# Enhancing Historical Understanding with Retrieval Augmented Generation

**Srianusha Nandula**
snandula@ucsd.edu

**Saachi Shenoy**
svshenoy@ucsd.edu

**Colin Jemmott**
cjemmott@ucsd.edu

**Abstract**

Although the amount of resources available on the internet seems to be growing daily, it's becoming increasingly challenging to navigate the sea of information. While one might expect the influx of articles to make it easier to answer questions, the information accessible is often written well after the historical events have concluded, reflecting modern perspectives and interpretations. When answering historical questions, people often tend to rely on a simple Google search, which leads to websites such as Wikipedia. These sites can be problematic due to the unreliability of sources and the tendency to project 21st-century viewpoints onto historical information. These tools provide users with a surface-level summary that lacks the historical nuance and contextualization that is needed for a thorough understanding of the matter at hand. This project aims to address this gap in relevant and credible information retrieval. This project accomplishes these goals through Retrieval Augmented Generation (RAG), which is a combination of a search algorithm and Large Language Models (LLMs) to answer user queries. Retrieval Augmented Generation avoids the limitations of traditional search functions, which can struggle with large amounts of data, causing inaccurate or oversimplified results. A user can ask a question about historical events, and our model will search, retrieve, and synthesize data, ultimately responding to the user's questions using a diverse array of historical news sources.

Website: https://saachishenoy.github.io/Capstone-Website/
Code:
https://github.com/s-nandula/anusha-saachi-DSC180-Quarter2-Project

# 1 Introduction

This project serves to increase access to accurate and historically relevant information by combining search methods and LLMs via Retrieval Augmented Generation. Ideally, this project bridges the gap in finding reliable and context-rich historical information, countering the challenges of modern information retrieval platforms. If a user were to ask, "How did the Great Depression affect the agricultural economy in the Midwest United States between 1928 and 1929?" they will receive an answer sourced from multiple primary and secondary references that are rigorously fact-checked. When compared to the experience of searching through numerous sources and potentially encountering unnecessary, redundant, and incorrect information, using this model can be a more efficient and dependable tool. Given these potential advantages, this model can be useful for educational research and professional settings in order to help users understand historical context. This model stands out when it comes to narrow, focused historical questions because our model is only informed by relevant articles that are selected based on keywords in the user's query.

Retrieval Augmented Generation is the general framework for using data outside of a large language model's training data to improve the accuracy and clarity of the LLM's output (AWS 2023). Large Language Models have been used in tangent with search previously; however, due to the recent popularization of LLMs and LLM research, RAG has emerged as a relatively new method of information retrieval. Research by Lewis et al. (2020) has shared that, generally, RAG has the capability to handle a diverse array of "knowledge-intensive tasks" (tasks that could not be performed by a human without external intervention) by combining retrieval models and LLMs to refine AI outputs. Our model is an extension of the current research by testing the abilities of RAG for another knowledge-intensive task, the task of synthesizing information to answer historical questions. This extension differs from previous applications of RAG for knowledge-intensive functions as we are able to gauge its performance specifically on historical newspaper data. Additionally, the research of Cuconasu et al. (2018), shares that an often overlooked critical component to the success of a RAG pipeline is the nature and organization of the retrieval system. Our project seeks to build upon this literature by developing a retrieval algorithm based on our knowledge of search and the characteristics of our dataset. We will test the efficacy of our search methodologies and tools and hope to gauge how they can impact the overall success of a RAG pipeline

This dataset is comprised of newspaper data from the early twentieth century, specifically between 1925 and 1929. We sourced a variety of publications with the intention of representing the range of American perspectives and voices during this time period. The publications we used are the *Chicago Tribune*, the *Los Angeles Times*, *The New York Times*, *The Wall Street Journal*, *The Washington Post*, and the *San Francisco Chronicle*. These publications range from historically mildly conservative to mildly liberal but are generally treated as neutral sources of information. They also represent a variety of regions in the United States, specifically the East, West, and Midwest, as regional attitudes toward significant events can differ. In order to optimize space and memory costs, we chose only to include articles, excluding items such as obituaries and advertisements. In addition to the text in the news articles, the datasets also provide us with the title of the article, the date it was written, and the publication. We know the datasets have been cleaned and refined by UCSD and ProQuest TDM, who acquired the data directly from each of these publications. This data was prepared for use in this project through a mutual effort between the UCSD Data Science Librarian and ProQuest TDM.

