Retrieval Augmented Generation With Knowledge Graphs

CS4.406 - Information Retrieval & Extraction

Authors

- Amogha A Halhalli (2021101007)
- Nikunj Garg (2021101021)
- Pranav Gupta (2021101095)

1 Naive Retrieval Augmented Generation

1.1 Limitations of LLMs

LLMs have revolutionised the way tasks like summarization and query answering is being done nowadays. But, it is observed that after a certain point of time, even LLMs are not able to give information due to lack of a proper information source. This is where RAG comes into picture.

1.2 Introduction to RAG

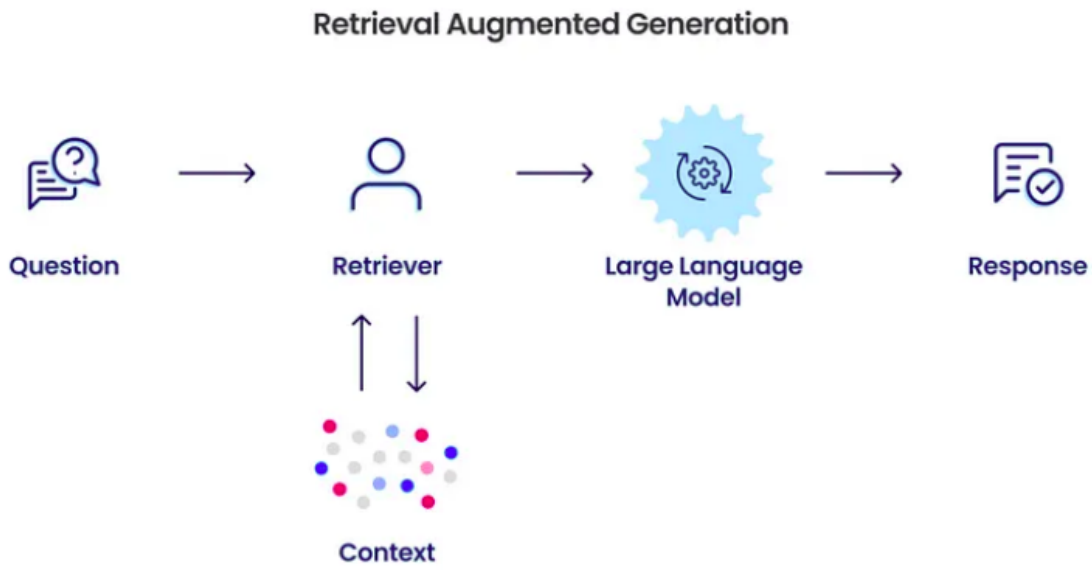


Figure 1: Working of RAG Stack.

RAG (Retrieval Augmented Generation) is a very popular information retrieval tool used across various NLP and Vision Tasks. A simple RAG stack is conceptually simple. Documents are turned into embeddings which are a mathematical representation of the text. The vectors are put into a vector database. When a query is asked, the query is turned into a set of mathematical representations, which is compared against the vectors in the database. The information that is most semantically similar by comparing the best fit mathematical representation is retrieved and returned to the LLM to construct the answer.

1.3 Components of RAG

Three primary parts make up a standard RAG system:

- **Retriever Component**: The retriever's job is to find relevant documents or pieces of information that can help answer a query. It takes the input query and searches a database to retrieve information that might be useful for generating a response.
- **Generator Component**: The generator is a language model that produces the final text output. It takes the input query and the contexts retrieved by the retriever to generate a coherent and relevant response.
- **Knowledge Base**: A database used to store the embeddings of the documents.

These are the underlying foundational models that RAG systems sit on top of, to return the appropriate, relevant information for LLMs to formulate answers on top of.

