GURU - GenAi Unified Restwave Utilisation

Prof. Jaya Nag Mathur, Harsh Jha, Krishna Mistry, Harsh Gahukar, Aryan Sharma.
Department of Artificial Intelligence and Data Science
Dr. D. Y. Patil Institute of Technology, Pimpri. 411018

***Abstract*—This paper introduces Restwave, a high- A performance web framework designed as a fast alternative. To the highly popular Express framework. With Restwave provides a lightweight, scalable infrastructure for a special TCP server in handling JSON requests and responses efficiently, thus making It is best suited for building real-time applications and RESTful APIs. The major characteristics of Restwave include simplified server routing, Request-response data on middleware development every individual handling contributes to a quicker pace of scalability. With Restwave's robust architecture, we have developed GURU: an innovative educational platform powered by Retrieval-Augmented Generation (RAG) systems. GURU allows users, primarily teachers, to design their own chatbots systems based on customized datasets. Incorporation of Restwave's perform with GenAI technologies, real-time publication of GURU Adaptive learning, trend analytics, and educational resources. Observations: It has mentioned the architecture and the capability. Restwave was that one product which advanced GURU's capability Convert old learning spaces into AI-based ecosystems. Besides, the paper also reflects the It's possible for Restwave in scalable and responsive systems. Large Langage Models (LLMs) are pre-trained on large-scale corpora and Excel in many general natural language processing tasks. Such jobs include question answering (QA), among others. Their advanced language competencies, not to mention nothing of domain-specific and LLM suffers from hallucinations in knowledge-intensive tasks. knowledge cutoffs, and lack of knowledge attributions. Fine-tune LLMs' inductive knowledge to extremely specific domains is time consuming and a costly process. The retrieval-augmented generation process has been recently It seemed to be a strategy to fine-tune LLM responses. By relating them to some predefined ontology, one can demonstrate that While a knowledge graph ontology for RAG does improve , QA accuracy, taking into the consideration relevant sub-graphs that Preserve the same in an orderly form.**

***Index Terms*—Restwave, Express.js, Retrieval Augmented Generation (RAG), Knowledge Graph, Agents**

I. Introduction

There has been an enormous gold rush over pre-trained language models in the last years and reaching state-of-the-art performance for most challenges in NLP tasks. However, the same models are really limited for rather complex tasks like knowledge retrieval and generation. There seems to exist a new hybrid model, mainly the Retrieval-Augmented Generation model. In other words, it merges the parametric memory of the pre-trained general language tasks with the non-parametric memory, which may include a large corpus of Wikipedia, into a retrieval helper for augmenting language generation. The architecture contributes accuracy in responses by retrieving more relevant passages of external knowledge [8], [13], such as when applied to the task of question answering, fact verification, or abstractive question generation [4], [5].

The high-performance web framework, *Restwave*, would make building real-time applications easier. This should be especially considered in making NLP tasks easier to be successfully implemented. Compared with Express.js or other fast frameworks, Restwave can deal with massive requests for lightweight, scalable high-performance TCP server operations and mainly with the effective handling of JSON data requests and responses, making it an ideal solution for building RESTful APIs [1]. Using the facilities of Restwave, we built the tool *GURU*, through which one can produce customized chatbots via RAG systems. The entity GURU provides an RAG-based architecture through which an instructor will find it possible to prepare his or her personalized datasets, resulting in dynamic interactions for the user. Real-time data retrieval by Restwave makes easy providing GURU to the user with customized educational content [2], [3].

This paper discusses the architecture of Restwave, how it can be integrated to serve NLP tasks with RAG, and its usage in GURU. The work enlightens how the capabilities of Restwave form a powerful infrastructure for developing responsive knowledge-intensive applications both for the educator and developer.

The sheer and increasing volumes of data across large databases and information collections make it imperative that appropriate, specialized extraction of pertinent knowledge be done without an in-depth understanding of the underlying database resources. Recent progress in LLMs makes it possible to create innovations as well as utilize LLM-based chatbots for conversations trying to give answers. These models performed quite well for general needs of query processing, but the use of such models within particular domains is seriously constrained by related limitations. Similar to producing factually incorrect answers-so-called "hallucinations"; ignorance of recent news or events beyond what they have learned on-so-called "knowledge cutoff"; inability to attribute sources of information appropriately-so-called "implicit knowledge"; lack of specific technical knowledge suitable for a particular domain of expertise [7].

