University of Pennsylvania
Department of Electrical & Systems Engineering

ESE 451: Senior Design

Spring 2022

Team 7

Viz

VIZ

Data Visualization made easy

Team Members:
Effie Guo: ESE, WH, effieguo@seas.upenn.edu
Shannon Lin: ESE, ROBO, WH, shannon1@seas.upenn.edu
Anna Nixon: CIS, anixon@seas.upenn.edu
Sophie Thorel: ESE, WH, sthorel@seas.upenn.edu
Gregory Zhu: CIS, gregzhu@seas.upenn.edu

Advisor:
Professor CCB: ccb@seas.upenn.edu

Table of Contents

I. Executive Summary

We are propelled into an even more data-driven world than ever before, where learning to visualize and understand your data is now an essential part of decision-making in any industry. Simultaneously, there is a growing demand for no-code solutions to generate faster and simpler outputs to obtain the low-hanging fruit of simple data analysis, which begins with easy visualization.

Viz is a low-cost, easy-use, educational data visualization platform. Viz aims to eliminate the technical requirements and steep learning curve required to create quick and easy data visualizations from a data set. Our platform allows the user to enter a data set (.csv format) and a simple plain-text English command to generate the associated Python code and plot a visual graph from the dataset (eg. line plot, scatter plot, histogram, word cloud, pie chart, etc).[1] We have also embedded an interactive Python interpreter to allow the user to modify the code and see the updated graph, which would serve as an educational launchpad for more complex analysis.

On the back-end, Viz utilizes OpenAI's GPT-3 Davinci text neural network model to convert this plain-text query into Python code that will output the desired graph, along with the code. GPT-3 is the largest language neural network ever built, with over 175 billion ML parameters.[2] We have fine-tuned the model to our use case. Our goal with this Senior Design project is to demonstrate a proof-of-concept for one example of how models such as GPT-3 will fundamentally alter our society's relationship with coding and data.

II. Overview and Motivation

Given the exponential increase in the amount of data being generated and used today, it is more important than ever to empower our society to extract helpful information from it. However, there is a steep learning curve blocking many individuals from being able to create simple visuals from datasets.

<u>Value Proposition & Users</u>

This platform is aimed at two main groups of users. The first group consists of the small business owners and professionals who do not have technical experience to plot more complex graphs from a dataset, but who need to produce quick visuals to validate or better understand insights and trends from data. Viz allows them to instantaneously visually understand their data without having to learn Python or resort to a Data specialist immediately. While Viz does not replace the Data Scientist, it is meant to be a starting tool for the user to understand their data that would otherwise have been blocked by the steep technical learning curve that comes with traditional visualization tools. The second group is centered around the purpose of education. Educators wishing to display the power of data and visualization to younger audiences could use Viz to simultaneously teach students about the types of graphs and outputs that can be

---

[1] See video for a quick tutorial of Viz: https://www.youtube.com/watch?v=qHN711le3CY.

[2] Language Models are Few-Shot Learners: https://arxiv.org/pdf/2005.14165.pdf.

generated from data, and better understand the code that generates them. Our product includes the exportable Python code generated and an embedded window where the user can edit the code from our platform and see the resulting changes in real-time on the output graph. This additional learning tool provides a more long-term solution to bridging the knowledge gap about data visualizations.

III. Business Analysis

Market opportunity & customer segments

Data analysis is becoming an increasingly important role in small and large businesses as exponentially more data is being created with time. 67% of small businesses spend more than $10k/year on analytics - according to a survey by Onepath.[3] In addition, there is a growing desire for “low-code solutions" to allow non-technical individuals to understand data, as seen by the explosion of low-code data analysis platforms for small businesses. Businesses are exponentially turning away from Excel in search for more efficient and accessible tools that can more easily be integrated with other platforms: the Enterprise Times reported that 21% of businesses in the US are moving toward other software solutions.[4] For this reason, Viz has high potential for success among small businesses. While we are not competing with more complex data analysis platforms, we are providing companies with a faster and simpler alternative to visualize data as a starting point for more complex analysis.

