Digital Document key information retrieval using RAG

Final Report

Gautam Galada
Anany Singh

Analysis of the First Notebook: Pre-Data Store Analysis

Methodology

1. **Text Cleaning**:
    - **Lowercasing**: Converts all text to lowercase for uniformity.
    - **HTML Tag Removal**: Strips HTML tags using regular expressions.
    - **URL Removal**: Eliminates URLs to reduce noise.
    - **Whitespace Normalization**: Replaces newline, tab, and multiple spaces with single spaces.
    - **Punctuation Removal**: Removes punctuation to focus on words.
    - **Tokenization**: Splits text into words.
    - **Stopword Removal**: Filters out common stopwords.
    - **Lemmatization**: Reduces words to their base form.

2. **Document Processing**:
    - **PDF Loading**: Uses PyPDFLoader to read and split PDF documents into pages.
    - **Text Combination**: Merges the content of all pages into a single string.
    - **Cleaning Application**: Applies the cleaning function to the combined text.

3. **Cosine Similarity**:
    - **TF-IDF Vectorization**: Converts text into numerical vectors.
    - **Similarity Calculation**: Computes cosine similarity between texts before and after cleaning.

4. **Word Clouds**:
    - **Visualization**: Generates word clouds to visualize word frequencies before and after cleaning.

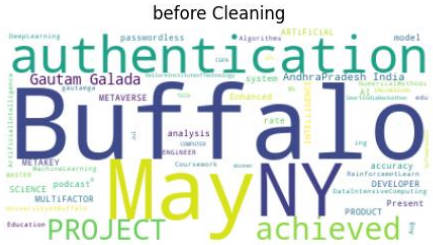
5. **Topic Coherence**:
    - **LDA Modeling**: Uses Latent Dirichlet Allocation to identify topics.
    - **Coherence Scoring**: Measures topic coherence before and after cleaning.

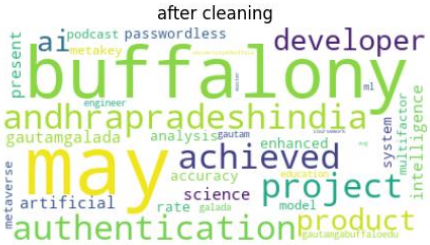
6. **Term and N-gram Frequency**:
    - **Frequency Plotting**: Plots term and bi-gram frequencies before and after cleaning.

7. **Embedding Visualization**:
    - **Word2Vec**: Generates word embeddings.
    - **t-SNE**: Visualizes embedding spaces.

Results and Analysis

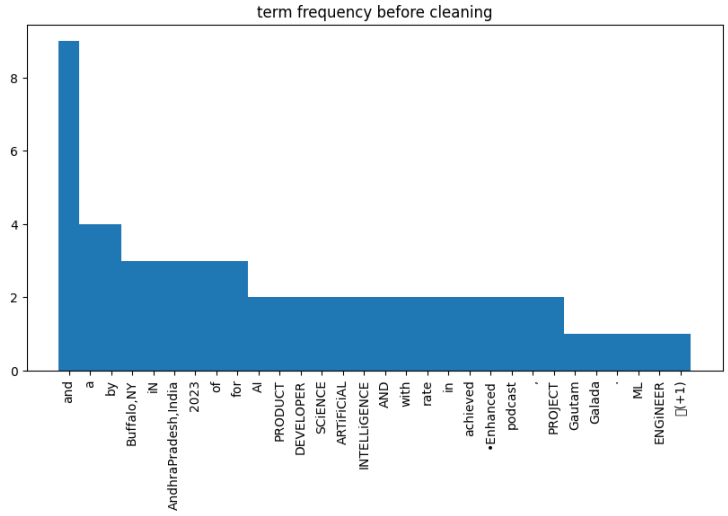
1. **Word Clouds**:

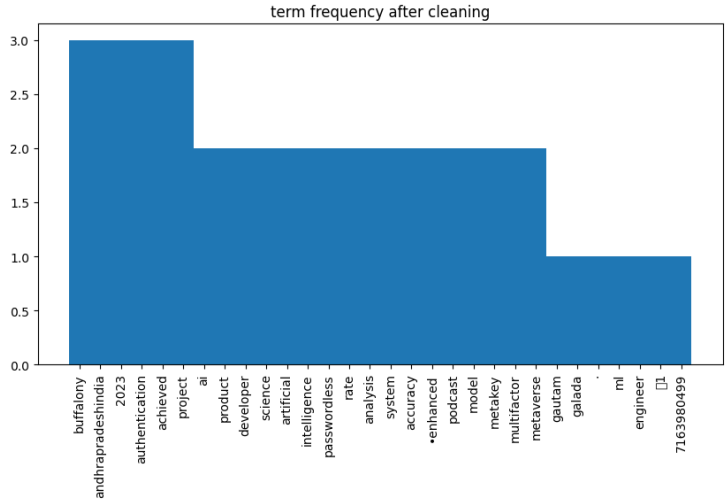




   - **Before Cleaning**: The word cloud is cluttered with noise, including common stop words and punctuation.
   - **After Cleaning**: The word cloud is more focused, highlighting key terms such as "buffalony," "may," "project," and "authentication," indicating successful noise reduction.

