Process Book

Leveraging AI and Vector Databases for User-Driven Document Analysis and Interactive Chart Transformations

CS573 - Data Visualization Group Project - Spring 2024

Andrew Kerekon

Aviv Nur

Bijesh Shrestha

Contents

List of Figures

Abstract

Efficient extraction and interpretation of information from documents containing diverse chart types are complex tasks that significantly depend on the reader's prior knowledge and proficiency. Recognizing the challenges inherent in chart question and answer (CQA) systems, our project aims to advance this field significantly. Motivated by ongoing research and building upon existing studies that explored the integration of Large Language Models (LLMs) and data synthesis techniques, we implement methodologies for interacting with and extracting data from various chart formats. Our project addresses the deficiencies in current systems by implementing a robust framework that combines a vector store database with advanced embedding techniques for semantic search, enhancing the reliability and accuracy of the retrieved data. This integration improves system performance and ensures scalability, crucial for handling diverse and voluminous chart data. Our contributions refine the interaction between users and document-based chart data through sophisticated transformations and user-assistant interfaces, enabling more precise and insightful analyses. This approach not only bridges the gaps identified in the literature but also sets a new standard for reliability and functionality in CQA systems.

1 Background and Motivation

1.1 Background

Extracting and interpreting information from documents containing various charts often demands considerable time and relies significantly on a reader's prior knowledge and background. The state-of-the-art paper by [1] highlights the critical need for chart questions and answer (CQA) systems capable of synthesizing text and visuals for comprehensive insights, and provides problem space as shown in Figure 1.

Various research is being done that attempts to fill this gap through the integration of LLM and techniques to synthesize texts and/or chart data that enable users to interact at various levels and depths. Recent papers include methodologies for extracting chart data from diverse sources such as embedded SVGs in PDFs, screenshots, and user uploaded images [2]–[4]. Chen et al. [5] worked on application design that integrates various APIs to ensure the right fine-tuned LLM is used for proper interpretation and insights drawn from those user-uploaded chart data However, we see some gaps in terms of the reliability of information that mostly comes from the LLM's general knowledge.

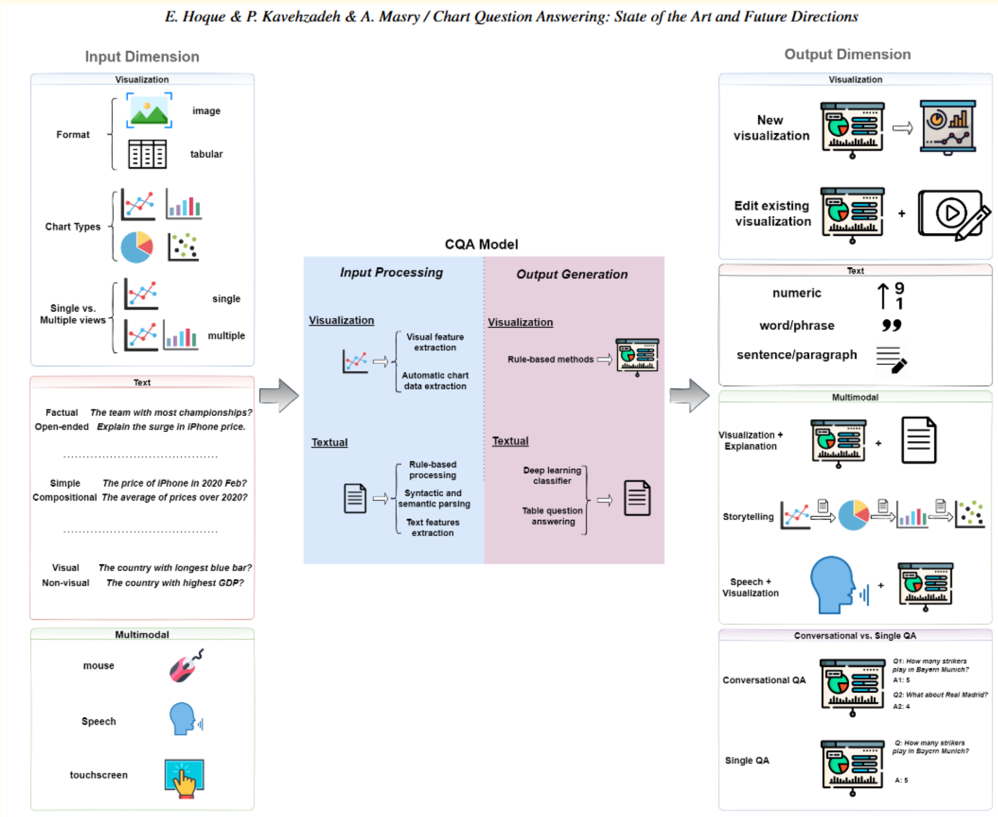


Figure 1: State of the Art and Future Directions of Chart Question Answering.

1.2 Motivation

The importance of efficiently extracting and interpreting information from documents featuring diverse chart types cannot be overstated, given the complex nature of these tasks and their dependency on a reader's prior knowledge and proficiency. Recognizing the challenges inherent in chart question and answer (CQA) systems, which must effectively synthesize textual and visual data to yield comprehensive insights, our project seeks to significantly advance this field. Motivated by the ongoing research in this area and building upon existing studies that have explored the integration of Large Language Models (LLMs) and data synthesis techniques, our project introduces innovative methodologies for interacting with and extracting data from various chart formats, including SVGs embedded in PDFs and images uploaded by users. Despite the progress detailed in recent literature, current systems often fall short in terms of reliability, especially when relying on the generalized knowledge provided by LLMs.

Our project addresses these deficiencies by implementing a robust framework that combines a vector store database with advanced embedding techniques for semantic search, thereby enhancing the reliability and accuracy of the data retrieved. This integration not only improves the system's performance but also ensures that it can operate at scale, a crucial factor given the diverse and voluminous nature of chart data. In summary, our contributions are aimed at refining the interaction between users and document-based chart data through sophisticated transformations and user-agent interfaces, enabling more precise and insightful analyses. This approach not only bridges the current gaps identified in the literature but also sets a new standard for reliability and functionality in CQA systems.