The initial market focus will be in the US, specifically for the 6 million firms with under 100 employees in the US[5], because we assume that these small companies would most likely be looking for quick and cheap data tools versus larger companies with more sophisticated products.

The second market opportunity is in the education space with 2 main customer segments: educators using this tool for teaching data visualization to young elementary and middle school children and the young children themselves who would use this tool for their own learning. With 87,498 elementary schools in the US,[6] this is a high potential customer segment given the higher number of paid licenses we expect per school (in order to charge per student - see revenue model section below).

---

[3] https://smallbiztrends.com/2020/03/data-analytics-trends.html#:~:text=The%20biggest%20takeaway%20for%20small,to%20make%20more%20informed%20decisions

[4] https://www.enterprisetimes.co.uk/2016/04/22/are-spreadsheets-on-the-brink-of-extinction/

[5] https://sbecouncil.org/about-us/facts-and-data/#:~:text=Based%20on%20data%20from%20the,in%202018%20(latest%20data)%3A&text=Firms%20with%20fewer%20than%20500,99.7%20percent%20of%20those%20businesses.&text=Firms%20with%20fewer%20than%20100%20employees%20accounted%20for%2098.1%20percent

[6] https://www.statista.com/topics/1733/elementary-schools-in-the-us/#dossierKeyfigures

Competition

There does not yet exist a direct competitor in our field where the user can input a query in plain English, and output a graph and code on a user-selected dataset. The two closest types of competitors are the 1) educational platforms such as Wolfram Alpha and easy graphing platforms such as onlinecharttool or plotvar; and 2) more complex, low-code data analysis platforms for business analytics (such as domo.com, zoho.com, monkeylearn, or Excel).

However, Viz is differentiated from those platforms. Wolfram Alpha is designed to process mathematical equations and cannot convert a plain-text query to code, nor does it take in a dataset to compute tailored visualizations. Easy educational chart-making tools have non-children-friendly UIs that require the user to go through several steps to select which columns to graph. Domo, Zoho and monkeylearn are all subscription-based platforms which are essentially more user-friendly and sophisticated versions of Excel, where you have to learn to use the platform's UI to output the desired result (again, costly and not educational-friendly). English to code conversion doesn't exist in any of these platforms. Also, the prices for these platforms are cost-preventative for the size of organizations that we are looking to serve. Excel has its limitations in collaboration and user-friendliness, and again does not provide the ability to export or convert to a coding language. We are thus the only platform that allows for no-code, plain-English prompted data visualization while providing the unique ability to export the generated python code.

Cost

The only cost for Viz is utilizing OpenAI's davinci-text model, totaling $0.0045 per visualization. This model costs $0.006/1K tokens where 1000 tokens are about 750 words. Each time a user inputs a query (average 25 tokens), we send 3-5 additional input/output examples to the model to utilize the few shot learning approach. Each additional input averages 25 tokens and the output averages 150 tokens based on experimentation. This totals 25 + 4*(25+150) = 725 tokens per user query = $0.0045 per user query. With our current senior design's budget of $400, we can support about 90K user queries before utilizing team's out of pocket costs.

The server database is hosted for free on AWS RDS (relational database service) free tier. Because only one instance will be running at all times, the maximum hours spent per month is 31*24 = 744 which satisfies the maximum allotted 750 free hours. To deploy the first iteration, AWS Elastic Beanstalk will be utilized for free.

Revenue Model & Market Sizing

Currently, the OpenAI neural network model we use is under beta, so we cannot immediately commercialize our product. However, we expect the model to be open for productization within the next few months given the last beta model's timeline. Once we are able to charge for utilization, a freemium model will be put in place. Users will only be able to use our platforms up to 5 times a month. After this limit, they will have to pay for a monthly subscription of unlimited queries for $6/month. The break-even point given a $0.0045 cost per query comes down to 1333 queries per month, or about 5 per working

hour (assuming 8 working hours per day), which is more than we expect users to be using our platform. For elementary schools, we would want to charge at a discount to encourage schools to buy bulk licenses for each student. Charging $5 per student would again be profitable by expectation (students would need to input >32 queries/day).