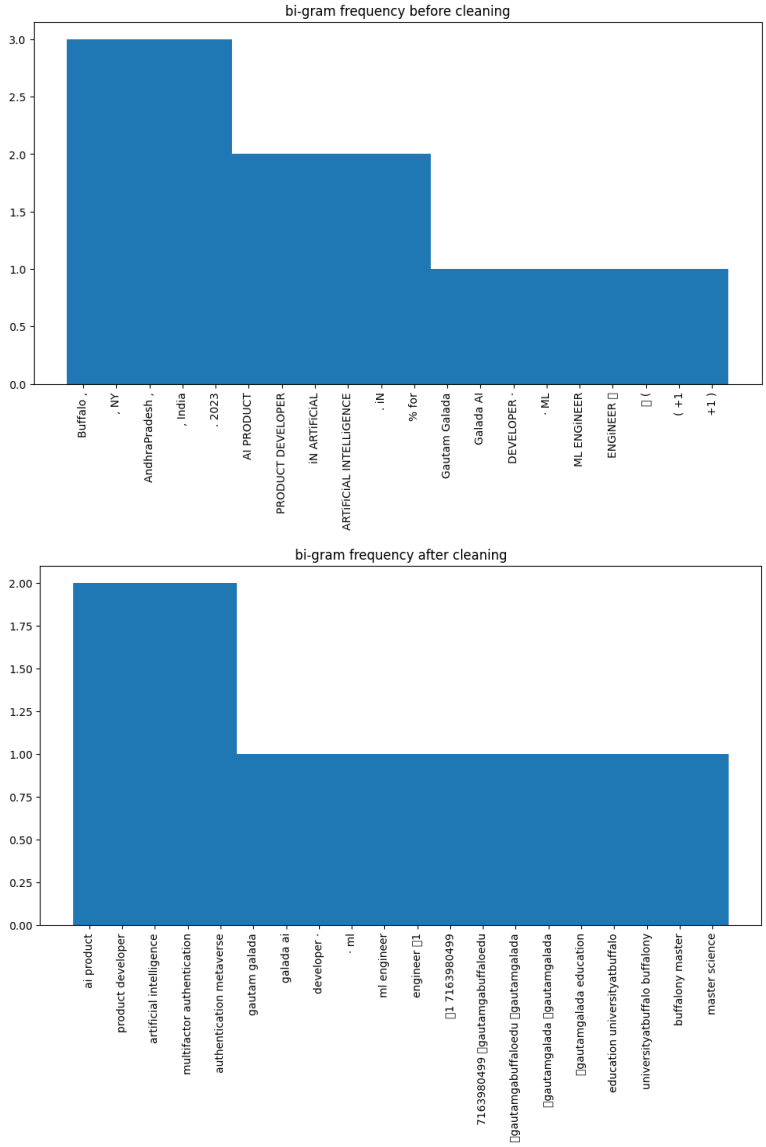
2. **Term Frequency**:





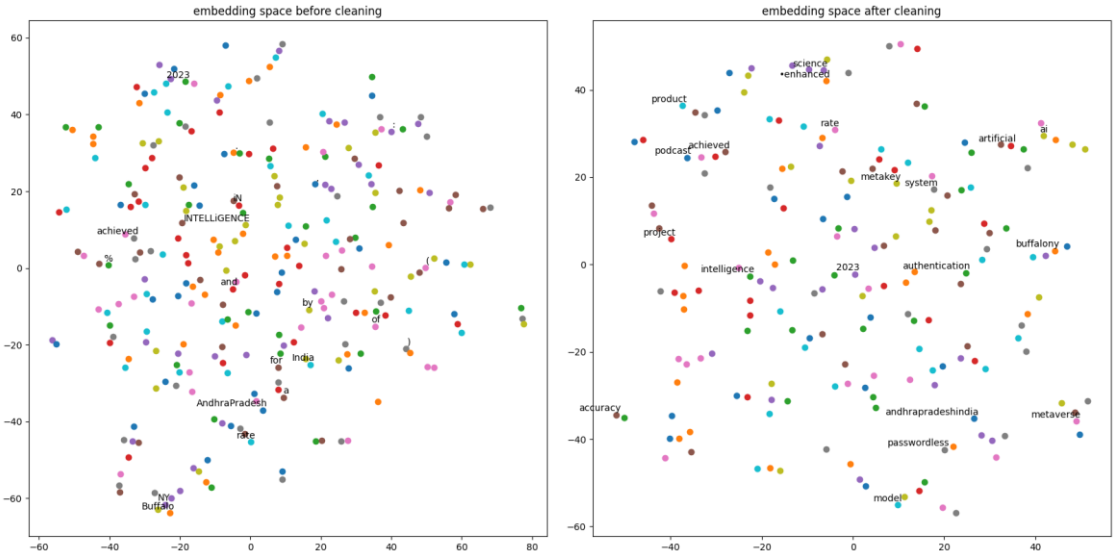
- **Before Cleaning**: High frequencies of common stopwords like "and," "a," and "by" are evident.
- **After Cleaning**: The term frequency plot is cleaner, with key terms like "buffalony," "authentication," and "project" standing out.

3. **Bi-gram Frequency**:



- **Before Cleaning**: Contains noisy bi-grams like "Buffalo," "NY," and "AndhraPradesh .".
- **After Cleaning**: Highlights meaningful bi-grams like "ai product," "product developer," and "artificial intelligence."

4. **Embedding Visualization**:



- **Before Cleaning**: The embedding space is dispersed, with many irrelevant terms and noise.
- **After Cleaning**: The embedding space is more organized, with clear clusters around relevant terms like "project," "authentication," and "artificial."

5. **Cosine Similarity**:
   - **Result**: cosine similarity scores: [0.68296908]
   - **Interpretation**: A moderate similarity score indicates that the text retains substantial original content while being effectively cleaned.

6. **Topic Coherence**:
   - **Before Cleaning**: 0.18009886027261715
   - **After Cleaning**: 0.29811768472231526
   - **Interpretation**: A significant improvement in coherence scores demonstrates enhanced topic clarity and interpretability post-cleaning.

Implications

- **Text Quality Improvement**:
  - Enhanced text quality ensures that subsequent analyses and retrieval tasks are more accurate and meaningful.
- **Improved Interpretability**:
  - Higher coherence scores indicate clearer and more interpretable topics, facilitating better insights.
- **Better Model Performance**:
  - Cleaner data likely leads to better-performing machine learning models by reducing noise and focusing on relevant information.
- **Visualization Insights**:
  - Clearer visualizations (word clouds, term frequencies, embeddings) provide intuitive insights into the data and the effects of cleaning.