1.4 Working of RAG

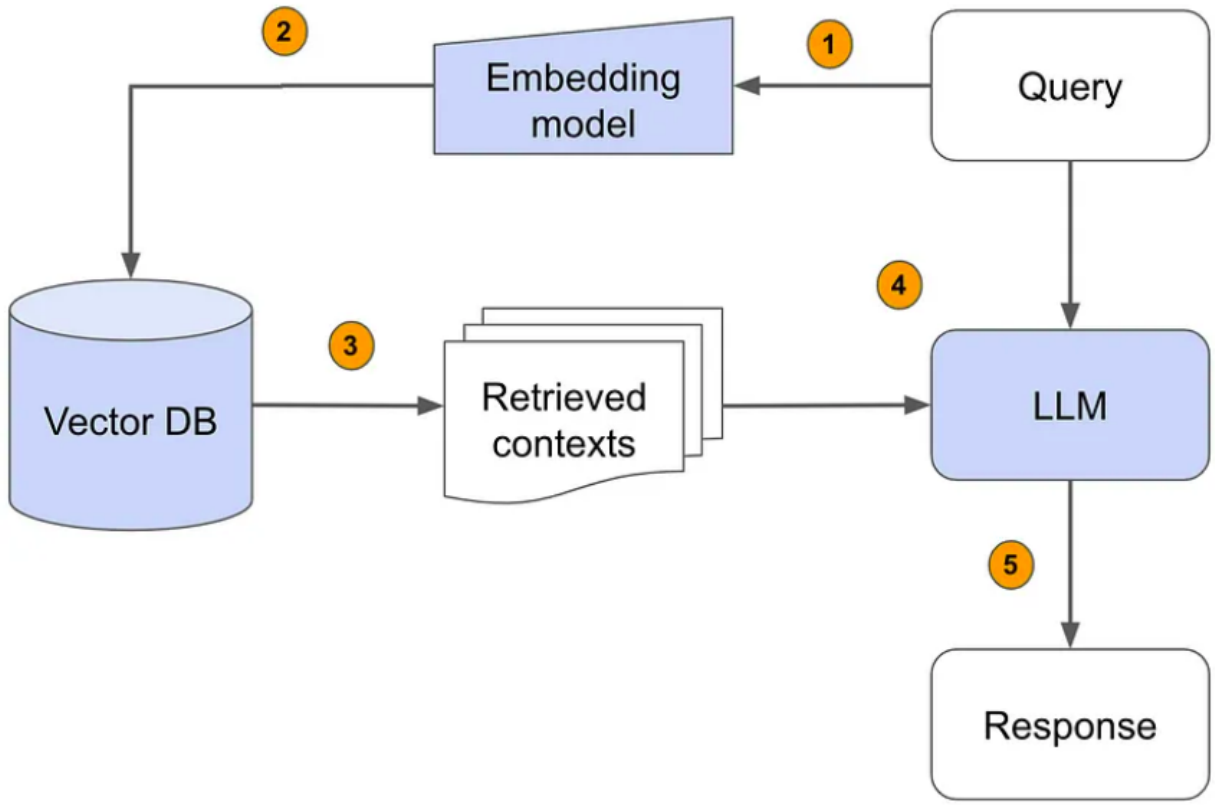


Figure 2: Workflow of RAG System.

1. **Query Processing:** Once the RAG application completes processing all the documents, the user is prompted to input a question. The query is combined with the prompt_text to create the full prompt. The query is then passed into a rag_chain pipeline.

2. **Embedding Model:** The embedding model used is OpenAIEmbeddings, which converts documents and queries into vector embeddings. Precomputed embeddings for the split documents are stored in the Chroma vector database.

3. **Vector Database (DB) Retrieval:** The vector database is managed using Chroma, which stores the precomputed embeddings of document chunks. It provides a retriever method which retrieves the most relevant contexts by comparing the query embedding with the stored embeddings.

4. **Retrieved Contexts:** The retriever outputs the relevant document chunks (contexts), which are formatted into a readable string using format_docs. The formatted contexts are part of the input passed to the language model through the RAG chain.

5. **LLM Response Generation:** The query and retrieved contexts are passed through a predefined RAG prompt template for instruction formatting. The gpt-4o-mini model from ChatOpenAI then generates the final response based on the formatted prompt and the combined context.

6. **Final Response:** The invoke method of the rag_chain generates the response. The final response is printed to the console for the user.

1.5 Results

```
Enter question (press Ctrl+D to exit):
Explain the concept of Attention
Attention is a mechanism that allows models to focus on different parts of the input sequence
when producing an output, enabling the modeling of dependencies regardless of their distance i
n the sequence. Self-attention, specifically, relates different positions within a single sequ
ence to compute its representation, and it has been effectively applied in various tasks like
reading comprehension and summarization. The Transformer model uniquely relies entirely on sel
f-attention instead of recurrent networks, allowing for better parallelization and improved pe
rformance in tasks such as translation.

Enter question (press Ctrl+D to exit):
Can SVD be used to compute Attention Scores in Transformers?
The information is not available in the provided context.

Enter question (press Ctrl+D to exit):
How does Dropout help in preventing Overfitting in Transformers?
Dropout helps prevent overfitting in Transformers by applying it to the output of each sub-lay
er before it is added to the sub-layer input and normalized. Additionally, dropout is applied
to the sums of the embeddings and positional encodings in both the encoder and decoder stacks.
 This regularization technique encourages the model to learn more robust features by randomly
deactivating a portion of the neurons during training.
```

Figure 3: Results using Naive RAG

2 Knowledge Graphs

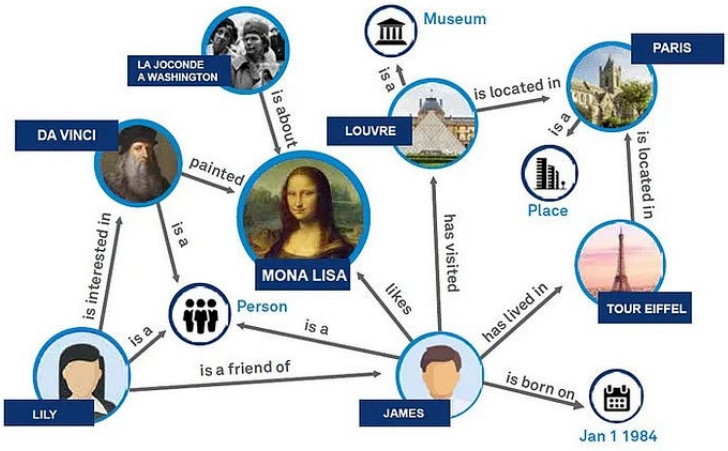


Figure 4: Knowledge Graphs Integration with RAG

Knowledge graphs is one of the components of the RAG Stack that help in realizing the full potential of RAG, and by extending Generative AI, by enhancing baseline RAG with additional context that can be specific to a company or domain.

- Knowledge graphs are fundamentally a collection of nodes and relationships. Nodes represent individual data points, while relationships define the connections between them. Each node can possess properties, which provide additional context or attributes about the node.
- This approach offers a flexible and intuitive way to model complex relationships and dependencies within data. It has often been described that knowledge graphs are a close way to mimic how the human brain thinks.
- Knowledge Graphs fit within RAG as data store of the semantic structure to retrieve vector chunks from

2.1 GraphRAG

Graph RAG is an advanced version of the RAG approach that incorporates graph-structured data. Instead of treating the knowledge base as a flat collection of documents, it represents information as a network of interconnected entities and relationships.