# 2 Methods

## 2.1 Data Preprocessing and Storage

In order to use the newspaper data through ProQuest TDM, we first created datasets containing the subset of newspapers in the range of years that we wanted to use for the tool. The data from these datasets automatically transfer to a folder in TDM Studio's virtual environment which is then able to easily import into our Jupyter notebook. When we created our datasets, we filtered by articles only to minimize the amount of documents as well as remove any unrelated content. Each dataset has a limit of 2,000,000 documents. The data was originally stored in XML format so we made functions to strip the HTML tags and retrieve the metadata from each file. Specifically, we parsed each file for the following features: title, date released, publisher name, and text.

We began our code by configuring our notebook to incorporate all the necessary libraries and packages for the task at hand. Some general purpose libraries such as BeautifulSoup, sqlite3, and os were imported to parse XML documents, handle database operations, and maneuver files. We decided to store our data in a SQLite database rather than a Pandas dataframe. Since SQLite is a lightweight database engine it does not take up as much storage as a Pandas dataframe. After connecting to sqlite3 and creating a new database, we made a new table with five fields: goid, title, date, publication, and text. Then, we defined a function to insert the data from our XML files into the new table.

In order to create the document embeddings and index for search, we first access the documents from the SQLite database. To do so, we call a set of functions that connect to the SQLite database, retrieve all document IDs, and then obtain document text based on their IDs. Retrieving document IDs before fetching their full text from the SQLite table is a strategic approach to support batch processing and dynamic content retrieval. The functions were implemented using parallelization to optimize the retrieval process based on the scale of the data.

## 2.2 Initial Methodology

In the initial version of our tool, we created the document encodings utilizing a Bidirectional Encoder Representations from Transformers, or BERT, model. We specifically imported the model “bert-base-uncased” as this model is pre-trained in the English language. The BERT model was used to encode each document, which is when each document is represented by a dense numerical vector that is based on the wording and context of the text. We also used batch processing to first tokenize the text for each document in the batch. For each batch, padding and truncation were applied to fit the model's input requirements, and then the processed batch was fed into the BERT model to encode the document and get the respective embedding. Our encoding function then took the embeddings from the model's last hidden state, averaged them, and combined these embeddings into a NumPy array. Now that the documents were encoded, our code worked to create a Facebook AI Similarity Search index, or FAISS index - designed for efficient similarity search of a large scale of vectors. The document vectors were partitioned and compressed using a quantizer and the IndexIVFPQ method, working to decrease processing and memory needs. It is important to note that creating the encoding and FAISS index is a process that occurs only once. After the encoding and index are created, they are saved, and the saved files are loaded in for all subsequent uses.

Our semantic search allows the user to input a query, and the query is encoded to find a vector representation. Then, the search utilizes the FAISS index to find the 5 nearest neighbors of the query's vector. The 5 nearest neighbors documents were then chunked into 500 character subsections. Of these 500 character subsections, another search is done to find the 8 most relevant subsections. The LLM can use these 8 relevant subsections, that are generated from the nearest neighbors, to answer the user's query.

For our LLM we decided to use OpenAI's GPT-3.5-turbo model to generate our final response. To use this model, we generated an API key and created a function that would output the final response to the user's query. This function takes in the query itself along with the text from the 7 most relevant documents. The model then uses this context to come up with a specific and informative answer to the user's question.