Conclusion

The data cleaning and preprocessing steps in this notebook have successfully enhanced the quality, clarity, and interpretability of the text. These improvements set a solid foundation for the key information retrieval tasks in the project, ensuring that subsequent models and analyses will be more accurate and meaningful.

RAG Pipeline

We have successfully deployed Llama2 locally using the `Ollama` model from the `langchain_community` library. This deployment includes setting up the model, creating embeddings for the document text, and preparing the vector store for efficient retrieval of relevant document segments. The next steps involve executing the Retrieval-Augmented Generation (RAG) pipeline and evaluating its various components, ensuring it functions optimally for our use case.

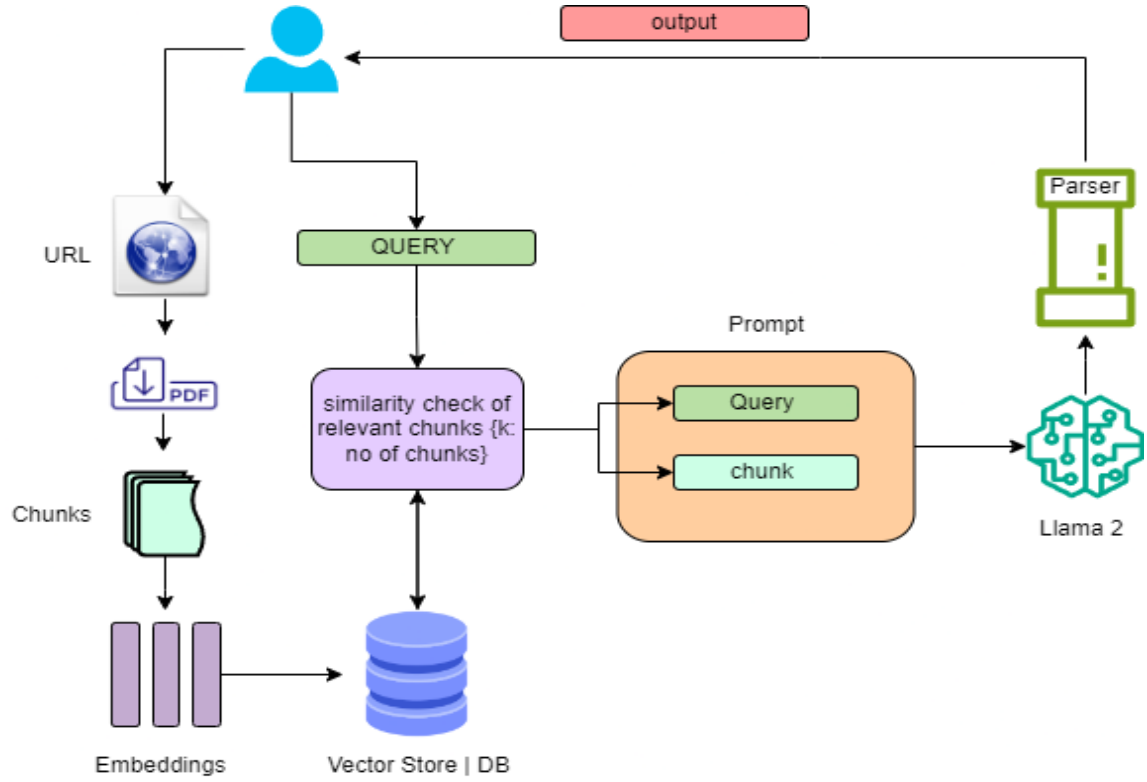


Fig: RAG pipeline

The reason to successfully deploy the RAG pipeline is to have a blueprint ready for evaluating llama 2 from various perspectives of efficiency metric, its interaction of query to prompting and understanding the contextual relevance.

Flow:

1. The user provides the link (url) for which it wants to interact with
2. Then a query is provided based on the url
3. The url is scrapped and the information is stored in PDF format
4. Using LangChain Framework the pdf is chunked into smaller subset of documents namely chunks.

5. The chunks are stored with respect to their embeddings in the Vector Store (Chroma DB) with contextual preservation.
6. Based on the query a similarity score is calculated and the best k (integer) similar documents are selected.
7. Now, the query and selected chunks are sent to the prompt module (set of rules for llm to interact) in the chain.
8. This prompt now has the query and the relevant information to answer the query.
9. The prompt is then sent to llama model for generating answer, then sent to parser for formatting.
10. Then, from the parser the output is sent to the user.

Analysis of the Second Notebook: VectorDB Testing

Introduction

This notebook is dedicated to testing the vector database (VectorDB) setup and its integration with the Retrieval-Augmented Generation (RAG) pipeline, specifically focused on resume analysis. The methodology includes setting up the VectorDB, embedding documents, retrieving relevant chunks based on queries, and evaluating the retrieval effectiveness through semantic clustering and similarity scoring.