Knowledge graphs and vector stores will dramatically improve the context of LLMs, lessening their reliance on fine-tuning to specific domains. Knowledge graphs store factual

information in structured form and it can easily be accessed and used versus let one store documents in unstructured form and preserve the semantics of text. It will, therefore, make updated domain-specific information available to LLMs, which have known gaps in the generative models for a long time. Nonetheless, it's yet difficult to construct such a system. Actually, the development of domain-specific ontologies that are accurate and representative from which knowledge graphs and visualisations can be developed is challenging. Developing such specific text data that constitutes datasets for both the knowledge graphs and the visualisations is also demanding. All these are critical steps that allow the augmented LLMs to reliably generate quality responses relevant across various domains.

- We describe how to make a building frame General scientific corpora using a combination of texts Mining, Information Retrieval, Artificial Intelligence (AI), and human-in-the-loop methods.
- And now, we introduce the KG creation task, namely domain-specific VS ontology that uses observable metadata. and the raw text of the corpus of domain-specific open- A source of scientific articles, and underlying structure of it Obtained by nonnegative tensor/matrix factorization with automatic model selection.
- We show that the capabilities of the **Restwave**'s RAG-enhanced LLM system, that is, using prompting. With LLM agents, the right answer to scientific ques- conditions.

II. Related Works

The newest approaches to developing RAG-enhanced chatbot applications exploit vector databases full of unstructured text for the QA task. Although the idea of injecting KGs into AI systems is not novel, it is increasingly being used by researchers to improve the LLM's reasoning abilities at the same time as addressing the above reliability issues raised in Section. Of course, this comes with tremendous advantages but entails a lot of effort to infuse domain-specific knowledge into a chatbot. We briefly mention some prior work on common designs of chatbots and RAG pipeline with knowledge integration of domains, along with steps required for building KGs. Prioring education to teachers as use case.

A. KGs in RAG Pipelines

Advanced chatbots are such complex research efforts that prior work is unlikely to culminate in a fully engineered system like ours. Most attempts focus on enhancing specific elements of the RAG pipeline-for example, retriever design, query intent recognition, and KG reasoning. Our approach is similar to the techniques already used with chain-of-thought prompting over KGs ; combined with LLM-agents in a manner that enhances the reasoning capability [**?**] [6]. Apart from leveraging such cutting-edge techniques, we enhance our RAG pipeline by swapping our retrieval strategy to become K-Nearest Neighbors with Levenshtein metric to seed context search instead of cosine distance. We further engineer a "highly-specific" knowledge base tailored for the given QA tasks.

Although expensive and time-consuming, some of the work above include domain knowledge into their pipeline as part of RAG; most rely on existing KGs of fairly general medical text or do not provide details on their process for creating the dataset. We note our method is "highly-specific" since it was elicited subject matter expertise that informed our processes on how to curate and clean the dataset.

B. KG Development

At a minimum, anything that produces Knowledge Graphs needs to specify at least a corpus, define an ontology, and extract at least relevant entity-relation triplets from unstructured text.

Corpus Building. We first describe and expound upon what we mean by the term "highly-specific", and then our dataset collection approach. One of the hallmarks of our dataset collection is that we use unsupervised methods to decompose corpora into document clusters to finer specificity than the author-provided tags available on open access websites. This is quite far from prior approaches. We leverage latent-topic information that arises from our NMFk method to filter and select the best data for our knowledge base and prune documents based on citation information and embedding distances. Our text cleaning pipeline is informed by SME, thereby way beyond standard methods in including expert-derived rules for document cleaning, e.g, acronym and entity standardization.

KG Construction. Our ontology is SME-based traditional design with task-specific feature catching. We, however, are doing something novel by taking the latent information that we gain through our decomposition process in KG as entities. We are not limited to using just the current learning-based techniques that can be used both for entity and for relation extraction; we are using the very latest advances which use LLM-agents instead of other LLM prompting methods. This results in non-sparse KGs, with a very high average outdegree of entities. To our knowledge, no previous work uses all these methods as part of their KG construction process.

III. Methods

Over the last couple of years, pre-trained language models did revolutionize the face of natural language processing. However, the trade-off was that to achieve this state-of-the-art performance, they were not very strong in retrieval and generation abilities for complex, knowledge-intensive tasks. Not so long ago, hybrid models like Retrieval-Augmented Generation or RAG promised to deal with some or most of these shortcomings. This developed an architecture using parametric memory integrated, pre-trained on general language tasks, to support non-parametric memory so that it could retrieve a large text corpus like Wikipedia so that it could be retrieved to enrich the generation process of the language. The architecture is developed to facilitate more access to

retrieve more accurate and contextually relevant responses through retrieving relevant passages of external knowledge in questions like question answering, fact verification, and abstractive question generation.