2 Related Works

Hoque et al. [1] report on open research questions concerning the integration of chart visualizations with textual explanations to enhance user query responses. Chen et al. [5] showcase the LLaVA-interactive, which leverages existing AI technologies for a multi-modal interaction experience involving visual chat, image segmentation, and image generation, without the need for additional model training. Han et al. [4] focus on the generation of datasets and visualizations using GPT-4 through the ChartLlama platform, a specialized version of the LLaVA-1.5 model, to facilitate data analysis and Q&A.

Masson et al. [6] introduce Charagraphs, dynamic interactive charts with annotations integrated into text, aimed at improving data interpretation and usable in physical documents through OCR and augmented reality. Ko et al. [7] propose the VL2NL framework to develop natural language datasets for data visualization, thereby advancing NLI development by synthesizing chart semantics. Peng et al. [2] explore the application of domain-specific APIs for chart data interpretation within context, enabling interactive analysis and modifications through natural language. Lastly, Wang et al. [3] develop a language model, FinViz-GPT, tailored for financial chart analysis that excels in chart description, finance-related Q&A, and market trend prediction.

From the state-of-the-art paper of CQA [1], we also try to refine what was done in terms of existing works and proposed future works as follows:

Existing Work

- **Textual Dimension:**
  - Focus on factual questions with specific answers using heuristic methods and transformer architectures.

- Exploration of questions based on visual elements vs. those without visual references.
- Examination of single-task vs. multi-operation questions, highlighting challenges in compositional questions.

- **Visualization Dimension:**
  - Emphasis on common chart types and scenarios involving single visualizations.
  - Utilization of computer vision and OCR for feature extraction, with difficulties in handling dynamic encodings and complex queries.
- **Multimodal Dimension:**
  - Integration of natural language with modalities such as touch and pen, but limited exploration beyond these interactions.
- **Output Dimension:**
  - Focus on textual outputs using fixed or open vocabularies, with emphasis on questions requiring deeper analytical reasoning.

Future Directions

- **Textual Dimension:**
  - Development of models that can effectively handle complex, compositional questions.
  - Enhancement of data extraction accuracy from vector images of charts.
- **Visualization Dimension:**
  - Improvement in managing multiple views and refining the interaction between chart elements and question-answering processes.
- **Multimodal Dimension:**
  - Expansion of research into integrating various modalities to improve user interactions with CQA systems across different platforms.
- **Output Dimension:**
  - Advancement of systems supporting open-ended queries and integrating visual data with explanatory texts for richer answers.

3 Questions

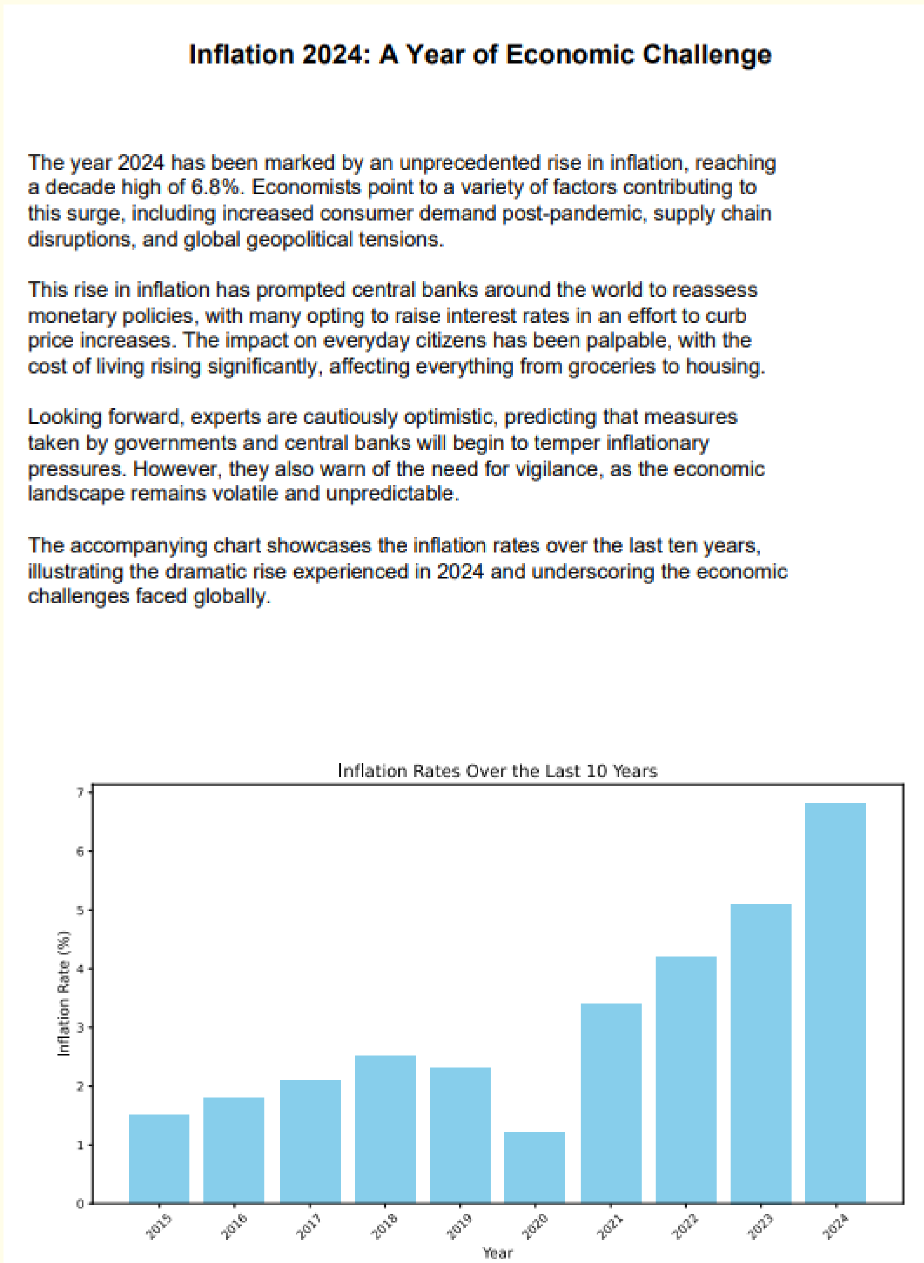
Through this query tool, we hope to answer the following questions:

1. How can we utilize an LLM to extract information from a document containing embedded charts?
2. Which insights are most important to a user when extracting information from a document? Does this depend on the use-case of that particular user?
3. How can we ensure an LLM has context of past conversations while also answering questions specific to the most recent document correctly?
4. How can we most accurately recreate charts based on user queries and data?