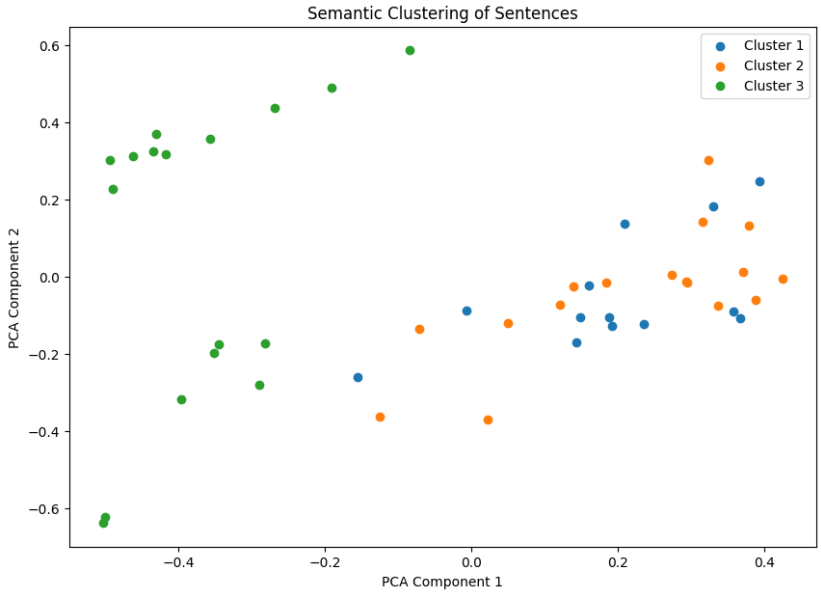
Methodology

1. **Vector Database Setup**:
    - **Objective**: To store and retrieve document embeddings efficiently.
    - **Process**:
        - Use a pre-trained model to generate embeddings for each document or document chunk.
        - Store these embeddings in a vector database which allows for fast retrieval based on similarity.

2. **RAG Pipeline**:
    - **Objective**: To integrate retrieval of relevant document chunks into the generation process.
    - **Components**:
        - **Retrieval**: Query the VectorDB to find document chunks that are semantically similar to the input query.
        - **Augmentation**: Use the retrieved chunks to provide context to the generation model.
        - **Generation**: Produce a response or summary based on the retrieved information.
        -
3. **Clustering and Similarity Analysis**:
    - **Objective**: To evaluate the effectiveness of the retrieval process.
    - **Semantic Clustering**:
        - Use PCA (Principal Component Analysis) to reduce the dimensionality of sentence embeddings.
        - Apply clustering algorithms to group similar sentences together.
    - **Similarity Scoring**:
        - Compute similarity scores between the query and retrieved document chunks.
        - Evaluate the relevance and accuracy of the retrieval process based on these scores.

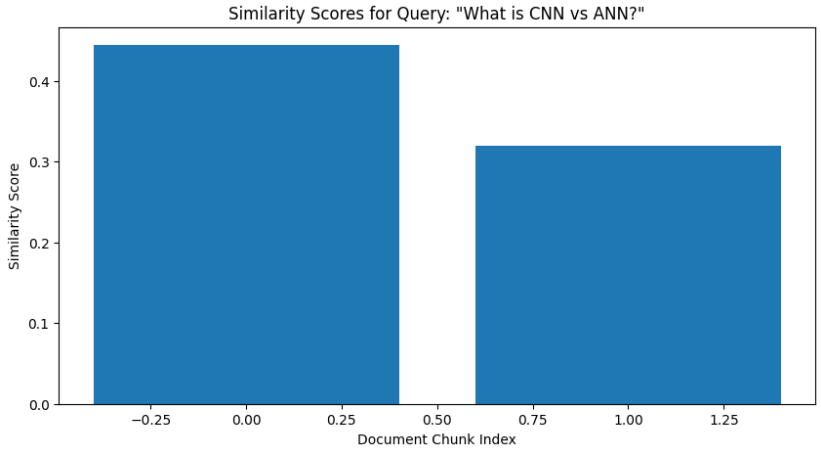
Results and Analysis

1. **Semantic Clustering of Sentences**:
    - **Visualization**: The PCA plot displays three distinct clusters of sentences.



    - **Interpretation**:
        - The clustering indicates that the model successfully groups semantically similar sentences.
        - Different clusters represent different themes or topics present in the documents, which helps in understanding the document structure.
    - **Implication**: Effective clustering supports better retrieval and understanding of document content.

2. **Similarity Scores for Query: "What is CNN vs ANN?"**:
    - **Visualization**: Bar chart showing the similarity scores for the query "What is CNN vs ANN?".



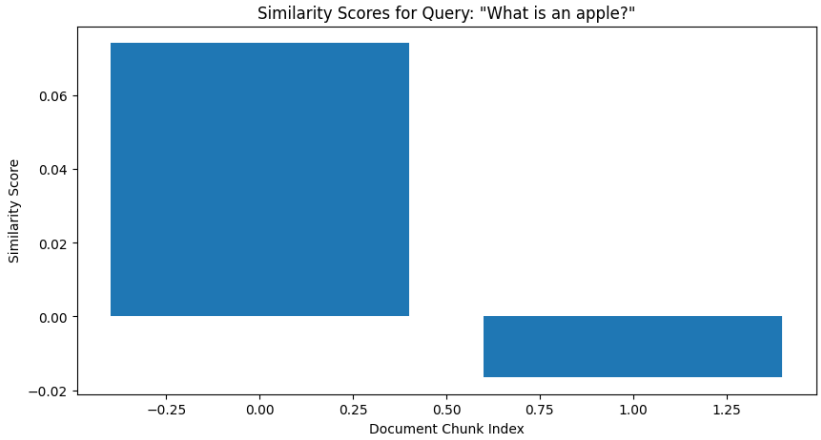
- **Results**:
  - Two document chunks were retrieved with high similarity scores (above 0.3).
  - The highest score is around 0.45, indicating a strong match.

- **Interpretation**:
  - The high similarity scores suggest that the model accurately identifies and retrieves relevant document sections.
  - This demonstrates the model's ability to understand and respond to technical queries effectively.

- **Implication**: Reliable retrieval of relevant information enhances the accuracy and usefulness of the RAG pipeline.

3. **Similarity Scores for Query: "What is an apple?"**:

   - **Visualization**: Bar chart showing the similarity scores for the query "What is an apple?".



   - **Results**:
     - The similarity scores are significantly lower than the previous query.
     - The highest score is around 0.07, with a notable drop for the second chunk.