Limitations of Naive RAG overcomed by Graph RAG

- **Relational context**: It captures and utilizes the relationships between different pieces of information, providing richer context.

- **Multi-hop reasoning**: Graph structures enable the system to follow chains of relationships, facilitating more complex reasoning.
- **Structured knowledge representation**: Graphs can more naturally represent hierarchical and non-hierarchical relationships than flat document structures.
- **Efficiency**: Graph structures can make certain types of queries more efficient, especially those involving relationship traversal.

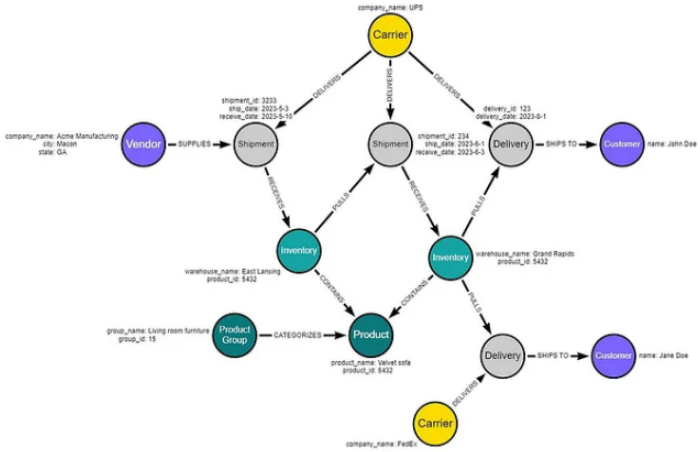
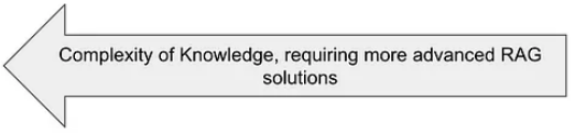


Figure 5: An example of a Knowledge Graph that acts as data store

2.2 Use Cases of GraphRAG



	Typically Networks	Typically Hierarchies	Typically Flat
Examples	- Transactions - Network-Based Standard Operating Procedures (SOPs) especially in Digital Twins, and common in Manufacturing / Construction - Social Networks - Legal documents	- Medical Diagnosis - One-Directional Standard Operating Procedures (SOPs) - Matching and Recommendation systems	- Customer Support / FAQ documents - Price lists - Singular news articles
Explanation	Network information structures are structures where each node in a network can link to any other node, creating a web-like structure. Retrieving the right information relies on navigating the network.	Hierarchical information structures are where information is structured in layers in a pyramid-like structure. Retrieving the right information relies on navigating a hierarchical structure.	Flat information structures are where the underlying information is clearly and fully represented discreetly within each chunk, and the main task within RAG is to retrieve the best discrete chunk for retrieval.

Figure 6: Working of GraphRAG

An example of putting GraphRAGs to use is linking clinical care practices to patient records in a healthcare context. For example, the external best practices for clinical care are being updated with new research and clinical trials. This information can be directly transmitted to patient records and their specific patient histories. The result is a direct communication of the latest safe, tested clinical care practices to the specific patients that are most likely to benefit from the new research.

Another example is customer support, which is traditionally not seen as a complex information system. One hypothetical problem in a retail customer support use case is to be able to ensure that the right opening hours are retrieved for the right store. However, limitations with semantic similarity mean that random opening hours are sometimes returned as the answer.

3 Problem Statement

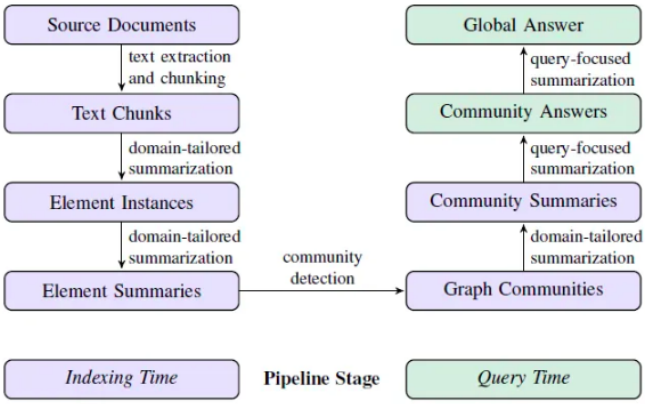
It is observed that depending upon the use case, either of RAG or GraphRAG can be used. RAG is suitable for places where semantic similarity needs to be captured based upon the embeddings of the documents, while GraphRAG can be used to capture intricate relationships existing between data and is suitable for tasks like multi-hop reasoning, Question answering, query focussed summarization, etc. Our Project aims to enhance the performance of existing QnA systems using knowledge graphs.

4 Relevant Resources

In order to build **Research Nexus** (our newly proposed methof), we turn to the existing literature for inspiration/idea regarding the current state of the art systems.

4.1 From Local to Global: A Graph RAG Approach to Query-Focused Summarization

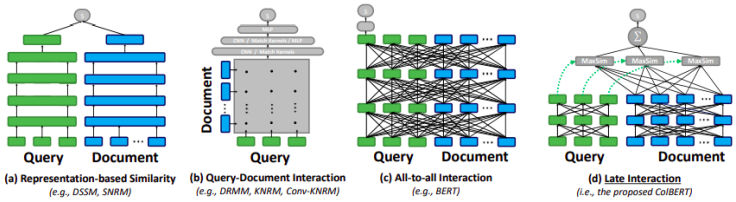
By leveraging community detection algorithms and a map-reduce-style processing pipeline, paper enables LLMs to generate comprehensive and informative summaries in response to complex, open-ended queries, even for datasets that exceed the context window limitations of traditional RAG approaches.