As the project evolved, we considered new questions relating to hallucinations (can we change system prompts or GPT model to minimize hallucinations?) as well as how we can ensure data is in an optimal format to be read by our LLM.

4 Data

In the process of developing our tool, we created two static PDF documents pre-uploaded as context for our vector databases. The first document contains a GPT-generated set of paragraphs about inflation in 2024 alongside a bar chart graphic representing the past 10 years of inflation as follows:



Inflation 2024: A Year of Economic Challenge

The year 2024 has been marked by an unprecedented rise in inflation, reaching a decade high of 6.8%. Economists point to a variety of factors contributing to this surge, including increased consumer demand post-pandemic, supply chain disruptions, and global geopolitical tensions.

This rise in inflation has prompted central banks around the world to reassess monetary policies, with many opting to raise interest rates in an effort to curb price increases. The impact on everyday citizens has been palpable, with the cost of living rising significantly, affecting everything from groceries to housing.

Looking forward, experts are cautiously optimistic, predicting that measures taken by governments and central banks will begin to temper inflationary pressures. However, they also warn of the need for vigilance, as the economic landscape remains volatile and unpredictable.

The accompanying chart showcases the inflation rates over the last ten years, illustrating the dramatic rise experienced in 2024 and underscoring the economic challenges faced globally.

Figure 2: Inflation Paragraph and Chart.

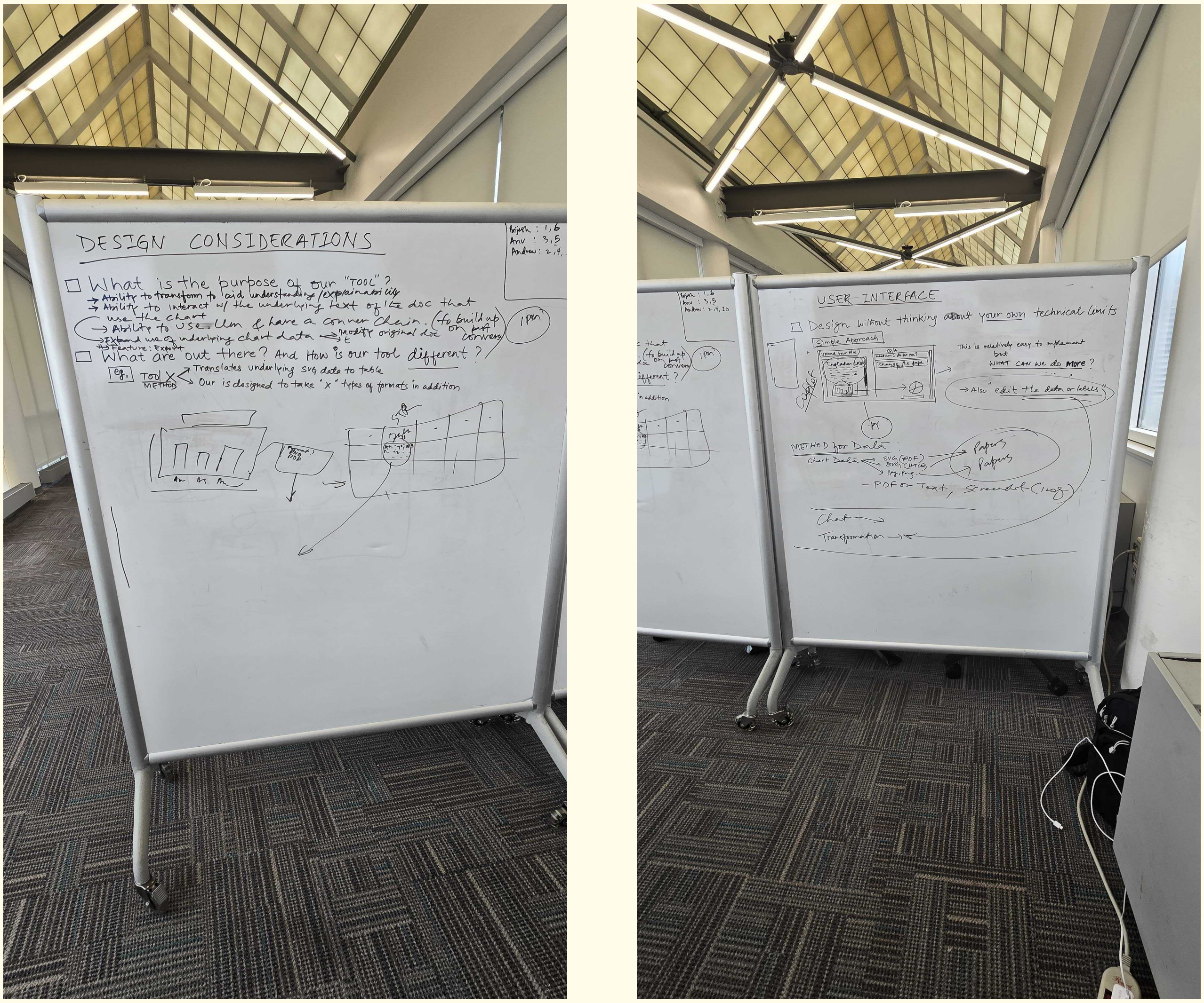
To ensure the LLM had the proper context for reading this visualization, and to overcome limitations of reading directly from a graphic, we also created a second document describing the embedded chart:

- Title of the chart is "Inflation Rates Over the Last 10 Years".
- Chart type is a bar chart.
- Chart created using matplotlib.
- X-axis is labeled "Year".
- Y-axis is labeled "Inflation Rate (%)".
- Data points are colored in skyblue.
- The chart measures inflation rates from 2015 to 2024.
- Year 2015 had 1.5 percent inflation.
- Year 2016 had 1.8 percent inflation.
- Year 2017 had 2.1 percent inflation.
- Year 2018 had 2.5 percent inflation.
- Year 2019 had 2.3 percent inflation.
- Year 2020 had 1.2 percent inflation.
- Year 2021 had 3.4 percent inflation.
- Year 2022 had 4.2 percent inflation.
- Year 2023 had 5.1 percent inflation.
- Year 2024 had 6.8 percent inflation.
- X-axis ticks are rotated 45 degrees for better readability.
- The layout is adjusted for tight fitting of the chart elements.

Figure 3: Description of Embedded Chart.

5 Design Evolution

We started our design by make it simple and minimalistic. As you can see on Figure 4, we split the main area into two part, the first for conversation part and the second is for PDF preview. By design like this, we expect user can side by side looking at PDF uploaded while they asked the questions.



(a) Design Consideration.

(b) Our First UI Mockup.

Figure 4: Design Brainstorming

In our final design, since we use streamlit as a web framework, we need to adjust our design to accommodate streamlit limitations. For the conversation

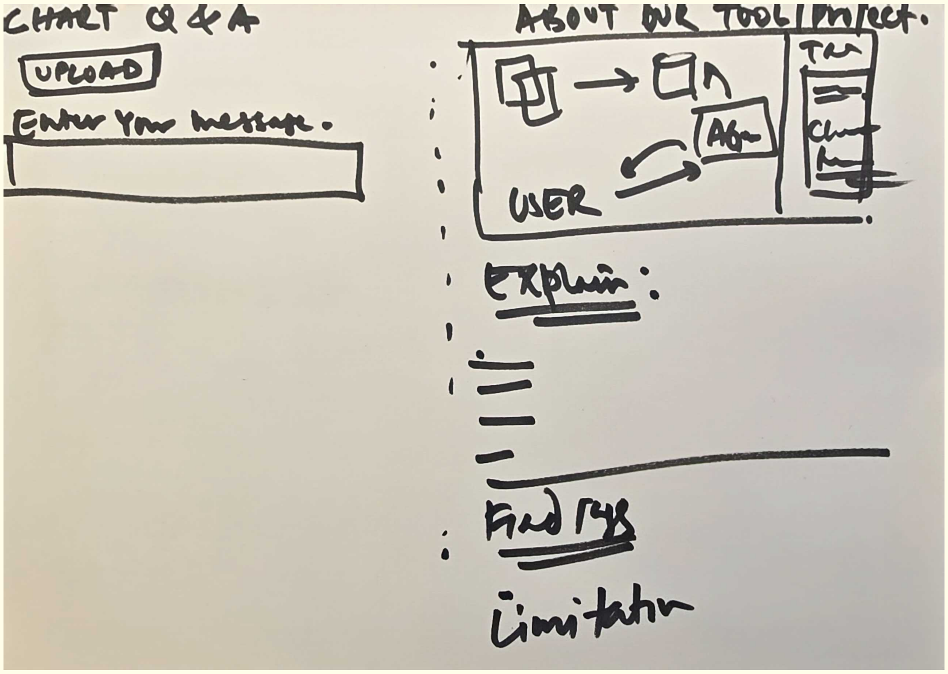


Figure 5: Our First UI Prototype

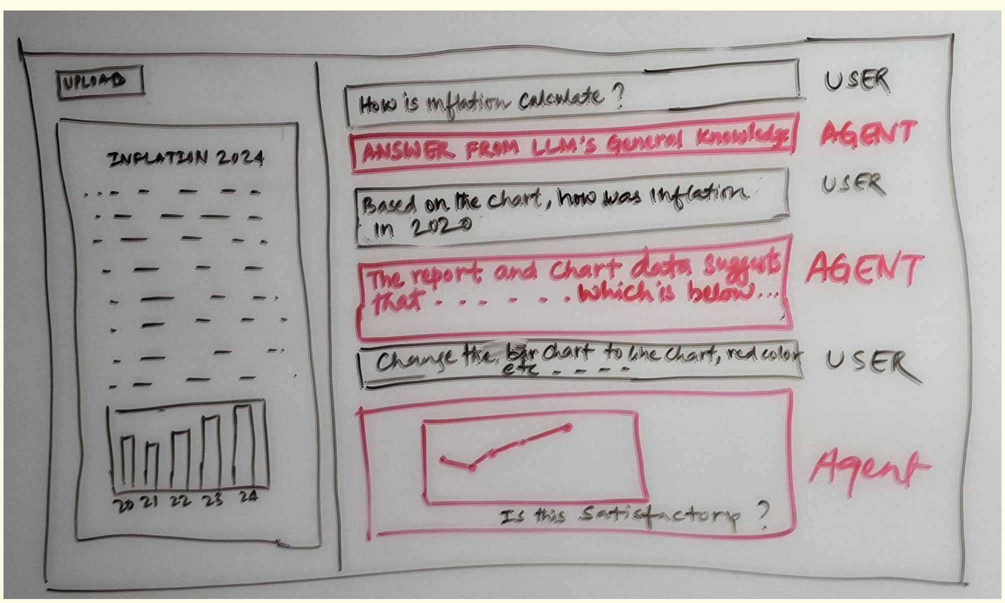


Figure 6: First UI Design.

part, we prefer to use send button because we can fully set it to a specific function instead of using enter button. We also switch sides between PDF preview and conversation section since it is easier for us to chat when we put it on the left hand side after we tested it several times.