## 2.3 Improved Methodology

As our project progressed and we began to test our tool, we encountered difficulties regarding accuracy and scalability, causing us to pivot from our initial approach. The BERT model encodings brought up concerns regarding the efficiency and relevance among the top documents for each test query. Additionally, using SQLite databases to manage our embeddings and conduct searches was not feasible for the amount of data we had. To manage these concerns, we shifted to a new approach to improve the performance and scalability aspects of our tool.

The initial steps of preprocessing and storing the documents remained the same in our improved approach. However, to process the text and create embeddings, we utilized SentenceTransformer and SentenceTransformerEmbeddings rather than relying on BERT. SentenceTransformer models were more efficient for our data since we were dealing with articles that contained multiple passages. Since these models are designed to generate less dense embeddings, the processing times are faster than they would have been for BERT. BERT models focus on token embeddings while SentenceTransformer models provide embeddings with content of the entire text. This is advantageous for our search because our search algorithm needs to extract the most relevant documents in order to answer the user's query. After experimenting with our model, we chose to chunk our data into segments of 500 characters with an overlap of 50 characters for relevant embedding generation. The SentenceTransformer model and tokenizer were used to encode the documents and generate vectors based on the textual meaning.

Instead of storing these embeddings in a SQLite database, we made the switch to use a Chroma database instead. Due to our large volumes of data, using a Chroma database was a better suited approach since it is superior in both performance and scalability when compared to an SQLite database. Chroma optimizes storage space and ensures high-speed access to the embeddings when they need to be retrieved. Once the user's query is inputted, Chroma performs semantic similarity search by efficiently comparing query embeddings with article embeddings stored in the database. After this process, the four most relevant chunks are retrieved in order to answer the user's query. We tested our model with different numbers of retrieved chunks to see what would facilitate the most relevant results. We noticed that if we had more than four chunks it would create noise and start to return less relevant information in the LLM output.

For our LLM, we decided to use the multi-qa-MiniLM-L6-cos-v version of the LLamas LLM instead of OpenAI's GPT-3.5-turbo model to generate our final response. Although we had a variety of options to choose from, Llamas fit the goals of our project best. Our mission was to answer complex historical questions and LLamas excels at generating informative responses based on given context. This function takes in the query itself along with the text from the four most relevant documents. The model then uses this context to come up with a specific and informative answer to the user's question. By leveraging LLamas we are able to deliver more accurate answers to user queries and improve the overall satisfaction with our question-answer system. This transition aligns more closely with our project goals and gives us the backbone to optimize our tool.

# 3 Results

This section shares the outputs of our Retrieval-Augmented Generation pipeline and an evaluation of our tool's performance. This investigation not only had the objective of improving the refinement of LLM outputs but also evaluating the performance and efficiency of our search/retrieval information methodology. Through a systematic comparison of embedding techniques, an in-depth analysis of the initial method's performance, and a detailed examination of user response satisfaction, we offer clear insights into the model's capabilities, justifications for changes in methodology, and areas for improvement.

The first step in creating the search function is creating the vector embeddings for each document that the search operates upon. These vector embeddings are generated by a model, and each model contains its own architectural complexities that affect both embedding accuracy and computational efficiency. The models we compared for computational efficiency and embedding dimensionality were DistilBERT, bert-base-uncased, TinyBERT and SentenceTransformer's all-MiniLM-L6-v2. We conducted a comparative analysis, shown in Table 1, that examines each model's architectural complexity by quantifying their training parameters, evaluates their encoding efficiency through the encoding time of a dataset of 100,000 documents, and assesses their ability to handle long text encoding through the maximum token length limit.