Restwave is the ultra-fast web framework designed to boost real-time application development especially in NLP operations. It's a more adaptive variant compared to the traditional Express.js framework, with the engine optimized for lightweight and scalable TCP server operations, hence with great advantages in the management of JSON data requests and response mapping, which makes it uniquely fit for building a perfect RESTful API. With this great performance based on what Restwave has to offer, we actually came up with the design of an extraordinary platform called *GURU*. We enabled people to build extremely personalized chatbots out of advanced Retrieval-Augmented Generation systems. GURU, through its architecture based on RAG, allows teachers to design and curate specially personalized datasets based on their needs while engaging them with interactive, evolving experiences for the users. That aside, exceptional speed and scalability from Restwave are also integral enabler factors that would get real-time data retrieval for engaging GURU users with personalized educational content.

The architecture of Restwave and its integration with RAG for NLP tasks is presented in this paper. This architecture is then deployed within the scope of GURU and experiments will demonstrate how all capabilities of Restwave add up to form a very powerful infrastructure for developing responsive, knowledge-intensive applications, empowering both educators and developers.

This section outlines our framework, covering corpus extraction, KG ontology, VS construction, and the RAG process [7].

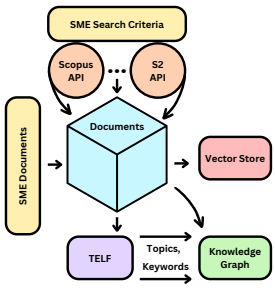
A. Platform Architecture

GURU's architecture consists of three primary components

- **Restwave Web Framework:**Is a high-performance, TCP server optimized for servicing JSON-based requests and responses [1]. It has to handle client requests as well as RAG in real-time [5]. This makes the web framework possible for GURU with multiple connections and data-intensive operations.
- **RAG System:**GURU has within it an AI engine in the form of the Retrieval-Augmented Generation system [10]. It combines a sequence-to-sequence model (be it transformer-based, as in the case of BART or T5) for text generation with dense retrieval mechanisms in accessing external knowledge sources such as educational databases or Wikipedia . With this hybrid approach, GURU is able to retrieve any relevant data and thus present contextually aware and accurate responses to user queries.
- **User Interface (UI):**This UI allows teachers to use the product, develop their own customized chatbots, import any datasets needed for the model, and also tailor the AI model according to learning goals. It is also possible to have real-time interaction between the AI-driven chatbot and end-users like students or other educators.

B. Domain-Specific Dataset [11]

The image shows an overview of our system. To write the body here, this important data represents the specific area where we want to create the body. We also extract tuples from the primary data to query APIs to search for related data. When we expand the collection starting from the primary data, we can also add data that is not directly related to the primary data. Pruning can be done in the following two ways:



- **Human-in-the-Loop Pruning**: SMEs manually review and select a few documents that match key terms. Here, we use UMAP to reduce the TF-IDF data matrix to two dimensions and allow SMEs to see the data in the middle of a group. SMEs can choose which files to delete
- **Automatic Pruning of Document Embeddings**: Based on the selection of SME in the previous step, we then remove data that is distant from the selected text and important data. The data is transformed into data using SCI-NCL, a BERT-based model fine-tuned by data science to evaluate the similarity of the content to the main data and information. Select SME. Information beyond similar topics is removed, ensuring that only relevant and relevant information related to the selected SME information is retained..

Despite human intervention, the system still has the capacity to share information. Group analysis allows workers to consider all data in a group, making analysis more efficient even when working with large files without being limited by group size.

We also implemented a pre-processing using a publicly available Python library **T**ensor **E**xtraction of **L**atent **F**eatures (**T-ELF**). The cleaning procedure consists of the following steps before the treatment:

- Remove non-English languages
- copyrights, and unnecessary content: stop words
- phrases, and email addresses
- ASCII characters and borders
- HTML tags, stop words, and numeric characters.

The pre-cleaning and modeling process is important in preparing the data for further analysis, thus increasing the quality and consistency of the data.

C. Dimension Reduction [11]

Extracting the hidden pattern from the dataset is done with the following method. First, prepare the data and create the necessary calculation process by following the steps below:

- Creation of the TF-IDF matrix, $\mathbf{X}$, of the cleaned corpus
- $\mathbf{X}$ is decomposed using nonnegative tensor factorization from **T-ELF** enhanced with our new binary search strategy , to classify document clusters.