We add an expander for the PDF preview as in the Figure 7, so user can hide/show the PDF, so as well for the framework and finding-limitation information. We also provide a short instruction to make it clear.

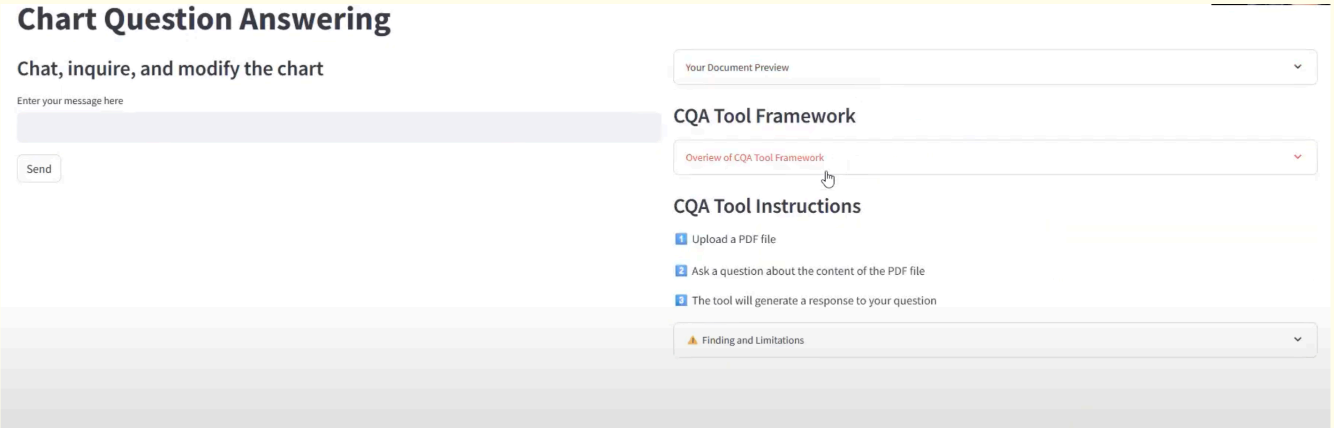


Figure 7: Final UI Design.

For future works, we consider to accommodate an upload button on the sidebar (Figure 8), so user can actually upload the document directly.

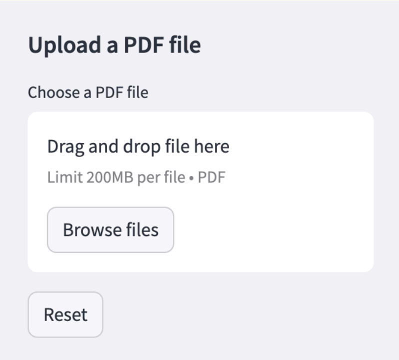


Figure 8: Upload Button on the Sidebar.

6 Implementation

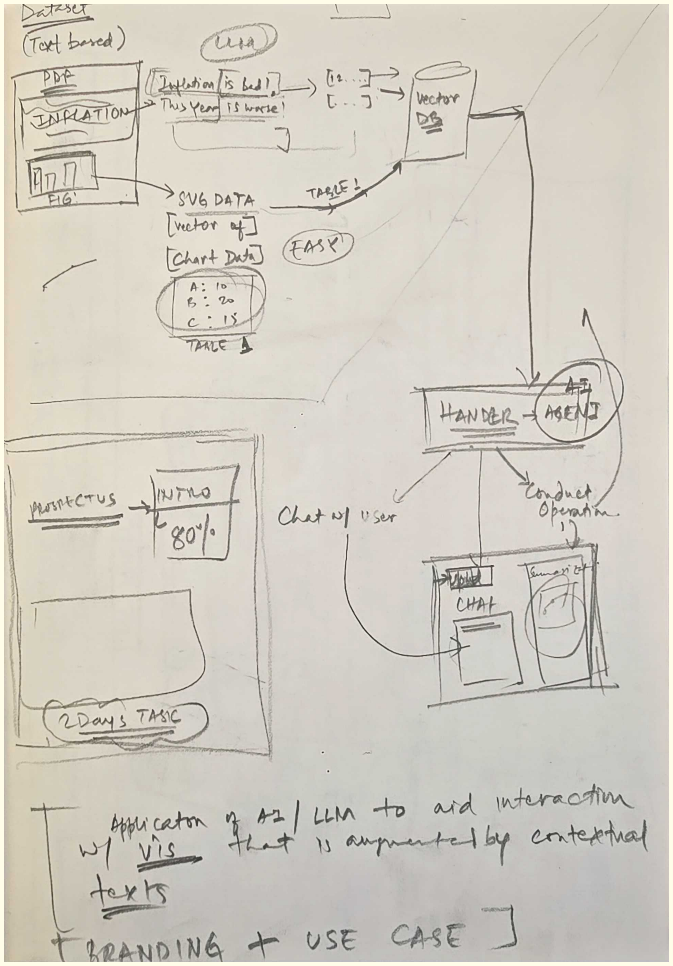


Figure 9: Pipeline Brainstorming

To accomplish developing a pipeline to answer user questions about charts, it was necessary for us to develop a full-stack application, with a Streamlit frontend passing user queries to a Python backend that utilizes an LLM.

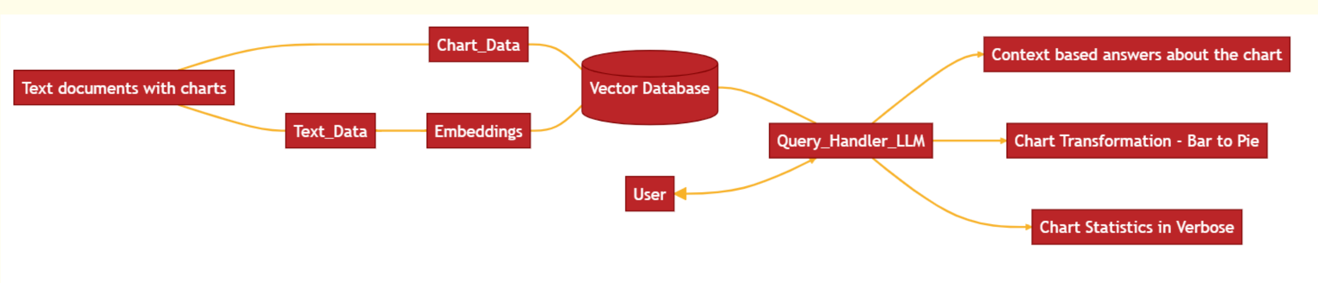


Figure 10: High-level Overview of Pipeline.