Table 1: Model Comparison of Architectural Complexity and Encoding Efficiency

| Model Name | Number of Training Parameters | Maximum Input Length (in Tokens) | Time to Encode 100,000 Documents (seconds) |
|---|---|---|---|
| bert-base-uncased | 110,000,000 | 512 | 336 |
| DistilBERT | 66,000,000 | 512 | 224 |
| TinyBERT | 14,500,000 | 512 | 84 |
| all-MiniLM-L6-v2 | 22,000,000 | 256 | 100 |

Table 1 compares four language models, showing that TinyBERT is the fastest at encoding, while all-MiniLM-L6-v2 offers a balance between speed and a higher parameter count. DistilBERT also demonstrates increased speed over bert-base-uncased, which has the highest number of parameters and longest encoding time.

The evaluation of our initial search methodology, which used BERT embeddings, a FAISS index, and a quantized search, resulted in significant issues. The search function produced nearly identical output document indices for a diverse array of test queries. This unchanging output indicated that the algorithm was not effectively differentiating between the meanings of different queries and possibly even different documents. As such, the search could not retrieve documents that were contextually relevant to the specific historical context of each query. Despite this issue, we chose to complete the RAG pipeline for model evaluation purposes. Given that the LLM relies on relevant context to answer user queries, the lack of relevant documents had a substantial impact on the quality of response. Due to the poor search methodology, the LLM received irrelevant context documents, and this caused the LLM to consistently generate the same response: "I cannot answer the question as no relevant context was provided." This outcome was consistent across 20 diverse queries. These results were not suitable for formal evaluation; however, there was a clear pattern of obvious failure and objectively poor performance.

To test overall tool performance, we conducted multiple rounds of user satisfaction testing. In the initial round of user satisfaction assessment, five distinct queries, labeled A through E, were distributed among seven users, who were instructed to input each query into the tool and rate their satisfaction with the generated response on a scale from 0 to 5. Users were also encouraged to provide comments and feedback on the quality responses generated by our tool. The common feedback indicated that while the outputs were somewhat relevant, they lacked general depth and details. After the first round of satisfaction testing, we implemented improvements to the LLM's prompting and reduced the number of relevant chunks presented as context to the model. The original prompt, "Use this as context to answer the question" was changed to "Please provide a detailed answer that is based solely on the context provided," with the goal of prompting the model to produce more specific and detailed answers. Reducing the number of chunks from seven to four, aimed to reduce noise and complication within the context provided to the LLM. For the second round of user satisfaction testing, the mean user satisfaction scores for each of the five queries increased. The results, shown in Figure 1, plot mean user satisfaction scores before and after the improvements were made across all queries. Notably, queries A and E showed the most substantial increase in satisfaction, with scores rising from 3.42 to 4.14 and 4 to 4.42. Query D was the only instance where the satisfaction score slightly decreased from 3.71 to 3.57 in round two.