Since our cost structure is constant per usage for now, revenue must grow exponentially for Viz to scale. A subscription model, with proper marketing and ease of adoption, would allow for scaling by increasing the number of users paying for licenses exponentially. Long-term, we would explore possibilities for a contract with OpenAI to lower the costs per usage.

Given the entire space of the 6 million small businesses under 100 employees[7] and all 32.6 million elementary school students[8], TAM totals $6.5B. Viz's SOM is estimated to be $32M. This comes from the sum of $10.8M from schools, assuming we can draw contracts with 1% of elementary schools and from these schools, about half the students (this product is mostly targeted towards grades 2-5), and $21.6M from small businesses, assuming we can get 1% of the 6 million small businesses and get subscriptions form 5 employees per company. These assumptions are based on market adoption of competitors' tools such as Domo in companies, as well as a substantial surge in adoption of digital learning platforms that Viz benefit from with low costs, high speed and clean simplicity.

IV. Technical Description

IV.A Specifications and Requirements

In our minds, the look and feel of the final deliverable was quite clear. As we were targeting non-technical professionals and younger audiences, we wanted a simple to understand interface to use our product (think Desmos, but for data visualization instead of plotting). Because of this, we decided upon a clear text prompt and file upload button with no complex options in the home page so that users could dive right in. More functionality and information is displayed after pressing the "start graphing" button that allows users to modify code and manipulate their graph. We landed on this approach because of the following requirements that we defined:

- Easily accessible with Internet
- Understood by non-technical users and low barrier to entry
- A valid CSV file inputted by user
- Balance simplicity of use with flexibility for power users
- Quick webpage response times

---

[7] https://sbecouncil.org/about-us/facts-and-data/#:~:text=Based%20on%20data%20from%20the,in%202018%20(latest%20data)%3A&text=Firms%20with%20fewer%20than%20500,99.7%20percent%20of%20those%20businesses.&text=Firms%20with%20fewer%20than%20100%20employees%20accounted%20for%2098.1%20percent

[8] https://www.statista.com/topics/1733/elementary-schools-in-the-us/#dossierKeyfigures

As we used GPT-3, which is a proprietary machine learning model from OpenAI that we were granted private beta access, we were subject to OpenAI's distribution requirements as well. Constraints under senior design regulation and OpenAI specifications included the following:

- <$400 for development
- Follow OpenAI's safety best practices and usage guidelines:
    - Think like an adversary
    - Limit your input and output lengths
    - Know your customers
    - Rate-limit pace of usage
    - Filter sensitive and unsafe content
    - Keep your application on-topic
    - Capture user feedback
    - Keep a human "in the loop"
    - Draw upon validated content
    - Gain access to private Beta version of Codex

In the end, we were able to offer 8 different plot types. Users are able to specify the plot type in their query and modify the python code used to generate the plots in our graph landing page. See Appendix Figure 1 for a visual of our various plot types Viz can generate.

IV.B Alternative Solutions and Iterations

The end goal is to make data more accessible to non-technical users. The top 3 solutions we considered to address that problem are as follows:

- Use web crawler to pool data on certain topics and visualize in usable playground (a search engine for graphs and datasets)
- An excel-like program that is more user friendly (the kids programming language Scratch but for Excel)
- A text-to-visualization program (give me your desired graph in words and I'll give you the visualization that you want)

The last and current approach was ultimately chosen because of the innovative possibility to meet the non-technical users' needs but also complement technical users' preferences. Text to visualization gives flexibility for customizing your graphs and providing a large range of visualization types while also being very accessible. Additionally, there does not yet exist a lightweight hosted website application that performs a plain text English to visualization conversion so the market is less crowded, unlike the other two top options.

Throughout the year, our project went through many important iterative improvements. The central improvement was the fine-tuning of our GPT-3 model using OpenAI's fine-tuning feature. This allowed us to significantly improve the accuracy and relevancy of our results. We also continued to iterate on our back-end pre- and post-processing pipelines to provide better results, stability, and response times.

IV.C Societal and Economic Considerations

Societal and economic considerations were central to the technical approach of our project. As we are an educational and productivity boosting project, user experience was central to the frontend design. A lot of effort was put into our front-end design work to make the experience inviting and as frictionless as possible.

