2019, pp. 5754–5764.
[11]. K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,” in Proceedings of the 8th International Conference on Learning Representations (ICLR), 2020.
[12]. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ Questions for Machine Comprehension of Text,” in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016, pp. 2383–2392.
[13]. J. Kossen, A. Stanisavljevic, J. Driessens, J. Nunkesser, and S. de Rooij, “Dynamic Memory-Augmented Networks for Few-Shot Learning,” in Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS), 2020.
[14]. M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, and L. Zettlemoyer, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 7871–7880.
[15]. S. Izacard and E. Grave, “Distilling Knowledge from Reader to Retriever for Question Answering,” in Proceedings of the 9th International Conference on Learning Representations (ICLR), 2021.
[16]. W. Fedus, B. Zoph, and N. Shazeer, “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity,” in Proceedings of the 2021 Conference on Neural Information Processing Systems (NeurIPS), 2021.
[17]. L. Dong, S. Xu, and F. Wei, “Unified Pre-training for Natural Language Understanding and Generation,” in Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS), 2020.
[18]. H. Yang, R. Jia, and P. Liang, “Extracting Training Data from Large Language Models,” in Proceedings of the 11th International Conference on Learning Representations (ICLR), 2023.