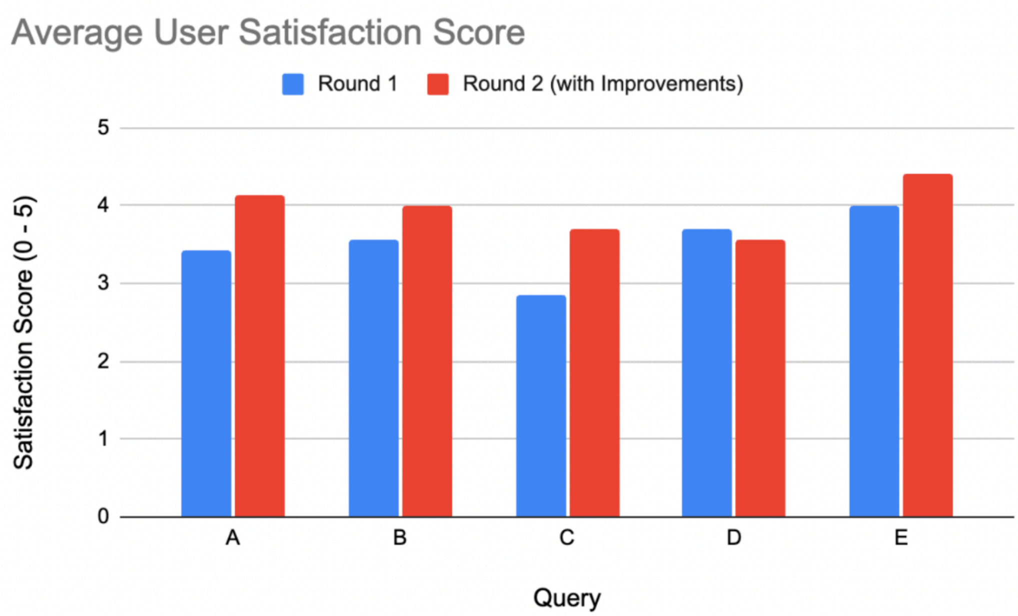


Figure 1: User Satisfaction Scores Before and After LLM Prompting Improvements

To demonstrate the practical results of our tool, we present a sample query along with the corresponding responses generated by the LLM in Figure 2. Readers can view more results from the tool in the Appendix A.2 , which contains a set of queries and the corresponding tool-generated outputs.

LLM

Hello! What historical question can I help you with today?

You

What were the impacts of the Great Mississippi Flood of 1927?

LLM

Based on the context provided, the impacts of the Great Mississippi Flood of 1927 were: 1. Devastating damage: The flood caused an estimated $10 million in damage, which is equivalent to over $100 million in today's money. 2. Loss of life: Thousands of people were killed or injured during the flood. 3. Displacement: Many people were forced to leave their homes and move to other areas for safety. 4. Economic impact: The flood had a significant impact on the economy of the region, including the suspension of purchasing power in affected communities and districts. 5. Long-term benefits: While the flood was devastating at the time, it led to steps being taken to prevent such events from happening again in the future, including flood relief measures that were implemented and are still in place today. 6. Increased agricultural production: The flood resulted in a greater agricultural district in the Mississippi Valley than had ever been known before.

Figure 2: Response to a Sample Historical Query

# 4 Discussion

This project successfully explored the potential to utilize RAG to address historical understanding between 1925 and 1929. Using user satisfaction and the quality of output as metrics of success, this discussion analyzes the details of our results, and how they connect to the efficacy of search technologies, current research, and the general implications of our project.

As a brief summary of our methodology change, we shifted from using the bert-base-uncased embedding model in combination with an FAISS index for our search function. These changes were made due to the inefficiency and inaccuracy of the embedding models and created embeddings. Due to this poor performance, we did a series of efficiency tests on multiple embedding models to find a model that was efficient enough to scale up to 2 million documents while being complex enough to maintain the meaning of a document. The architectural complexity of a model, which we gauged via the number of training parameters, directly affects the ability of a model to capture the nuances and meaning of language better. However, due to the density of these models, they are often accompanied by longer encoding times, meaning the ideal embedding model would serve as a balance between the two. This was a particular concern when we factor in the scaling of the embedding to a corpus of 2 million documents. We found that TinyBERT and all-MiniLM-L6-v2, while less complex, were optimized for efficiency and were, therefore, ideal candidates for embedding models. However, we ultimately chose to use SentenceTransformer's all-MiniLM-L6-v2, despite its slightly slower performance over TinyBERT, due to the increase in complexity and training parameters. Additionally, although all-MiniLM-L6-v2's limit is 256 tokens, this limitation was circumvented by the addition of a chunking methodology. We found that the increased quality of embeddings and increased speed of embedding offset the cost of increasing the number of documents from 2 million documents to 10 million chunks of documents. This change increased the relevancy of each chunk retrieved in the search, likely due to the exclusion of noise that often comes with long text encoding.