The Graph RAG Approach:

- **Source Documents to Text Chunks**: The process begins with source documents being split into smaller, manageable text chunks. This chunking is crucial for balancing the efficiency of processing with the quality of extracted information.
- **Text Chunks to Element Instances**: Next, the LLMs identify and extract instances of entities (such as people, places, organizations, etc.) and relationships between these entities from the text chunks. This extraction process involves several steps: LLMs scan the text chunks to identify and categorize entities, assigning attributes like names, types, and descriptions.
- **Element Instances to Element Summaries**: Following this, the summarized entities and relationships are used to construct a graph. Nodes represent entities, and edges represent relationships between them. The Leiden community detection algorithm is applied to detect communities within the graph.
- **Element Summaries to Graph Communities**: Following this, the summarized entities and relationships are used to construct a graph. Nodes represent entities, and edges represent relationships between them.
- **Graph Communities to Community Summaries** : Once the communities are detected, summaries are generated for each community. This involves creating comprehensive descriptions that encapsulate the key information within each community. .
- **Community Summaries to Community Answers to Global Answer**: Given a user query, the system uses the community summaries to generate partial answers. Each community summary is evaluated for its relevance to the query, and relevant community summaries generate partial answers independently and in parallel. These partial answers are then aggregated to form a final global answer that comprehensively addresses the query.

4.2 ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

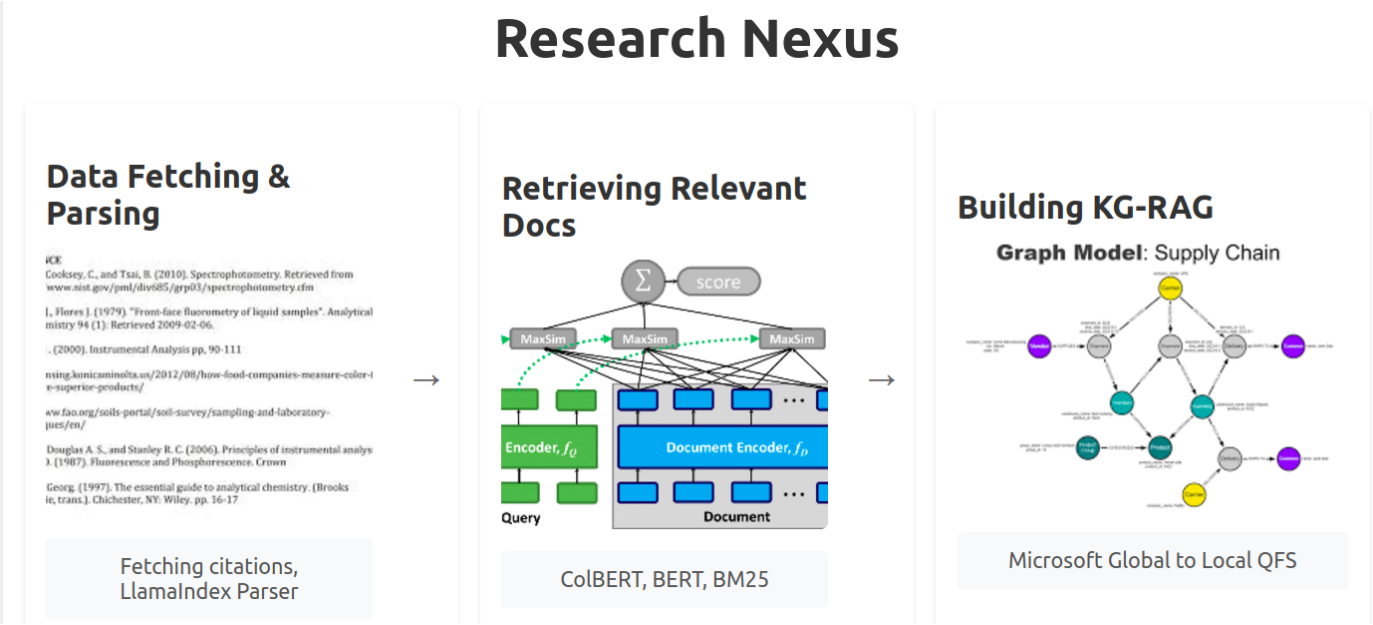


- Proposes a novel neural retrieval approach that enables late interaction between query and document terms
- Achieves BERT-level accuracy while being significantly more efficient
- Uses MaxSim operator to match contextualized query and document term representations
- Enables fine-grained contextual matching while maintaining computational efficiency through:
  - Independent encoding of queries and documents
  - Late-stage interaction between query-document term pairs
  - Efficient index-based search

Architecture balancing effectiveness and efficiency

- Using BERT to encode queries and documents independently
- Computing fine-grained similarity matrices between query-document term pairs
- Employing dimension reduction and quantization for practical deployment

5 Research Nexus (Our proposed model for enhanced research paper analysis and QnA)



5.1 Introduction

The exponential growth in academic publications necessitates automated systems for efficient research paper analysis and knowledge extraction.

We present a comprehensive model **Research Nexus** for advanced research paper analysis and advanced QnA. It provides a novel approach to streamline QnA tasks on research Papers.It addresses this challenge through a three-stage pipeline, combining citation analysis, intelligent document filtering, and knowledge graph-based reasoning.

Performs better than Naive Knowledge Graphs RAG in terms of **comprehensiveness** and **diversity** of the response generated.