T-ELF Let us extract unique features from the data. This method identifies hidden themes in the body by grouping the data according to shared themes. To avoid over/underfitting, an automated decision model is used where the final number of groups is determined by obtaining the highest score silhouette above the pressure determined using the binary discharge method. This method uses the binary search method k values, selectively skipping those k values that do not surpass the silhouette threshold. The search criterion for an optimal k is defined as $k_{\text{optimal}} = \max\{k \in \{1, 2, \ldots, K\} : S(f(k)) > T\}$, where $S(f(k))$ denotes the silhouette score of the k-th configuration and T the threshold. Importantly, even after identifying an initial "optimal" k, higher k values are visited regardless to ensure no better configuration is overlooked.

The factorization of $\mathbf{X}$ yields two non-negative factor matrices $\mathbf{W} \in \mathbb{R}_+^{m \times k}$ and $\mathbf{H} \in \mathbb{R}_+^{k \times n}$, ensuring $X_{ij} \approx \sum_s W_{is} H_{sj}$. Distribution of words over topics are captured in $\mathbf{W}$. The matrix $\mathbf{H}$ shows the topic Classification across files to identify the main content of each file after processing. Complete tensor and matrix factorization implementations of various algorithms are available at **T-ELF**

D. Knowledge Graph Ontology [11]

T-ELF and features from metadata are mapped to a set of head, center, and tail relationships to create direction triplets, which are then injected into Neo4j KG. knowledge and background features. The principle of knowledge in the knowledge graph comes from the data injected into the image along with the main characteristics. Each document node contains information such as DOI, title, abstract, and location API identification. Other indexes include author, publication year, Scopus category, affiliation, participating country, abstract, publisher, table of contents, keynote, references, instructions, and NER entities generated from spaCy's NER tags. These NER tags include events, people, places, products, organizations, and geographic locations. Interact with topics to provide thought-provoking questions and feedback to RAG.

E. Vector Store Assembly [11]

To develop RAG, we introduced a vector library for the raw data using Milvus. Additionally, a subset of the full text of the data is vectorized and placed in the vector storage. The full text (if any) is split into small paragraphs and each is given an ID number to indicate its location in the original document. These sentences are then vectorized into embeddings using the OpenAI text embedding-ada-002 model and transferred to the vector storage to support the RAG process.

RAG applications can query the vector repository to find relevant sentences from the entire text.

If the archive contains the necessary information, the LLM can answer the question and add the reference showing the full facts. If important information is needed, applications can continue their research via QA using metadata (e.g. DOI, author). This approach allows us to preserve the semantics of the original data and provide relevant answers.

F. Retrieval Augmented Generation [10]

Retrieval-Augmented Generation (RAG) is one of the State-of-the-Art method used to make the AI based answers more precise as well as relevant. This involves the collection of external information in relation to user queries and enables the model to provide answers that are more specific to the context. RAG integrates both generative artificial intelligence with retrieval processes, whereby the outputs are improved by providing access to suitable data sources at all times hence producing answers that are very specific and relevant to the context. The flexible structure of the RAG model caters for many applications thanks to its dependency on fresh and relevant information.

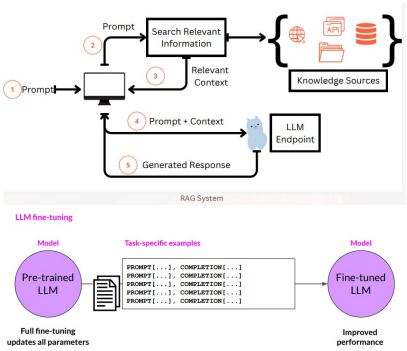


Fig. 1. User query routing overview.

The first step in the RAG process consists of a user query, which is vectorized by the Language Model (LLM) in the next step. These embeddings are then aimed to existing texts in the model's database to be recalled similar or relevant information. This information retrieved is appended to the user query, so that, the LLM can output more contextualized answers. The last step involves the LLM formulating an expansive response to the user query in the vernacular in such a manner that the response does not deviate much from the user's original question.

It is imperative to comprehend the particular requirements of the user in order to make RAG highly effective. In our case, we make use of a dataset that is specific to the domain, and the LLM combines both the domain specific and task specific fine tuned models. This lowers the cost of training but heightens the effectiveness of the models which are developed purposely for educational purposes both for learners as well as instructors. This finetuning approach

ensures that RAG system can provide accurate and reliable answers to educational questions, thanks to its high precision, which enhances its applicability to address specific user needs in this context.