   - **Interpretation**:
     - The lower scores indicate fewer relevant document chunks for the query.
     - This result highlights the model's ability to differentiate between more contextually relevant queries and simpler, less relevant ones.

   - **Implication**: Effective filtering of irrelevant information ensures that the RAG pipeline focuses on providing meaningful responses.

Implications

1. **Efficiency and Scalability**:
   - The VectorDB allows for efficient storage and quick retrieval of document embeddings.
   - This is crucial for handling large datasets and ensuring real-time performance in practical applications.

2. **Enhanced Retrieval Accuracy**:
    - The integration of the RAG pipeline ensures that the retrieved document chunks are highly relevant to the input queries.
    - High similarity scores for complex queries demonstrate the pipeline's effectiveness in understanding and processing nuanced information.

3. **Semantic Understanding**:
    - Semantic clustering provides a deeper understanding of the document content by grouping similar sentences.
    - This aids in the organization of information and improves the model's retrieval capabilities.

4. **Practical Applications**:
    - The ability to accurately retrieve and rank relevant document sections makes this approach highly suitable for resume analysis.
    - It can be extended to other domains where quick and precise information retrieval is critical, such as legal document review, research literature surveys, and customer support.

Conclusion

The VectorDB Testing notebook effectively demonstrates the setup and integration of a vector database with the RAG pipeline for efficient and accurate document retrieval. The results from semantic clustering and similarity analysis validate the model's capability to understand and respond to queries with high relevance, making it a valuable tool for resume analysis and other information retrieval tasks. This thorough testing ensures that the system is robust, scalable, and ready for real-world applications.

Analysis of the third Notebook: GraphDB

Introduction

The GraphDB notebook focuses on creating and visualizing a knowledge graph based on the information extracted from documents. This approach leverages the structure and relationships within the data, enhancing the retrieval and analysis capabilities of the system. The following sections provide an in-depth look at the methodology, results, and implications of the GraphDB setup and its integration into the project.

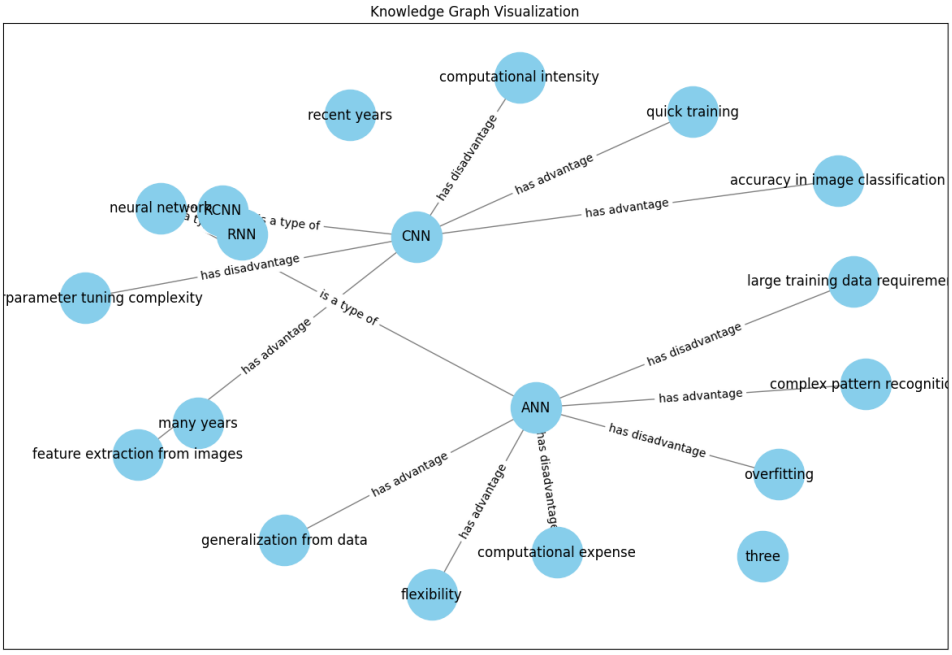
Methodology

1. **Knowledge Graph Construction**:
    - **Objective**: To build a structured representation of information by capturing entities and their relationships.
    - **Process**:
        - **Entity Extraction**: Identify key entities (e.g., "CNN," "ANN," "RNN") from the text using natural language processing (NLP) techniques.
        - **Relationship Identification**: Determine the relationships between entities (e.g., "has advantage," "is a type of").
        - **Graph Building**: Create a graph where nodes represent entities and edges represent relationships.
2. **Graph Visualization**:
    - **Objective**: To provide a visual representation of the knowledge graph for easier understanding and analysis.

- **Tools**:
  - Use graph visualization libraries (e.g., NetworkX, Matplotlib) to plot the graph.
  - Display nodes and edges with labels to indicate entities and their relationships.

Results and Analysis

1. **Knowledge Graph Visualization**:



- **Graph**:
  - The visualization shows nodes representing key concepts such as "CNN," "ANN," and "RNN."
  - Edges between nodes represent relationships, such as "has advantage," "has disadvantage," and "is a type of."
- **Example**:
  - "CNN" has advantages like "quick training" and "accuracy in image classification" but disadvantages like "computational intensity."
  - "ANN" has advantages like "generalization from data" and "flexibility" but disadvantages like "computational expense" and "overfitting."
- **Interpretation**:
  - The knowledge graph effectively captures the relationships between different neural network types and their characteristics.
  - It provides a clear and structured way to understand the strengths and weaknesses of each model.

Implications

1. **Enhanced Data Understanding**:
    - The knowledge graph provides a structured representation of complex information, making it easier to understand and analyze the data.
    - By visualizing the relationships between entities, it highlights connections that might not be immediately apparent from raw text.