System Architecture

5.2 Stage 1: Intelligent Data Ingestion

The first stage of our pipeline focuses on efficient data collection and preprocessing:

- **Citation Link Extraction**: **Gemini AI**-powered system identifies and extracts citation links from research paper to be analyzed

- **PDF parsing**: Conversion of PDFs to structured .txt file using **LLamaIndex** effectively. We observed **LLamaIndex Parser** processes docs much effectively as compared to **Naive pyPDF parser**.

5.3 Stage 2: Intelligent Information Retrieval

To increase the relevance of processed information and to reduce **hallucination** in Research Nexus we implemented a retrieval system to retrieve most relevant research papers out of all citations.

5.3.1 Model Comparison

We conducted extensive experiments comparing different retrieval approaches:

- **ColBERT**: Primary choice due to best performance in MRR@10 vs inference time performance
- **BM25**: Baseline traditional retrieval model
- **BERT**: Alternative Neural IR Approach

5.4 Stage 3: Knowledge Graph RAG Implementation

The final stage implements **Microsoft's Local to Global Query Focussed summarization** Paper:

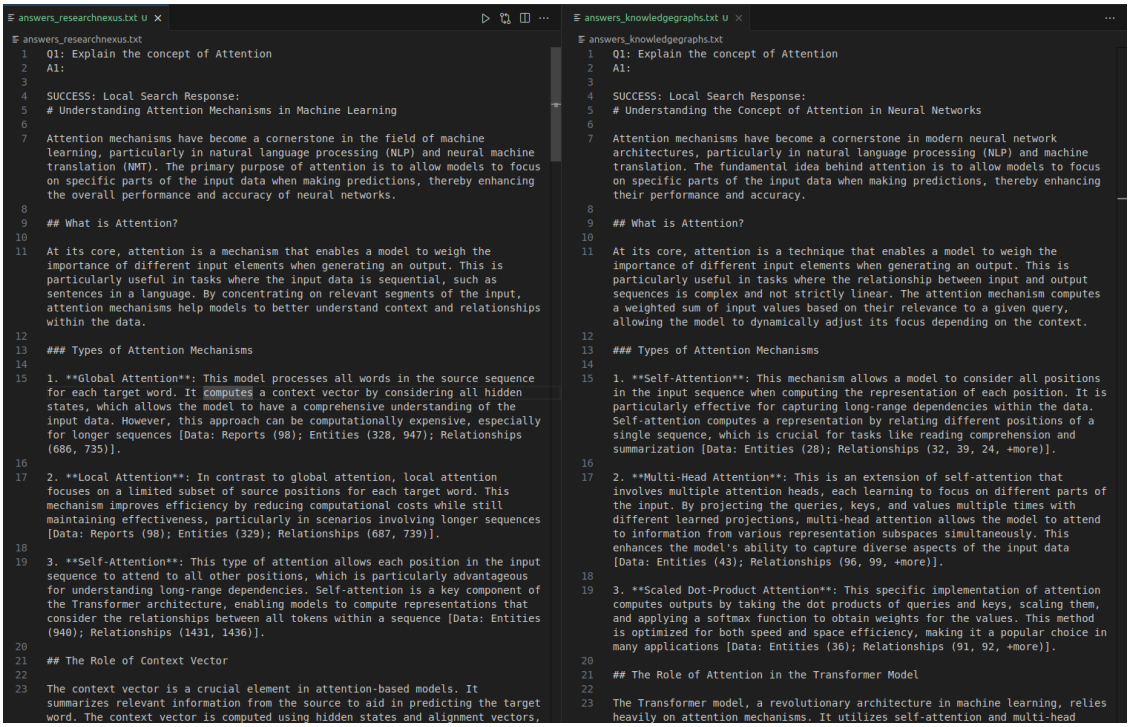
- **Knowledge Graph Construction**: Building a structured representation of information in documents
- **Question Answering on KG**: Implementing Question Answering on structured knowledge source.

5.5 Conclusion

Research Nexus is a 3-stage pipeline represents a significant advancement in research paper processing and analysis. The combination of intelligent data ingestion, selective information retrieval, and knowledge graph-based reasoning provides an advanced QnA approach. performs better than naive KG-RAG in terms of comprehensiveness and diversity of results being generated.

5.6 Results

Our system demonstrates significant performance improvements as compared to baseline methods implemented as shown:

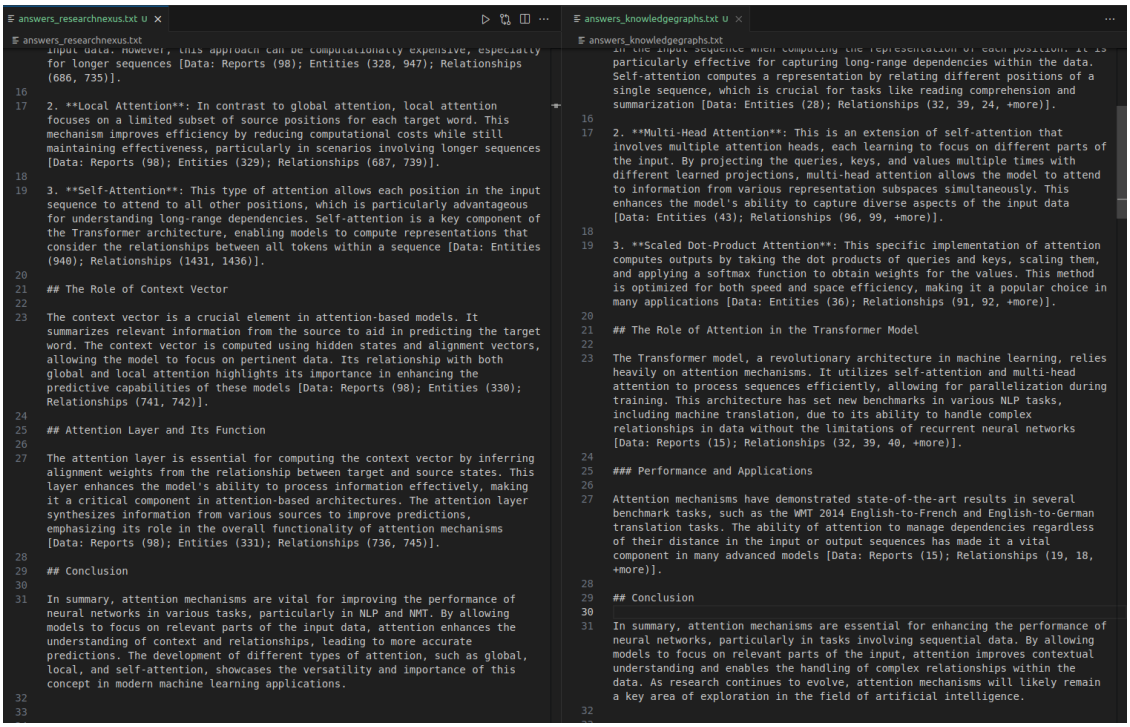


answers_researchnexus.txt U

```
Q1: Explain the concept of Attention
A1:

SUCCESS: Local Search Response:
# Understanding Attention Mechanisms in Machine Learning

Attention mechanisms have become a cornerstone in the field of machine
learning, particularly in natural language processing (NLP) and neural machine
translation (NMT). The primary purpose of attention is to allow models to focus
on specific parts of the input data when making predictions, thereby enhancing
the overall performance and accuracy of neural networks.

## What is Attention?

At its core, attention is a mechanism that enables a model to weigh the
importance of different input elements when generating an output. This is
particularly useful in tasks where the input data is sequential, such as
sentences in a language. By concentrating on relevant segments of the input,
attention mechanisms help models to better understand context and relationships
within the data.

### Types of Attention Mechanisms

1. **Global Attention**: This model processes all words in the source sequence
for each target word. It computes a context vector by considering all hidden
states, which allows the model to have a comprehensive understanding of the
input data. However, this approach can be computationally expensive, especially
for longer sequences [Data: Reports (98); Entities (328, 947); Relationships
(686, 735)].

2. **Local Attention**: In contrast to global attention, local attention
focuses on a limited subset of source positions for each target word. This
mechanism improves efficiency by reducing computational costs while still
maintaining effectiveness, particularly in scenarios involving longer sequences
[Data: Reports (98); Entities (329); Relationships (687, 739)].

3. **Self-Attention**: This type of attention allows each position in the input
sequence to attend to all other positions, which is particularly advantageous
for understanding long-range dependencies. Self-attention is a key component of
the Transformer architecture, enabling models to compute representations that
consider the relationships between all tokens within a sequence [Data: Entities
(940); Relationships (1431, 1436)].

## The Role of Context Vector

The context vector is a crucial element in attention-based models. It
summarizes relevant information from the source to aid in predicting the target
word. The context vector is computed using hidden states and alignment vectors,
```

answers_knowledgegraphs.txt U

```
Q1: Explain the concept of Attention
A1:

SUCCESS: Local Search Response:
# Understanding the Concept of Attention in Neural Networks

Attention mechanisms have become a cornerstone in modern neural network
architectures, particularly in natural language processing (NLP) and machine
translation. The fundamental idea behind attention is to allow models to focus
on specific parts of the input data when making predictions, thereby enhancing
their performance and accuracy.

## What is Attention?

At its core, attention is a technique that enables a model to weigh the
importance of different input elements when generating an output. This is
particularly useful in tasks where the relationship between input and output
sequences is complex and not strictly linear. The attention mechanism computes
a weighted sum of input values based on their relevance to a given query,
allowing the model to dynamically adjust its focus depending on the context.

### Types of Attention Mechanisms

1. **Self-Attention**: This mechanism allows a model to consider all positions
in the input sequence when computing the representation of each position. It is
particularly effective for capturing long-range dependencies within the data.
Self-attention computes a representation by relating different positions of a
single sequence, which is crucial for tasks like reading comprehension and
summarization [Data: Entities (28); Relationships (32, 39, 24, +more)].

2. **Multi-Head Attention**: This is an extension of self-attention that
involves multiple attention heads, each learning to focus on different parts of
the input. By projecting the queries, keys, and values multiple times with
different learned projections, multi-head attention allows the model to attend
to information from various representation subspaces simultaneously. This
enhances the model's ability to capture diverse aspects of the input data
[Data: Entities (43); Relationships (96, 99, +more)].

3. **Scaled Dot-Product Attention**: This specific implementation of attention
computes outputs by taking the dot products of queries and keys, scaling them,
and applying a softmax function to obtain weights for the values. This method
is optimized for both speed and space efficiency, making it a popular choice in
many applications [Data: Entities (36); Relationships (91, 92, +more)].

## The Role of Attention in the Transformer Model

The Transformer model, a revolutionary architecture in machine learning, relies
heavily on attention mechanisms. It utilizes self-attention and multi-head
```