6.1 Backend Development

Our first step in developing a pipeline to answer questions about uploaded charts was to develop the necessary backend and vector databases to handle uploaded chart data.

Figure 11: Techstack Used in this Project.

When starting our program, the backend first attempts to load stored "text data" and "chart data" from a persistent vector database using the LlamaIndex [8] Python library. This data is used to load an index of contextual data based on previous input, which is then passed into separate "text engine" and "chart engine" wrappers that a GPT model can call as tools to interpret a user's query. If no data exists in storage, we test our extraction logic by uploading a static PDF containing a paragraph about inflation and a static PDF describing the features of a chart about this inflation as described in Section 6.1.1. We first chunk all extracted text, then use these chunks to create an embedding using the default text-embedding-ada-002 technique. From these embeddings, we form indices stored in a local vector database so our model has continuous context of the contents of a document while only needing to parse it a single time. These embeddings are shown in Figure 12 below:

{"embedding_dict": {"fb59630a-8dd8-4e15-b981-1b758404c7df": [-0.028477787971496582, -8.738884935155511e-05, -0.023047015070915222, 0.
004424324259161949, -0.014995420351624489, 0.034529995173215866, -0.0081529151648283, -0.03431384265422821, 0.011091207154095173, -0.
010618378408253193, 0.012131430208683014, 0.018224162980914116, -0.0015189617406576872, -0.02863990142941475, -0.00024929040228016675, 0.
014049762859940529, -0.00029699545120820403, 0.007680086884647608, -0.001062175608240068, -0.011395168490707874, -0.025181498378515244, 0.
035583727061748505, -0.01783239096403122, 0.00150291936006397, -0.01995336450636387, 0.011131735518574715, 0.026356814429163933, -0.
039771635085344315, 0.0036509118508547544, -0.005842810031026602, 0.0051909820176661015, -0.024897800758481026, -0.021155700087547302, -0.
02557327039539814, -0.027599679306149483, -0.017859410494565964, 0.01653549075126648, -0.04266264662146568, 0.01848084107041359, -0.
01048328448086977, -0.0016042398056015372, -0.0048870206810534, 0.006758071016520262, -0.01417134702205658, -0.03420576825737953, 0.
018859105184674263, -0.004505380522459745, 0.001364448107779026, -0.00237765209749341, 0.0393393374979496, 0.020574796944856644, 0.
01642741449177265, -0.015225079841911793, -0.012989276088774204, -0.01323244534432888, 0.0012521513272076845, 0.002406359650194645, 0.
010321171954274178, 0.017508165910840034, -0.01508998591452837, 0.03042314015328884, 0.0005180005682632327, -0.013414821587502956, -0.
03307098150253296, -0.017805373296141624, -0.019115783274173737, 0.008328537456691265, 0.024965347722172737, 0.01098313182592392, 0.

Figure 12: Embedded Data In Vector Database.

6.1.1 Extracting Data

With any static PDF or user uploaded PDF, we must extract the data within it before storing it in our vector database. We develop two wrapper functions for this purpose – pdf_processing() for uploaded PDFs, and svg_processing() for uploaded SVGs. At present, only PDF processing is supported.

A PDF may contain both text and chart data, so we both read the text data from this PDF as well as parse the document as an SVG using the Inkscape library to extract data from included charts. The parsed text data is passed to a LLaMA SimpleDirectoryReader as a text document, and the parsed chart data is passed to a LLaMA SimpleDirectoryReader as a chart document. Due to hallucinations experienced in testing, we currently disregard parsed chart data and focus primarily on parsed text data. To overcome this limitation, we instead upload a description of chart data for a particular graph to be included as a chart document.

6.1.2 Recreating Charts

In addition to the ”text engine” and ”chart engine” tools available to our LLM, we develop a ”bar chart tool” and ”line chart tool” that create bar charts and line charts using Matplotlib based on data provided by our vector context. This data is passed as a dictionary, tuple, or list of values by our LLM in response to a user's query, allowing us to modify user charts and return an image generated from this new data.

6.1.3 Processing User Queries

With data loaded into the vector database, we can now process incoming user queries. Using either GPT-3.5 or GPT-4-Turbo, we select the proper tool in response to a user's query. For instance, a user's prompt to change the provided inflation chart to use red bars might call on the "bar chart tool".

6.2 Frontend Development

With a full-fledged backend developed, we moved towards establishing a frontend using Streamlit. The frontend must be able to handle incoming user queries and respond appropriately with both text and images, as well as describe how to use our tool appropriately. We accomplished this through five main areas:

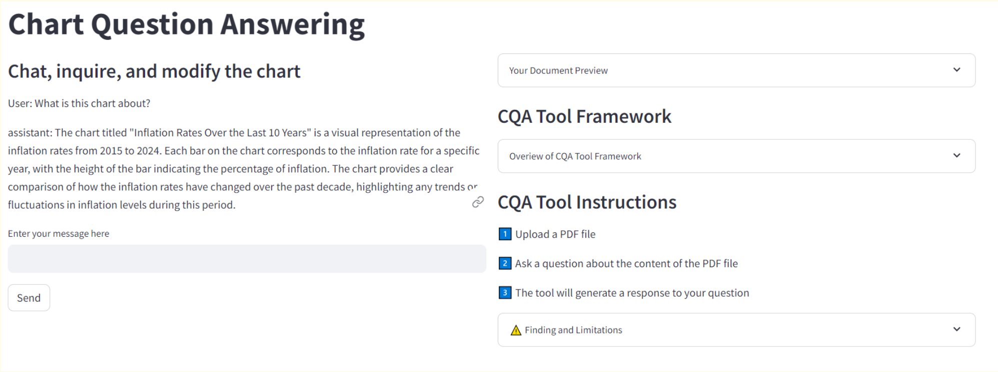


Figure 13: Frontend In Streamlit.