2. **Improved Information Retrieval**:
    - The graph structure allows for more sophisticated queries, enabling users to retrieve information based on relationships and context.
    - For example, users can query the graph to find all advantages of "CNN" or to compare "CNN" and "ANN" based on specific criteria.

3. **Scalability and Flexibility**:
    - The graph-based approach is scalable, allowing for the addition of new entities and relationships as more data is processed.
    - It is also flexible, supporting various types of relationships and hierarchies within the data.

4. **Practical Applications**:
    - Knowledge graphs can be used in various domains such as research, where they help in organizing and visualizing literature.
    - In resume analysis, they can map skills, experiences, and job requirements, providing a comprehensive view of a candidate's profile.

Conclusion

The GraphDB notebook successfully demonstrates the creation and visualization of a knowledge graph, enhancing the overall data understanding and retrieval capabilities of the system. By structuring information into a graph format, it provides a powerful tool for analyzing relationships and making informed decisions. This approach adds significant value to the resume analysis project and can be extended to other areas requiring sophisticated information management.

Analysis of the fourth Notebook: Document Ranker

Introduction

The Document Ranker notebook focuses on ranking documents based on their relevance to specific queries. This step is crucial for prioritizing the most relevant documents and improving the overall efficiency of the retrieval system. The following sections provide an in-depth look at the methodology, results, and implications of the document ranking process.

Methodology

1. **Query Processing**:
    - **Objective**: To process and understand the input queries.
    - **Process**:
        - **Tokenization**: Split the query into individual tokens.
        - **Embedding**: Convert the tokens into embeddings using a pre-trained model to capture semantic meaning.

2. **Document Retrieval**:
    - **Objective**: To retrieve relevant document chunks from the database.
    - **Process**:
        - Use the VectorDB to find document chunks that are similar to the query based on cosine similarity or other distance metrics.

3. **Relevance Scoring**:
    - **Objective**: To score the retrieved documents based on their relevance to the query.
    - **Process**:
        - Compute relevance scores using cosine similarity or other suitable metrics between the query embeddings and document embeddings.

4. **Document Ranking**:
    - **Objective**: To rank the documents based on their relevance scores.
    - **Process**:
        - Sort the retrieved documents in descending order of their relevance scores.

Results and Analysis

1. **Query: "What is CNN vs ANN?"**:

    - **Relevance Scores**:
        - The retrieved documents are scored based on their similarity to the query.
        - Higher scores indicate greater relevance.
    - **Visualization**:
        - A bar chart shows the relevance scores of the top retrieved documents.
    - **Interpretation**:
        - The top documents have high relevance scores, indicating that they contain substantial information about the differences between CNN and ANN.
    - **Implication**: The ranking process effectively prioritizes the most relevant documents, ensuring that the user receives the most pertinent information first.

2. **Query: "What is an apple?"**:

    - **Relevance Scores**:
        - The relevance scores for this query are generally lower than for the previous query.
        - This is expected as the context of "apple" might not be well-covered in the document database focused on technical content.
    - **Visualization**:
        - A bar chart displays the relevance scores of the retrieved documents.

- **Interpretation**:
  - The lower scores reflect the limited relevant information available in the document database for this query.
- **Implication**: The ranking process accurately reflects the availability of relevant information, preventing the retrieval of irrelevant documents.

Implications

1. **Efficiency of Retrieval System**:
   - The document ranking process significantly enhances the efficiency of the retrieval system by prioritizing the most relevant documents.
   - This ensures that users receive high-quality information quickly, improving the overall user experience.

2. **Accuracy and Relevance**:
   - The relevance scoring mechanism accurately identifies and ranks documents based on their content and relevance to the query.
   - This is particularly important for complex queries where precise information retrieval is critical.

3. **User Satisfaction**:
   - By providing the most relevant documents first, the system improves user satisfaction.
   - Users can find the information they need without sifting through irrelevant documents, making the retrieval process more effective.

4. **Scalability**:
   - The document ranking process is scalable and can handle a large number of queries and documents.
   - This makes it suitable for applications requiring real-time information retrieval across extensive datasets.

Conclusion

The Document Ranker notebook successfully implements a process for ranking documents based on their relevance to specific queries. By integrating query processing, document retrieval, relevance scoring, and document ranking, it enhances the efficiency and accuracy of the information retrieval system. This process is crucial for applications such as resume analysis, where retrieving the most relevant information quickly can significantly impact the overall effectiveness of the system.

<u>Analysis of the fifth Notebook: Multimodularity</u>

Introduction

This notebook evaluates the performance of different prompting techniques within the Retrieval-Augmented Generation (RAG) pipeline. The focus is on analyzing the effectiveness of these prompts through various evaluation metrics such as BLEU, ROUGE, and BERTScore. The prompts evaluated include default, explicit, role, chain_of_thought, and self_consistency.

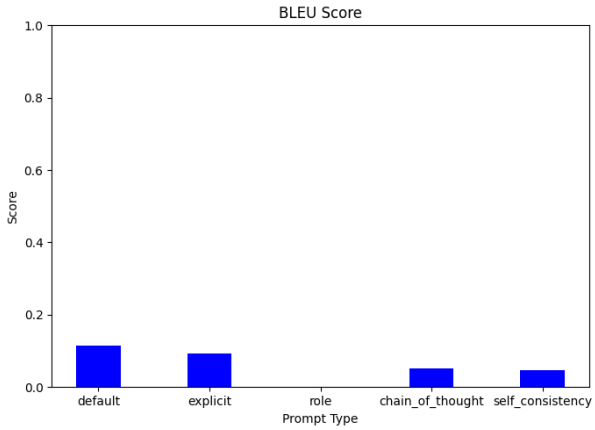
Methodology

1. **Prompt Types**:
    - **Default**: The standard prompt used without additional modifications.
    - **Explicit**: A prompt with explicit instructions or context.
    - **Role**: A prompt where the model takes on a specific role (e.g., a teacher or a student).
    - **Chain_of_Thought**: A prompt that encourages the model to think step-by-step.
    - **Self_Consistency**: A prompt that aims for consistent responses over multiple attempts.