Because we used GPT-3, an incredible language model, we also needed to consider the ethical ramifications of such a powerful tool. As per OpenAI's guidelines, we needed to make sure that our product could not be used for hate, harassment, violence, self-harm, adult, political, spam, deception, or malware purposes.[9] To do this, we followed OpenAI's safety best practices closely. Our tool should be a fun, easy graphing tool and as such, we limited the scope of our API usage to only graphing content. Any query that was not graphing related would simply direct users to an error page with a helpful message. Any code that was executed in our backend ran in a secure environment, and all datasets were handled with caution. In addition, our query entry box on our home page was cleaned to prevent code injection attacks to prevent malicious code from being sent to our server.

From an economic standpoint, we wish to have as low costs as possible to allow the technology to be as accessible as possible. In our design considerations, we tried to keep costs low to be able to offer the service to our users as a low-cost subscription model. We decided to host our application on AWS as this offered pricing flexibility using Amazon Web Services' economy of scale. Using the OpenAI API also incurs a low cost per token (which is about 1 word), so we set a query size limit. In practice, we were never even close to using up our query size limit that was set. Having one, high-quality API call per graph request allowed us to keep costs low without compromising on the quality of our results.

IV.D Approach

The central technology of our product is OpenAI's GPT-3. This model, GPT-3, is an extremely large and powerful language model which has 175 billion parameters and was trained on 45TB of text data. GPT-3 can write essays, act as a chat bot, and generate code. We use this last ability as a key technology for generating custom visualizations from almost any plain-text English prompt.

We began the year using OpenAI's Codex variant of GPT-3 as the basis of Viz. Codex is a model built to take natural language prompts and generate code output. When we began the process of creating Viz, we thought that this would be the ideal solution to the problem we were trying to address since it already had the baseline capability of generating code from plain-text prompts. However, our testing showed that it often gave incorrect and inconsistent outputs. Due to Codex being in closed beta at the time of development, we were unable to fine-tune this model in any significant capacity.

---

[9] https://beta.openai.com/docs/usage-guidelines/safety-best-practices

After speaking to our advisor and an engineer from OpenAI we decided to switch from using Codex to using the Instruct model with few shot learning. OpenAI's Instruct model is not trained specifically to create code but can follow instructions and is very responsive to prompt engineering (adding examples in the prompt to tune the model on a few data points). There was a model that could be fine-tuned called the Completion model but since it was built to complete prompts rather than follow the instructions specified in them we were advised against using it.

As we continued development, the Instruct and Completion models were rolled into one singular general purpose model that *could* be fine-tuned called the Text model. We manually generated example prompts and corresponding code completions and created a fine-tuned Text model that proved to have marked improvements over our original baseline Codex model. This fine-tuned model became the basis of our final design.

The second major improvement was the front-end user interface. We continued to add more functionality to improve the user experience including better navigation, dataset feedback, an interactive python console, and a more streamlined design with bolder font and instructions.

We began with a basic mockup wireframe using Figma based on our project requirements. This mockup consisted of the ability to upload a CSV and a plain text prompt, followed by ability to view the result on a separate screen. Users could also click a button to export the code that was generated. See Appendix Figure 2 for our Figma mockup.

Once we migrated our design to our web application we simplified the UI further in order to reduce the number of clicks users had to take in order to access their generated graph and data.

After speaking with our classmates and instructors during our MVP demo days, we decided to make some adjustments to our UI based on the feedback we received. We added a display of column names once a file had been uploaded in order to help users generate better prompts, and we also incorporated an interactive code editor for users to edit and play around with generated code.

Shown below is the final iteration of our UI. When a dataset is uploaded, column information is automatically displayed to give users more information to improve their graph query. Appendix Figure 3 contains our complete UI from start to finish.