1. Text Entry and Response: a user can enter a query, hit the send button, and receive a text response or charts generated from the assistant.

2. Document Preview: a user can view the document currently used as context for their query.

3. Overview Diagram: a user can view our pipeline and how their data is processed.

4. Instructions: a user can recognize how to use our tool to answer questions about their chart.

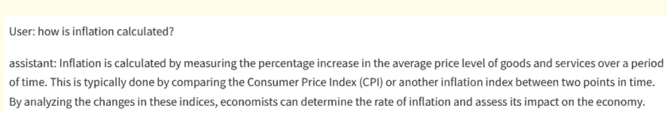
5. Findings and Limitations: a user can view our results from this project and challenges we encountered in our design.

We intended for users to additionally be able to upload their own PDF documents, but due to development challenges with processing files into a live database and LLM, we instead focus on statically uploaded PDFs.

6.2.1 Query Types

A user may query our tool with four types of queries:

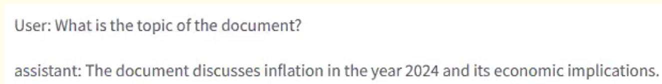
1. Information In Neither Text Nor Chart: Questions about general knowledge, such as "how is inflation calculated?" will be answered from the LLM's general knowledge training. Below, we can see the LLM responds to a user query on the topic of inflation with a general response that references neither provided text or chart data:"


User: how is inflation calculated?

assistant: Inflation is calculated by measuring the percentage increase in the average price level of goods and services over a period of time. This is typically done by comparing the Consumer Price Index (CPI) or another inflation index between two points in time. By analyzing the changes in these indices, economists can determine the rate of inflation and assess its impact on the economy.

Figure 14: A General Query Retrieves General Knowledge Results.

2. Information Within Text But Not Chart: Questions such as "What is the topic of the document?" will be answered from the "text engine" that pulls from text parsed from the document itself. We see the LLM responds appropriately from only text data without consulting any specific information from the included chart:

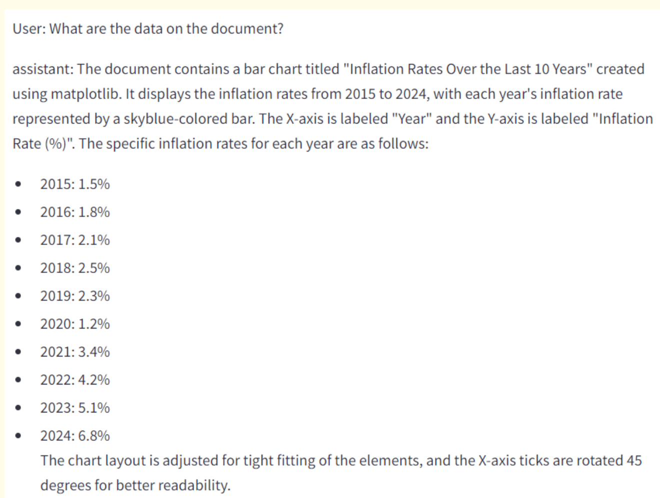


User: What is the topic of the document?

assistant: The document discusses inflation in the year 2024 and its economic implications.

Figure 15: Describing the Topic of the Document.

3. Information Within Chart But Not Text: Questions such as "What are the data on the document?" will be answered from the "chart engine" that currently answers according to statically passed chart data in text form, but ultimately will be interpreted from charts within a PDF in later pipeline iterations. Below, we see the LLM mentioning specific numerical figures included within the chart, but not referencing general information from the provided text:



User: What are the data on the document?

assistant: The document contains a bar chart titled "Inflation Rates Over the Last 10 Years" created using matplotlib. It displays the inflation rates from 2015 to 2024, with each year's inflation rate represented by a skyblue-colored bar. The X-axis is labeled "Year" and the Y-axis is labeled "Inflation Rate (%)". The specific inflation rates for each year are as follows:

- 2015: 1.5%
- 2016: 1.8%
- 2017: 2.1%
- 2018: 2.5%
- 2019: 2.3%
- 2020: 1.2%
- 2021: 3.4%
- 2022: 4.2%
- 2023: 5.1%
- 2024: 6.8%
  The chart layout is adjusted for tight fitting of the elements, and the X-axis ticks are rotated 45 degrees for better readability.

Figure 16: Querying About Chart Data.

4. Prompts to Recreate Chart Data: Prompts to change existing data, such as "recreate the bar chart from the chart data. Make the bars red" will pull

from the vector database of chart data and call an appropriate tool, such as "bar chart tool," to recreate this chart in Matplotlib. The recreated image is then shown to the user below their prompt:

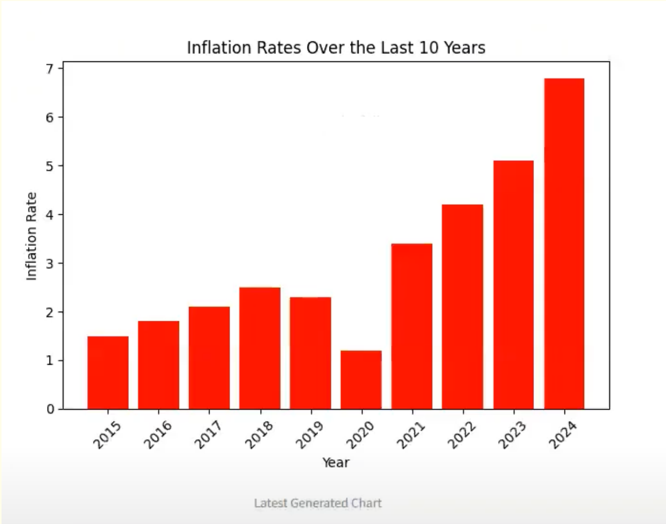


Figure 17: Recreated Chart In Different Color.