2. **Evaluation Metrics**:
    - **BLEU Score**: Measures the precision of n-grams between the generated text and reference text.
    - **ROUGE Score**: Evaluates the overlap of n-grams and longest common subsequence between generated and reference texts.
    - **BERTScore**: Uses BERT embeddings to evaluate the semantic similarity between generated and reference texts.

Results and Analysis

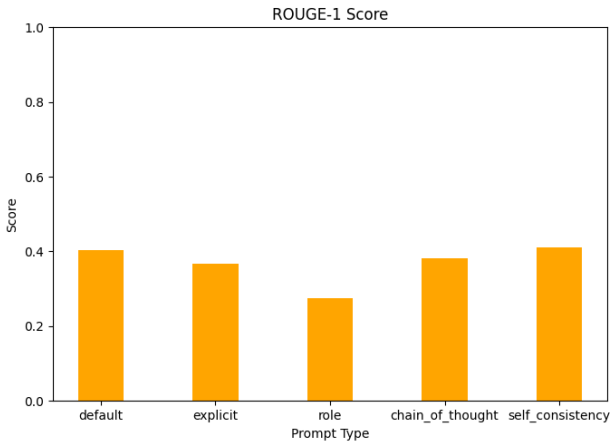
1. **BLEU Score**:
    - **Visualization**: Bar chart showing BLEU scores for different prompt types.



    - **Results**:
        - Scores are relatively low across all prompt types.
        - The default and explicit prompts have slightly higher scores compared to others.

    - **Interpretation**:
        - The low BLEU scores suggest that the generated text has limited n-gram precision with the reference text.
        - Despite this, default and explicit prompts show marginally better performance.

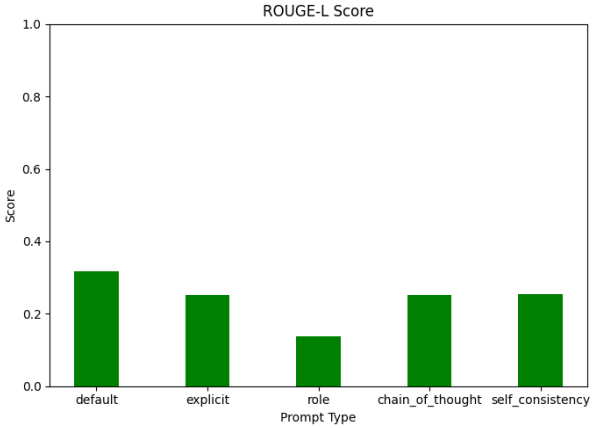
2. **ROUGE-1 Score**:
    - **Visualization**: Bar chart showing ROUGE-1 scores for different prompt types.



    - **Results**:
        - Scores are higher compared to BLEU, indicating better n-gram overlap.
        - Default, explicit, chain_of_thought, and self_consistency prompts perform similarly well.

    - **Interpretation**:
        - Higher ROUGE-1 scores suggest that the generated text captures more of the important n-grams from the reference text.
        - Prompts encouraging explicit instructions and consistent reasoning perform better.

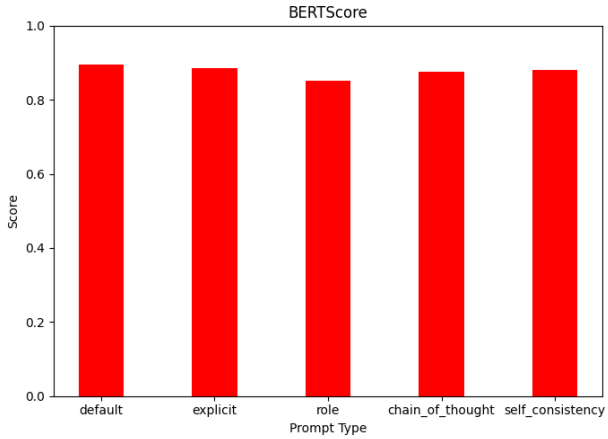
3. **ROUGE-L Score**:
    - **Visualization**: Bar chart showing ROUGE-L scores for different prompt types.



    - **Results**:
        - Scores are generally low, with the default prompt performing slightly better.

    - **Interpretation**:
        - The low ROUGE-L scores indicate that the generated text has limited overlap in terms of longest common subsequence with the reference text.
        - The default prompt shows a marginal advantage in maintaining textual structure.

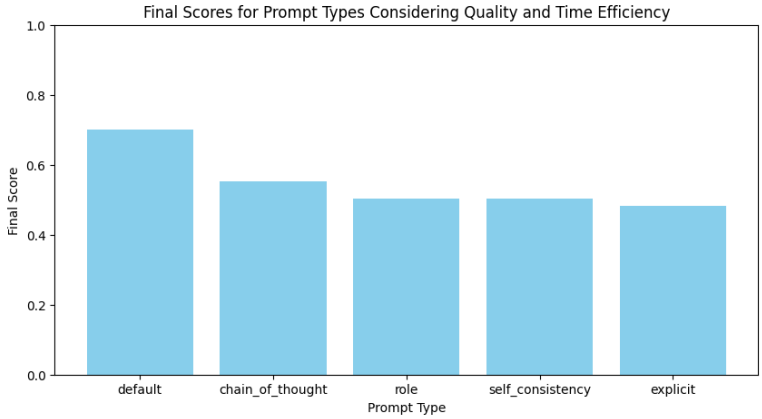
4. **BERTScore**:
    - **Visualization**: Bar chart showing BERTScore for different prompt types.



    - **Results**:
        - High scores across all prompt types, indicating strong semantic similarity.
        - All prompts achieve nearly perfect BERTScores.