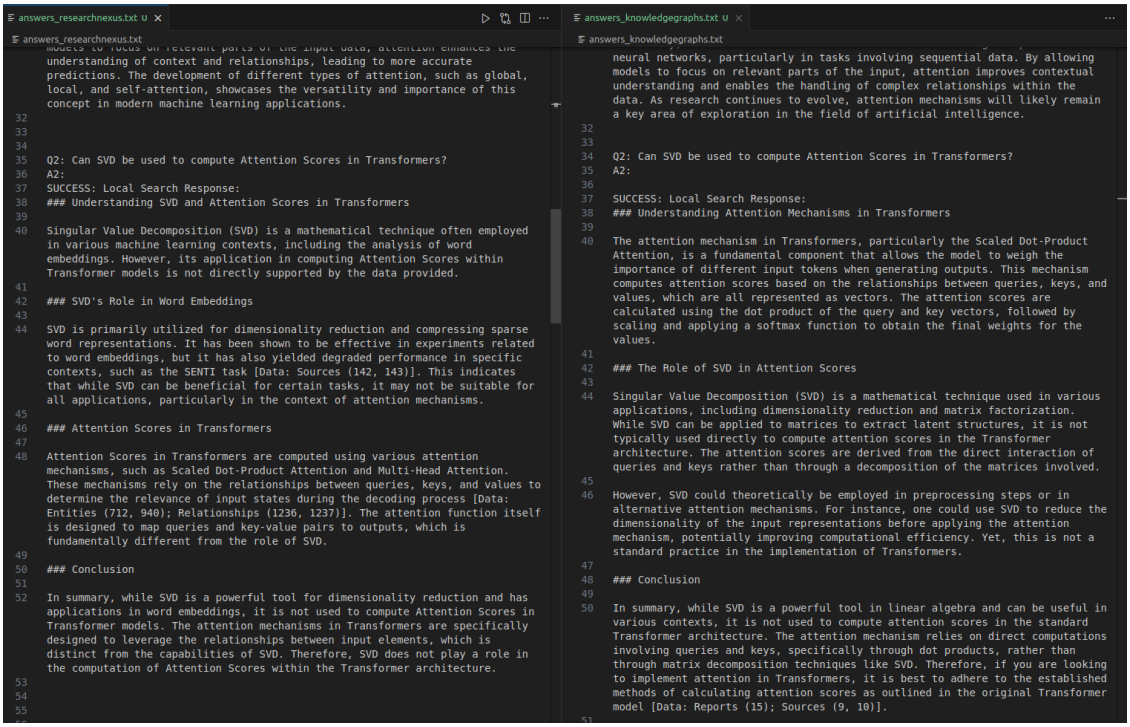


answers_researchnexus.txt U

```
input data. However, this approach can be computationally expensive, especially
for longer sequences [Data: Reports (98); Entities (328, 947); Relationships
(686, 735)].

2. **Local Attention**: In contrast to global attention, local attention
focuses on a limited subset of source positions for each target word. This
mechanism improves efficiency by reducing computational costs while still
maintaining effectiveness, particularly in scenarios involving longer sequences
[Data: Reports (98); Entities (329); Relationships (687, 739)].

3. **Self-Attention**: This type of attention allows each position in the input
sequence to attend to all other positions, which is particularly advantageous
for understanding long-range dependencies. Self-attention is a key component of
the Transformer architecture, enabling models to compute representations that
consider the relationships between all tokens within a sequence [Data: Entities
(940); Relationships (1431, 1436)].

## The Role of Context Vector

The context vector is a crucial element in attention-based models. It
summarizes relevant information from the source to aid in predicting the target
word. The context vector is computed using hidden states and alignment vectors,
allowing the model to focus on pertinent data. Its relationship with both
global and local attention highlights its importance in enhancing the
predictive capabilities of these models [Data: Reports (98); Entities (330);
Relationships (741, 742)].

## Attention Layer and Its Function

The attention layer is essential for computing the context vector by inferring
alignment weights from the relationship between target and source states. This
layer enhances the model's ability to process information effectively, making
it a critical component in attention-based architectures. The attention layer
synthesizes information from various sources to improve predictions,
emphasizing its role in the overall functionality of attention mechanisms
[Data: Reports (98); Entities (331); Relationships (736, 745)].

## Conclusion

In summary, attention mechanisms are vital for improving the performance of
neural networks in various tasks, particularly in NLP and NMT. By allowing
models to focus on relevant parts of the input data, attention enhances the
understanding of context and relationships, leading to more accurate
predictions. The development of different types of attention, such as global,
local, and self-attention, showcases the versatility and importance of this
concept in modern machine learning applications.
```

answers_knowledgegraphs.txt U

```
particularly effective for capturing long-range dependencies within the data.
Self-attention computes a representation by relating different positions of a
single sequence, which is crucial for tasks like reading comprehension and
summarization [Data: Entities (28); Relationships (32, 39, 24, +more)].

2. **Multi-Head Attention**: This is an extension of self-attention that
involves multiple attention heads, each learning to focus on different parts of
the input. By projecting the queries, keys, and values multiple times with
different learned projections, multi-head attention allows the model to attend
to information from various representation subspaces simultaneously. This
enhances the model's ability to capture diverse aspects of the input data
[Data: Entities (43); Relationships (96, 99, +more)].

3. **Scaled Dot-Product Attention**: This specific implementation of attention
computes outputs by taking the dot products of queries and keys, scaling them,
and applying a softmax function to obtain weights for the values. This method
is optimized for both speed and space efficiency, making it a popular choice in
many applications [Data: Entities (36); Relationships (91, 92, +more)].

## The Role of Attention in the Transformer Model

The Transformer model, a revolutionary architecture in machine learning, relies
heavily on attention mechanisms. It utilizes self-attention and multi-head
attention to process sequences efficiently, allowing for parallelization during
training. This architecture has set new benchmarks in various NLP tasks,
including machine translation, due to its ability to handle complex
relationships in data without the limitations of recurrent neural networks
[Data: Reports (15); Relationships (32, 39, 40, +more)].

### Performance and Applications

Attention mechanisms have demonstrated state-of-the-art results in several
benchmark tasks, such as the WMT 2014 English-to-French and English-to-German
translation tasks. The ability of attention to manage dependencies regardless
of their distance in the input or output sequences has made it a vital
component in many advanced models [Data: Reports (15); Relationships (19, 18,
+more)].

## Conclusion

In summary, attention mechanisms are essential for enhancing the performance of
neural networks, particularly in tasks involving sequential data. By allowing
models to focus on relevant parts of the input, attention improves contextual
understanding and enables the handling of complex relationships within the
data. As research continues to evolve, attention mechanisms will likely remain
a key area of exploration in the field of artificial intelligence.
```