To get the most out of RAG, it is important to clearly understand your users' problems. Our approach to RAG includes clear-cut types of queries that define the process of data retrieval and routing. Queries can be classified into general, document-specific, or all-inclusive, with each query being routed using the most suitable tools and tracking systems. Figure 1 presents the overall process of how queries are routed.

The RAG system communicates with several tools such as the ReAct Agent in the allocation and execution of queries. The ReAct agent is perhaps the most significant as it makes decisions, collects inputs, and analyzes outputs. The agent processing considers three nodes mainly: (1) ReAct agent, (2) tool executor, (3) end.

ReAct Agent Process: The agent node is the core of the ReAct diagram that encapsulates the LLM call. ReAct agents are responsible for gathering ideas, making decisions, and interpreting results. [12] The four available sections are:

a. Instructions
b. User query
c. Tool names, data
d. Tool Scratchpad

After the query is processed, the results from the RAG system are returned to the user. The ReAct agent effectively captures and interprets the required information, organizing it to address the user's needs with precision.

IV. Results

A. Dataset

Thirty key papers were identified based on their impact on web performance and educational applications. Extend the first option with network analysis/usage including: Monitor data in its initial form (Close topic). The final collection consists of 20 publications that have been carefully cleaned to ensure they are useful and helpful. The focus is on Restwave, a high-performance web framework designed [2] to replace traditional frameworks such as Express.js. Restwave features: Lightweight and scalable architecture: Designed for efficient processing of JSON requests and responses.

- **Core Features:**
- Simplified server routing.
- Middleware creation capabilities.
- Enhanced request-response data handling.

B. Vector Store

Many files are vectorized and fed into the Milvus vector repository. When asked questions about the framework, they are also vectorized using this model. Few percent of all documents have the full text vectorized in Milvus which is good. Each document has a DOI, as does the full text, including paragraph marks.

V. Conclusion

The GURU platform boasts a fantastic ability to go far beyond the education sphere, capable of fundamentally changing and innovating the ways in which different spheres can use artificial intelligence. Imagine a multifunctional tool that allows you to easily upload your very own unique data at the same time as providing you with the option to choose from a wide range of highly advanced AI models, such as LLaMA 3, GPT-3.5, or Claude. This new platform allows you to instantly generate a customized chatbot through an API, which you can then easily integrate into your website or any system you happen to be using. The type of flexibility being discussed here could prove to be a significant game-changer for various industries, including but not limited to healthcare, manufacturing, and customer support, where knowledge can play a critical role in operations and outcomes [8].

As an example, in the field of *healthcare*, a hospital might upload their medical records, guidelines, and procedural data and then decide which AI model to use to best interpret complex medical jargon— perhaps GPT-3.5, given its strong generation capabilities. GURU would create an API that could be used to develop an answering chatbot that could help patients with questions or assist in diagnosis or postoperative care, all customized to that individual hospital's dataset. The chatbot would possess a profound depth of knowledge that is essential for engaging in accurate and reliable interactions with patients, ensuring that all responses are both informative and trustworthy.

In *manufacturing*, the firm can upload relevant documents, which include equipment manuals, detailed maintenance logs, and valuable training data for its employees. Utilizing GURU, they can opt for a sophisticated model such as LLaMA 3, which was specifically optimized to retrieve intricate technical information efficiently. The innovative platform would then create an API to help the company in developing an ultra-interactive chatbot. This chatbot would be used to lead the technician through processes of repair, answer various questions on machinery problems, or even recommend adequate maintenance schedules for the smooth operation of their equipment. This type of tool is likely to significantly enhance efficiency on the shop floor by minimizing instances of downtime and decreasing their dependence on human assistance.

For the purpose of improving *customer support*, GURU has a novel approach that lets companies create and engage the most tailored chatbot for handling their company's specific questions related to product and services. During this exercise, a business can upload all its detailed FAQs database,

the history of previous interactions with customers and all the fine details of the products involved with useful insights.

Then, selecting a model like Claude, which is especially good at human-like conversation, they may develop a chatbot that delivers not only quick but also actual, reliable customer answers. This advanced chatbot, based on the company's own rich data, would be much more efficient than any over-the-counter similar product. Moreover, GURU's API would make this advanced chatbot integration into the company's website or mobile applications as easy and user-friendly as possible.