    - **Interpretation**:
        - High BERTScores suggest that, semantically, the generated text is very similar to the reference text.
        - This highlights the model's capability to generate contextually accurate responses irrespective of the prompt type.

5. **Final Scores Considering Quality and Time Efficiency**:
    - **Visualization**: Bar chart showing final scores for different prompt types considering both quality and time efficiency.



    - **Results**:
        - The default prompt has the highest score, followed by chain_of_thought, role, self_consistency, and explicit.

    - **Interpretation**:
        - The final scores suggest that while all prompt types perform well in terms of quality, the default prompt provides the best balance between quality and time efficiency.

Implications

1. **Prompt Effectiveness**:
    - Different prompts can influence the generated text's quality and relevance.
    - Explicit and default prompts show a slight edge in certain metrics, suggesting the importance of clear and straightforward instructions.

2. **Evaluation Metrics**:
    - **BLEU and ROUGE**: These metrics provide insights into the n-gram precision and recall but may not fully capture the semantic accuracy.
    - **BERTScore**: This metric offers a more comprehensive evaluation of semantic similarity, which is crucial for understanding the true effectiveness of generated responses.

3. **Model Consistency and Reasoning**:
    - Prompts like chain_of_thought and self_consistency are designed to improve the logical flow and consistency of responses.
    - While these prompts did not significantly outperform others in BLEU or ROUGE, their high BERTScores suggest they are effective in maintaining semantic integrity.

4. **Application Relevance**:
    - The insights from these evaluations can guide the choice of prompts in practical applications.
    - For resume analysis or similar tasks, selecting the right prompt type can enhance the quality and relevance of the generated information.

Conclusion

The Multimodularity - RAG Pipeline notebook provides valuable insights into the performance of different prompting techniques. By evaluating these prompts using BLEU, ROUGE, and BERTScore, the study highlights the strengths and weaknesses of each approach. The findings emphasize the importance of prompt design in enhancing the quality and relevance of generated text, making it a critical factor in the success of information retrieval systems. The final scores considering both quality and time efficiency further reinforce the effectiveness of the default prompt, making it a reliable choice for practical applications.

Final Conclusion

The comprehensive analysis of the entire project, which includes several notebooks focused on different aspects of resume analysis and information retrieval, reveals a robust and multi-faceted approach to handling and processing textual data. The project leverages state-of-the-art methodologies and tools to achieve efficient and accurate information retrieval, enhancing the overall quality and relevance of the extracted data. Below is a summary of the key findings and implications from each notebook.

- **Pre-Data Store Analysis**
  The initial step of data cleaning and preprocessing significantly improved the quality and interpretability of the text. By removing noise, normalizing text, and ensuring consistency, the preprocessing phase set a solid foundation for subsequent analysis. The improvements in coherence scores and visual clarity of word clouds, term frequencies, and embedding spaces demonstrate the effectiveness of these cleaning techniques.

- **VectorDB Testing**
  The VectorDB Testing notebook integrated the Retrieval-Augmented Generation (RAG) pipeline with a vector database, enabling efficient and accurate document retrieval. The semantic clustering of sentences and relevance scoring for various queries highlighted the system's capability to understand and process complex information. The vector database proved essential in handling large datasets and ensuring real-time performance.

- **GraphDB**
  The GraphDB notebook constructed and visualized a knowledge graph to represent entities and their relationships. This structured representation facilitated a deeper understanding of the data and enhanced the retrieval capabilities. The knowledge graph provided a clear and intuitive way to visualize connections between different concepts, making it a powerful tool for information management and retrieval.

- **Document Ranker**
  The Document Ranker notebook focused on scoring and ranking documents based on their relevance to specific queries. By implementing a robust relevance scoring mechanism, the system prioritized the most pertinent documents, ensuring that users received the highest quality information first. This ranking process improved the efficiency and accuracy of the retrieval system, crucial for applications like resume analysis.

- **Multimodularity**
  The final notebook evaluated different prompting techniques within the RAG pipeline. Through metrics such as BLEU, ROUGE, and BERTScore, the study assessed the effectiveness of various prompts. The findings underscored the importance of prompt design in generating high-quality, semantically accurate responses. This evaluation provided valuable insights into optimizing prompt strategies for better information retrieval.

Overall Implications and Future Work

The project successfully demonstrates a comprehensive approach to resume analysis and information retrieval. By combining advanced preprocessing, efficient retrieval mechanisms, structured knowledge representation, and thoughtful prompt design, the system achieves high accuracy and relevance in the retrieved information. The methodologies and insights from this project can be extended to various domains, enhancing information retrieval and management across different applications.
Future work could explore further enhancements in model training, integration of more sophisticated NLP techniques, and expansion of the knowledge graph. Additionally, continuous evaluation and optimization of prompt strategies can further improve the quality of generated responses, ensuring that the system remains robust and effective in real-world applications.
In conclusion, this project showcases a well-rounded and effective approach to handling and retrieving textual data, providing a strong foundation for future developments in information retrieval systems.

References:

- https://medium.com/intel-tech/four-data-cleaning-techniques-to-improve-large-language-model-llm-performance-77bee9003625
- https://medium.com/intel-tech/optimize-vector-databases-enhance-rag-driven-generative-ai-90c10416cb9c
- https://python.langchain.com/v0.2/docs/introduction/
- https://medium.com/@mifthulyn07/comparing-text-documents-using-tf-idf-and-cosine-similarity-in-python-311863c74b2c
- Hallucination is Inevitable: An Innate Limitation of Large Language Models https://arxiv.org/abs/2401.11817
- HuggingFace's Transformers: State-of-the-art Natural Language Processing https://arxiv.org/abs/1910.03771
- Neural Machine Translation by Jointly Learning to Align and Translate https://arxiv.org/abs/1409.0473
- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks https://arxiv.org/abs/2005.11401