answers_researchnexus.txt U
answers_researchnexus.txt
understanding of context and relationships, leading to more accurate predictions. The development of different types of attention, such as global, local, and self-attention, showcases the versatility and importance of this concept in modern machine learning applications.
Q2: Can SVD be used to compute Attention Scores in Transformers?
A2:
SUCCESS: Local Search Response:
Understanding SVD and Attention Scores in Transformers
Singular Value Decomposition (SVD) is a mathematical technique often employed in various machine learning contexts, including the analysis of word embeddings. However, its application in computing Attention Scores within Transformer models is not directly supported by the data provided.
SVD's Role in Word Embeddings
SVD is primarily utilized for dimensionality reduction and compressing sparse word representations. It has been shown to be effective in experiments related to word embeddings, but it has also yielded degraded performance in specific contexts, such as the SENTI task [Data: Sources (142, 143)]. This indicates that while SVD can be beneficial for certain tasks, it may not be suitable for all applications, particularly in the context of attention mechanisms.
Attention Scores in Transformers
Attention Scores in Transformers are computed using various attention mechanisms, such as Scaled Dot-Product Attention and Multi-Head Attention. These mechanisms rely on the relationships between queries, keys, and values to determine the relevance of input states during the decoding process [Data: Entities (712, 940); Relationships (1236, 1237)]. The attention function itself is designed to map queries and key-value pairs to outputs, which is fundamentally different from the role of SVD.
Conclusion
In summary, while SVD is a powerful tool for dimensionality reduction and has applications in word embeddings, it is not used to compute Attention Scores in Transformer models. The attention mechanisms in Transformers are specifically designed to leverage the relationships between input elements, which is distinct from the capabilities of SVD. Therefore, SVD does not play a role in the computation of Attention Scores within the Transformer architecture.
answers_knowledgegraphs.txt U
answers_knowledgegraphs.txt
neural networks, particularly in tasks involving sequential data. By allowing models to focus on relevant parts of the input, attention improves contextual understanding and enables the handling of complex relationships within the data. As research continues to evolve, attention mechanisms will likely remain a key area of exploration in the field of artificial intelligence.
Q2: Can SVD be used to compute Attention Scores in Transformers?
A2:
SUCCESS: Local Search Response:
Understanding Attention Mechanisms in Transformers
The attention mechanism in Transformers, particularly the Scaled Dot-Product Attention, is a fundamental component that allows the model to weigh the importance of different input tokens when generating outputs. This mechanism computes attention scores based on the relationships between queries, keys, and values, which are all represented as vectors. The attention scores are calculated using the dot product of the query and key vectors, followed by scaling and applying a softmax function to obtain the final weights for the values.
The Role of SVD in Attention Scores
Singular Value Decomposition (SVD) is a mathematical technique used in various applications, including dimensionality reduction and matrix factorization. While SVD can be applied to matrices to extract latent structures, it is not typically used directly to compute attention scores in the Transformer architecture. The attention scores are derived from the direct interaction of queries and keys rather than through a decomposition of the matrices involved.
However, SVD could theoretically be employed in preprocessing steps or in alternative attention mechanisms. For instance, one could use SVD to reduce the dimensionality of the input representations before applying the attention mechanism, potentially improving computational efficiency. Yet, this is not a standard practice in the implementation of Transformers.
Conclusion
In summary, while SVD is a powerful tool in linear algebra and can be useful in various contexts, it is not used to compute attention scores in the standard Transformer architecture. The attention mechanism relies on direct computations involving queries and keys, specifically through dot products, rather than through matrix decomposition techniques like SVD. Therefore, if you are looking to implement attention in Transformers, it is best to adhere to the established methods of calculating attention scores as outlined in the original Transformer model [Data: Reports (15); Sources (9, 10)].

6 Timeline and Workflow

- Month 1 – Understanding the Problem and Extensive Literature Review
  - Week 1: Gained in-depth insight into established techniques and tools in the field and have a fair idea about the differences in the methods implemented.
  - Week 2: Defined the Problem Statement, i.e. precisely outline the project's scope, objectives, and inherent challenges.
  - Week 3-4: Data Collection and Baseline Model Establishment done.
- Month 2 – Implementing Knowledge Graphs
  - Week 5-7: Constructed Knowledge graphs and stored them in neo4j database
  - Week 8-9: Explored different methods in order to overcome the computational limitations faced and refined the objectives of the project.
- Month 3 – Implementation of Method Research nexus, comparison, analysis and documentation
  - Week 10: Implementation of Data ingestion pipleline, scraping links and fetching citations papers
  - Week 11: Implementing of IR methods BM25, BERT, ColBERT to retreive relevant docd
  - Week 12: Implementation of nexus method and comparison between three different models and analysis.
  - Week 13: Comprehensive Documentation and Reporting: Prepare detailed reports and documents encapsulating the project's findings, methodologies, and outcomes.