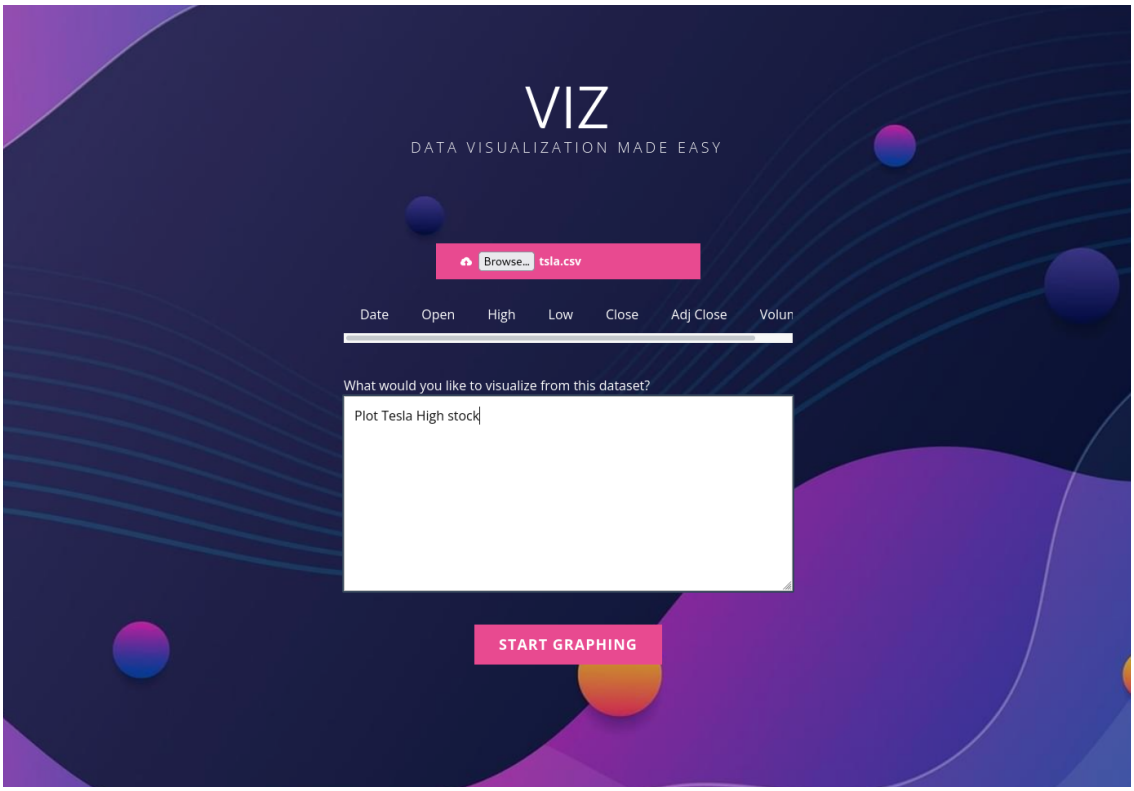


Figure IV.D 1: Viz Final UI

IV.E Technical Description

Viz is a full-stack web application. For our web framework we chose to use Django[10] for its ease of use, data flexibility, and because it is a python web framework. Choosing a python language web framework was especially important as we chose our graph generating code to be in python. This is because python is one of the most popular languages for data science. Python has excellent libraries such as matplotlib[11] and pandas[12] which are the backbones of our graph visualizations. We also used the python libraries: numpy, scipy, and wordcloud for our data visualizations. See Appendix Figure 5 for our system architecture.

Our application begins with a home page where users input their dataset and graph query. Once a file is uploaded, it is analyzed locally to extract column types and this information is then displayed to the user.

[10] https://www.djangoproject.com/

[11] https://matplotlib.org/

[12] https://pandas.pydata.org/

Once the user is happy with the query and file upload, they can select the “Start Graphing” button on our home page. This sends their dataset and query to our backend server. We first pre-process their input. This process includes:

- Cleaning their query of any malicious input
- Spell-checking their query input
- Analyzing their csv for well-formedness (proper column names and check for empty data)
- Augment the query to include special key words that are used by GPT-3 to improve the relevancy and accuracy of the results. We found that this step was extremely important to ensure good results and a lot of experimentation was done in this step to find the optimal augmentation.