The user also has the ability to change the type of chart to be a line chart through the query "change bar chart to line chart:"

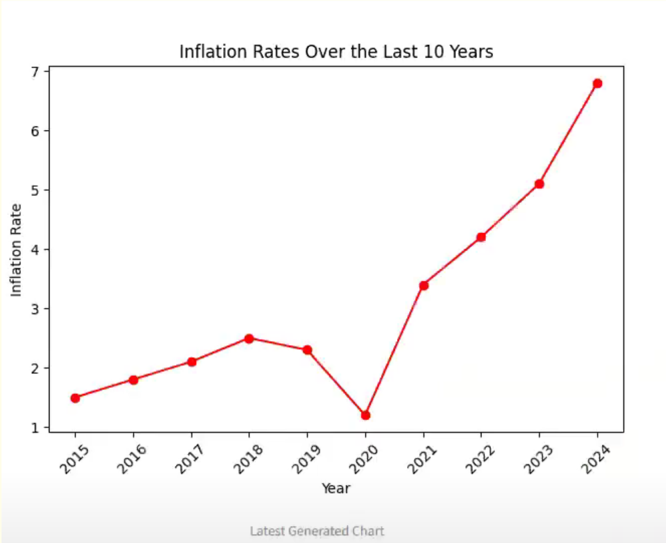


Figure 18: Recreated Line Chart.

By prompting the model to focus on only specific data points, the user can also recreate figures that reflect only a portion of the original chart. An example query is ”focus only on data from 2020 to 2024 on the line chart:”

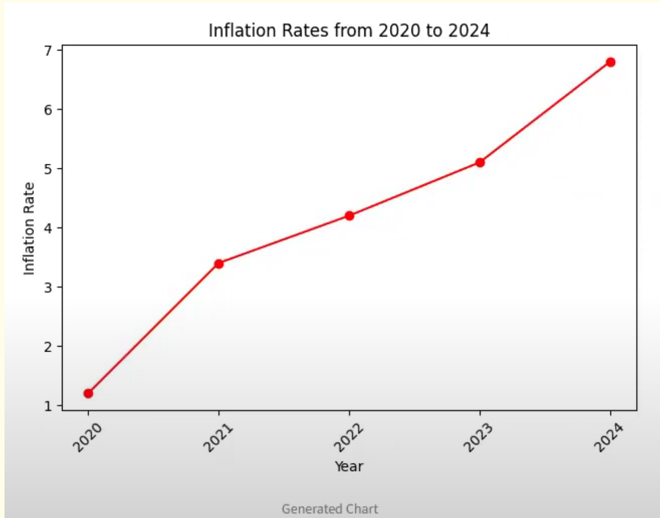


Figure 19: Line Chart With Only Certain Data Points.

Queries for which no context is available, for instance ”what is the inflation rate in 2014?” will result in the LLM responding that such data is unavailable, rather than hallucinating or extrapolating incorrect data:

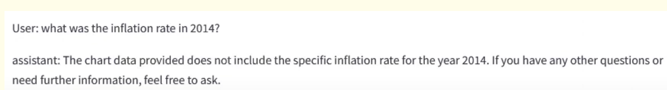


Figure 20: Prompting When Data is Unavailable.

7 Evaluation

Our tool are able to provide interactively chat assistant to gain deeper knowledge about the chart with in-context report. This tool also give user flexibility to transform chart to other type of chart. However we find some limitation when working on this project:

1. OpenAI Credits: Experimenting consumed substantial resources, especially when using GPT-4-turbo.
2. Hallucination occurred after several conversations, indicating that it can only sustain a conversation for three to four interactions.
3. GPT-3.5 facilitated a faster workflow, but it yielded poorer results.
4. The process of parsing chart and text data from a PDF could be improved. Currently, we use the Inkscape library, which is incompatible with Windows operating systems.
5. Occasionally, the agent erroneously invokes the incorrect function.
6. There is a noticeable delay between the prompt and the response from the language model.

For future work we consider to explore other methodology to extract data from PDF, other framework for UI to get more flexibility and reduce delay, and also consider local model to reduce dependency on OpenAI API credit.

References

[1] E. Hoque, P. Kavehzadeh, and A. Masry, "Chart question answering: State of the art and future directions," in *Computer Graphics Forum*, Wiley Online Library, vol. 41, 2022, pp. 555–572.

[2] Y. Peng, S. Lin, Q. Chen, *et al.*, "Chatgraph: Chat with your graphs," *arXiv preprint arXiv:2401.12672*, 2024.

[3] Z. Wang, Y. Li, J. Wu, J. Soon, and X. Zhang, "Finvis-gpt: A multimodal large language model for financial chart analysis," *arXiv preprint arXiv:2308.01430*, 2023.

[4] Y. Han, C. Zhang, X. Chen, *et al.*, "Chartllama: A multimodal llm for chart understanding and generation," *arXiv preprint arXiv:2311.16483*, 2023.

[5] W.-G. Chen, I. Spiridonova, J. Yang, J. Gao, and C. Li, "Llava-interactive: An all-in-one demo for image chat, segmentation, generation and editing," *arXiv preprint arXiv:2311.00571*, 2023.

[6] D. Masson, S. Malacria, G. Casiez, and D. Vogel, "Charagraph: Interactive generation of charts for realtime annotation of data-rich paragraphs," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–18.

[7] H.-K. Ko, H. Jeon, G. Park, *et al.*, *Natural language dataset generation framework for visualizations powered by large language models*, 2023. arXiv: `2309.10245 [cs.HC]`.

[8] LlamaIndex. "Using llms." (2024), [Online]. Available: `https://docs.llamaindex.ai/en/stable/understanding/using_llms/using_llms/` (visited on 04/01/2024).