As we envision the future, it is clear that one of GURU's greatest and most exciting advantages will certainly be its amazing capacity to *automatically generate APIs*. After uploading your data successfully and after you decide on which model is suitable for you, GURU will quickly generate an API that you can seamlessly embed into any application or website. And not just that-you will have an option to host your data and model on AWS, Google Cloud, or Azure, so you would have control over the way your AI is deployed and in use.

The platform is designed to provide robust support for continuous updates of both datasets and AI models. Especially in the health sector, guidelines, practices, and knowledge evolve continuously and need frequent updates. With GURU, it will be extremely easy to upload new data when it comes in and swap out models as needed; this ensures your chatbot stays fresh and relevant. Additionally, multi-model support would empower organizations to use various AI models specifically tailored for different tasks, ensuring each receives the optimal level of AI help and expertise it requires.
In a nutshell, the future of GURU is no longer just within education, but something much more holistic, focusing on developing an incredibly flexible and incredibly potent platform, which will allow customization across various industries to create highly specialized and intelligent chatbots. Regardless of whether you are working in healthcare, manufacturing, or even customer service, GURU is going to give you the power to take control of your data while at the same time also enable you to develop very custom AI tools tailored for your precise needs and unique challenges.

In Conclusion: In few words, the GURU project has a potential to lead big change and impact in the learning sector through making available a high performance, AI-driven platform tailored for educators. Since the advanced With the integration of some of the latest technologies such as Restwave and GenAI, this proposed platform aims at giving all visitors real-time recommendations and highly-personalized resources. The educators will, therefore, stay always at the top, up-to-date with new information, as well as of essential tools that guarantee their success.

Restwave	**Express.js**
High-performance, real-time applications	General-purpose web framework
Ultra-low, optimized for low-latency	Higher latency under heavy load
Designed for high volumes with quick responses	Efficient but may slow with high traffic
Scales easily with minimal configuration	Scalable but may require load balancing
Real-time, data-intensive applications	Flexible for various web applications

TABLE I
COMPARISON OF RESTWAVE AND EXPRESS.JS

VI. REFRENCE

REFERENCES

[1] M. Kam and J. Dean, "A survey of web Application Frameworks," Journal of System and Software, 2006.

[2] A. Golding, "The benefits of using Frameworks," Apress, 2005.

[3] A. Alalfi et al., "A Survey of Web Verification and Testing methods," International Journal of Software Engineering and Knowledge Engineering, 2007.

[4] G. Di Lucca and A. Fasolino, "Web Application Testing: A systematic Literature Review," Journal of Systems and Software, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121214000223

[5] G. Amalfitano et al., "A classification framework for rich internet Application testing," IEEE Transactions on Software Engineering, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5623569

[6] A. Zeng, M. Liu, and R. Lu, "Agent Tuning: Enabling Generalized Agent abilities for LLM's," on Arxiv 2023. [Online]. Available: https://arxiv.org/abs/2310.12823

[7] C. Muhlroth and M. Grottke, "Artificial intelligence in innovation: How to spot emerging Trends and Technologies," 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9102438

[8] A. Agarwala, A. Purwar, and V. Rao, "Culturevo: The serious game of utilizing GenAi for enhancing cultural intelligence" on Arxiv 2024. [Online]. Available: https://www.arxiv.org/abs/2407.20685

[9] E. Yng and L. Shen, "Model merging in LLMs, MLLMs and Beyond: Methods, Theories, Applications and Opportunities," on Arxiv 2024. [Online]. Available: https://arxiv.org/abs/2408.07666

[10] P. Lewis, E. Perez, and M. Lewis, "RAG (Retrieval-Augmented-Generation) for Knowledge-Intensive NLP Tasks," on Arxiv 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[11] R. C. Barron, V. Grantcharov, S. Wanna, and M. E. Eren, "Domain-Specific Retrieval-Augmented Generation Using Vector Stores, Knowledge Graphs, and Tensor Factorization," on Arxiv 2024. [Online]. Available: https://arxiv.org/abs/2410.02721

[12] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in the Eleventh International Conference on Learning Representations, 2023. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X

[13] Weijian Xie, Xuefeng Liang, Yuhui Liu and more ,"WeKnow-RAG: An Adaptive Approach for Retrieval-Augmented Generation Integrating Web Search and Knowledge Graphs" on Arxiv 2024. [Online]. Available: https://arxiv.org/abs/2408.07611