Then, the query is sent to our fine-tuned version of GPT-3. In our application, a fine-tuned GPT-3 Text model takes in a query and outputs python code that utilizes our selected python libraries (matplotlib, pandas, etc.) to generate visualizations. We take the python code generated by GPT-3 and post-process it by inserting the proper file names, deleting unnecessary code, and running the python code to generate the graph visualization. The python code is run in a container to ensure security. We then send the graph visualization and code back to the user. All of this happens within a few seconds.

When the user receives the graph, they also receive the code which they can then modify and rerun in a live Python interpreter. Users are able to zoom in and out of the graph and move it around to get the best view of the visualization.

Fine-Tuning Process

In order to improve the Davinci-Text model that we settled on as our final GPT-3 model, we fine-tuned using manually generated prompts and corresponding code. We wanted to introduce capabilities such as Venn diagram and word cloud generation that we found had little or no support with the original Codex model so we divided the training data generation into the seven visualization categories we wanted our solution to be adept in. For each category we generated a small set of sample prompts from different example datasets we had found. Each prompt contained the user inputted query, the column names of the csv, and the csv name. For each prompt we also wrote the corresponding code that would output the desired graphical result. For part of this data generation process we experimented with sending the prompts through the Codex model and editing the output code to fix any errors. However, the majority of our examples were generated manually from scratch.

This data was then formatted into a JSON file in accordance to the OpenAI API requirements and used to fine-tune Davinci-Text, the largest text model available, over 4 epochs. Once the fine-tune process was completed, we modified the calls from the web app to the model to include the same prompt information as our training data.

In order to measure if we were able to create performance improvement by fine-tuning we decided on a two-sided approach to evaluate our baseline (Codex) model and our new fine-

tuned model. For our analysis we defined accuracy as results that were both relevant and executable. This was based on our initial qualitative analysis of the output code that was being generated where we saw that some code was able to execute, but it did not provide the visualization requested—while other times we saw code that had a few minor errors preventing execution, but this code was actually closer to a relevant answer than other outputs that could execute.

We evaluated relevance and executability separately. Executability was calculated automatically with a script that attempted to run the output code for each prompt in our training data while relevance was evaluated manually. The scores for both were combined in order to create our final measure of accuracy as the number of examples that were both relevant and executable over the total number of examples in the dataset.

IV.F Final Status of Project and Test Results

We found that our fine-tuned model showed a dramatic increase in accuracy over the original baseline (Codex) model. Accuracy jumped from just 16.22% to 72.97% with the fine-tuned model presenting relevant results twice as often while also presenting a higher proportion of executable results. Although the test prompts came from the same distribution as the training data, as both were written by the same group of people, we believe the magnitude of difference is significant enough to validate our fine-tuning process.

Figure IV.F 1: Pre- and Post- Fine-Tuning Test Results

IV.G Overall Evaluation

Our fine-tuning test results provide a quantitative measurement that allows us to assess how much improvement fine-tuning has versus the base Davinci-Text model. However, our end goal as set in the requirements is the user experience. Using our fine-tuning, we were able to offer a much wider range of plot types while simultaneously improving the accuracy of the model. This meant that the application was more consistent and more expressive. Paired with the numerous quality of life improvements to the UI, our application was a lot more polished and user friendly. Because of this, our team can confidently say that we met our requirements for the project.

IV.H Conclusion

Over the course of the year, we created a full-stack interactive web application that takes in queries and robustly outputs graphs that are relevant to the user through a simple user interface. We created a clean and beautiful user experience and an easy to use workflow. In the backend, we used a powerful technology GPT3 by OpenAI which allows us to generate python code and beautiful visualizations. Using a special fine-tuned version of the model,

we were able to significantly improve the quality and relevancy of our automatically generated graph code which allows our program to handle 8 different plot types. Because of these improvements, Viz is a powerful graphing tool even in the hands of an inexperienced user.

V. Self-learning

This project has been an immense learning experience for all team members. At the start of our project, our team was not very familiar with OpenAI's models and architecture. We did, however, have decent background knowledge in machine learning from classes like CIS519 and CIS520, which gave us a foundation in understanding neural networks. Two of our team members also took our advisor CCB's Introduction to Artificial Intelligence class (CIS521), which aided in providing domain knowledge in large language models specifically.