When evaluating our initial search, we saw that the search function was unable to discern between the meanings of documents and the meanings of the user's query, making any document retrieval functionally random. This heavily impacted the quality of the LLM output as it could not complete the task at hand of responding to the user's query. In comparison, the new methodology allowed the LLM to produce nuanced, detailed, and thorough answers to the user's queries. Due to this change, in a comparison of the two methodologies and their outputs, one of the methodologies is very clearly more appropriate for the goals of this project. The subsequent rounds of user satisfaction testing provided us with valuable suggestions to improve the tool and allowed us to gauge the tool's performance from the user's perspective. The initial reception to the tool showed that users believed queries were relatively surface-level and were generally relevant but occasionally included seemingly irrelevant details. This feedback guided the tuning of the model and further improvements to reduce noise and confusion for the LLM. The tuning had a tangible impact on model performance, as there was a noticeable increase in user satisfaction after these adjustments. After the tuning was complete, we saw that users were holistically, very satisfied with the output of the tool, and based on the average satisfaction score exceeding a 4/5, we are choosing to evaluate our effort as successful. When querying the tool for answers to a diverse array of questions, the model consistently performed well, providing users with a thoughtful, complex, and digestible answer. In terms of the literature of the field, we find that our project lines up with current findings. Overall, the model highlighted the ability of RAG to help humans complete knowledge-intensive tasks and showcased a new domain of RAG application that could possibly be explored by other researchers. Additionally, our change in methodology, and dramatic change in model performance after refining our search function, echoed the findings of current literature that preaches the importance of the information retrieval algorithm in a RAG pipeline.

While our project represents advancements in the field of historical information retrieval, we want to address certain limitations of our tool that users should keep in mind. Despite efforts to select neutral sources, the newspaper articles included in our dataset may still exhibit inherent biases or inaccuracies. Furthermore, the reliability of historical information retrieved through RAG is contingent upon the quality and credibility of the underlying sources, which is an assumption we make that is not guaranteed. Moreover, the historical information retrieved may be influenced by the language prevalent during the time period covered by the dataset. This could mean that the terminology that users input as their query may not match the phrases that were used in the 1920s. Additionally, the LLM will not have cultural context prior to 1925 due to the range of data that we used. This could pose challenges for users seeking information on topics outside of the dataset's linguistic or cultural scope.

Looking ahead, there is scope for further refinement and optimization of our tool. Future iterations could focus on improving the efficiency of search algorithms, enhancing document encoding techniques, and exploring additional sources of historical data. The success of our project has important implications for educational research and professional settings, where access to reliable historical information is paramount. Our methodology can serve as a valuable tool for students, academics, and researchers seeking to deepen their understanding of historical events and contexts. The iterative improvement process demonstrates that our tool will have to be evaluated continuously to keep the information retrieval system up-to-date and efficient. Although this tool was specifically designed for historical text data, LLMs and RAG can be incorporated into various domains in the near future - allowing this technology to impact multiple fields of study. Overall, our project is a step forward in the field of information retrieval and can act as a foundation for future innovation in this area.

# References


**AWS.** 2023. "What is Retrieval-Augmented Generation?." [Link]

**Cuconasu, Florin, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri.** 2018. "The Power of Noise: Redefining Retrieval for RAG Systems." [Link]

**Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela.** 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." In *Advances in Neural Information Processing Systems*. [Link]

# Appendices



## A.1 User Satisfaction Testing

Table A 1: Round 1 of User Satisfaction Testing

| | Round 1: Prompt ("Use this as context to answer the question") and 7 Chunks | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Participant 1 Score | Participant 2 Score | Participant 3 Score | Participant 4 Score | Participant 5 Score | Participant 6 Score | Participant 7 Score | Mean Score |
| Query A | 3 | 4 | 4 | 3 | 3 | 3 | 4 | 3.42 |
| Query B | 4 | 4 | 5 | 3 | 3 | 4 | 2 | 3.57 |
| Query C | 2 | 3 | 4 | 4 | 1 | 3 | 3 | 2.86 |
| Query D | 5 | 4 | 3 | 4 | 3 | 3 | 4 | 3.71 |
| Query E | 4 | 4 | 5 | 3 | 5 | 4 | 3 | 4.00 |

Table A 2: Round 2 of User Satisfaction Testing (with improvements)

| | Participant Scores | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Participant 1 Score | Participant 2 Score | Participant 3 Score | Participant 4 Score | Participant 5 Score | Participant 6 Score | Participant 7 Score | Mean Score |
| Query A | 5 | 4 | 5 | 3 | 4 | 5 | 3 | 4.14 |
| Query B | 4 | 5 | 3 | 4 | 4 | 4 | 4 | 4.00 |
| Query C | 4 | 3 | 4 | 3 | 5 | 4 | 3 | 3.71 |
| Query D | 3 | 4 | 4 | 4 | 4 | 3 | 3 | 3.57 |
| Query E | 5 | 5 | 4 | 3 | 5 | 5 | 4 | 4.42 |

Table A 3: Queries Used for Testing

| | Questions |
|---|---|
| A | What were the main causes and effects of the Stock Market Crash of 1929 on the American economy? |
| B | What were the key events and outcomes of the 1928 Presidential Election in the United States? |
| C | Describe the political and social factors that led to the Kellogg-Briand Pact of 1928. |
| D | How did the introduction of talking movies in 1927 change the film industry? |
| E | How did advancements in aviation between 1925 and 1929 influence global perceptions of travel and trade? |

## A.2 Queries and Responses

Table A 4: Set of queries and responses to view tool functionality

| **Query** | **Response** |
|---|---|
| Discuss the significance of the First Five-Year Plan introduced by Joseph Stalin in 1928 in the Soviet Union. | The Five-Year Plan introduced by Joseph Stalin in 1928 was significant in the Soviet Union for several reasons: Improved Economic Performance, Modernization of Agriculture, Reduction of Inequality, Increased Control Over the Economy, and International Significance. The plan aimed to rapidly industrialize the country, modernize agriculture, reduce economic inequality, increase government control over the economy, and demonstrate the Soviet Union's commitment to socialism. This helped to establish the Soviet Union as a major world power. |
| Describe the Scopes "Monkey" Trial of 1925. Who were the key figures involved, and what was the trial's significance? | The Scopes "Monkey" Trial of 1925 was a highly publicized legal case in Dayton, Tennessee, over the teaching of evolution. Key figures included John Scopes, William Jennings Bryan, Clarence Darrow, and George Rappleye. The trial highlighted cultural and political tensions around the teaching of evolution, arguing for the importance of scientific inquiry and the separation of church and state. Despite Scopes' conviction (later overturned), the trial had a lasting impact on the debate over evolution in public schools. |

Table A 5: Set of queries and responses to view tool functionality (cont.)

| **Query** | **Response** |
|---|---|
| What was the purpose of the Locarno Treaties signed in October 1925, and which countries were involved? | Based on the context provided, the purpose of the Locarno Treaties signed in October 1925 was to establish a system of collective security in Europe, with the aim of preventing future wars. The treaties were signed by several countries, including Germany, France, Belgium, the Netherlands, and Italy. The treaties established a system of arbitration and disarmament, whereby any disputes between the signatory countries would be settled through arbitration rather than through force. Additionally, the treaties provided for the recognition of existing frontiers and the renunciation of the use of force in the settlement of conflicts. |
| What was the Kellogg-Briand Pact of 1928, and why did it ultimately fail to prevent future conflicts? | The Kellogg-Briand Pact was a treaty signed in 1928 aimed at outlawing war and promoting peace. It was the result of a long-standing movement towards consolidation of peace, following the failure of the League of Nations. The pact was significant because it was the first international agreement to outlaw war and provide for its settlement by pacific means. The treaty was signed by 62 countries, including the United States, and was seen as a major step towards achieving lasting peace. However, the pact ultimately failed to prevent future conflicts due to several reasons. Firstly, the pact was not ratified by the United States Senate, which limited its effectiveness. Secondly, the pact did not provide for any enforcement mechanism, making it difficult to hold countries accountable for their actions. Finally, the pact did not address the root causes of war, such as economic and political tensions. |