With our limited background knowledge, we had a steep learning curve in gaining familiarity with the OpenAI platform. To gain familiarity with the platform, we read OpenAI's documentation and guides–as well as their website blog posts. Our learning primarily occurred through experimenting with the OpenAI playground, where we tested many models along the way from the Codex model to the Davinci instruct model to our final Davinci text model.

Multiple advisors guided us in our learning process along the way. We learned from our classmate, Ben Wang (co-op at OpenAI), about how to toggle the parameters in the playground to control the randomness of the output. Our advisor, CCB, suggested methodologies for fine-tuning the OpenAI models. To this regard, we read research papers on zero-shot/transfer learning and fine-tuning OpenAI. Our team members also had the opportunity to participate in CCB's organized Q&A session with an OpenAI employee from the R&D team. There, we engaged in dialogue about how to improve our project. On a different call, we also interfaced with Juston Forte, who suggested alternative methods of fine-tuning to improve our accuracy.

Finally, we designed a front-end website from scratch. This involved learning to write CSS code and building a website using Django. We had to learn how to run our code and then display the resulting graph on our webpage, as well as design appropriate user interface features (eg. input boxes, buttons, loading pages, etc). We also investigated how to create our own sandboxed Python interpreter from scratch.

VI. Ethical and Professional Responsibilities

Given the incredible potential of GPT-3, we ensured we followed OpenAI's charter which describes the principles used to execute OpenAI's mission: to ensure that artificial general intelligence benefits all of humanity. One potential risk of our platform is that it creates misleading or erroneous visuals by, for example, not understanding the user's input query. We are mitigating this risk by cleaning up the query before it is inputted to GPT-3, and we also provide to the user the opportunity to check column names in their dataset.

We also have an ethical responsibility to ensure our platform is used with good intentions. We currently have no way of screening the CSV file for potentially problematic or sensitive datasets; however, we can definitely explore this in the future.

VII. Team Meetings

As a group, we met twice a week for between 30-45 minutes each session. In between sessions, we often worked independently on either testing out Codex, doing research on the model, or building parts of our backend.

We met with our advisor a couple of times at the start of the semester, and once at the end. We had more regular 30-minute meetings with him in the spring semester, as he was less busy this semester. We also were able to briefly engage with an employee of OpenAI and connected with him over Zoom.

VIII. Timeline & Milestones

All deadlines were met: see Appendix Figure 4 for a chart of our milestones.

IX. Teamwork

This semester, we tried to avoid splitting into separate workloads too early in order to first get a very clear sense of what our desired outcome would be. We all conducted initial research into OpenAI and would share our learnings and explore together during our bi-weekly meetings.

We started splitting on deliverables, where we had a UI subteam working on revamping the website (Effie helped with the python interpreter, Greg and Anna with the frontend), and a team working on the cleaning of the query, fine-tuning and backend (Anna, Shannon and Sophie). This worked well as we were able to make best use of our teammates' capabilities.

X. Budget & Justification

Software projects are relatively inexpensive and in the end cost $0. While we initially estimated a budget of $400 to fine tune the model in order to display a larger variety of visualization types, we were able to secure free credits from OpenAI by meeting with them over Zoom to share our project and goals. They were excited enough about Viz to lend us some credits which we did not fully use and therefore did not have to spend additional funds. We decided not to deploy Viz due to OpenAI's ongoing and changing guidelines. OpenAI, per the terms of use, ultimately decides whether a model can be released in another application. Thus, no additional funds were used for deployment.

XI. Engineering Standards & Compliance

Viz focused on three main compliance standards: OpenAI Charter, ISO 9126 (90003) Quality Management, and IEEE P7002 Data Privacy Standard.
OpenAI Charter includes satisfying the following: Broadly Distributed Benefits, Long-Term Safety, Technical Leadership, and Cooperative Orientation. If deployed, Viz is a platform that is accessible to everyone cost-free which is considered broadly distributing benefits. There is no interaction between users, and the users' uploaded datasets are immediately deleted after querying into it for the visualization, thus enabling long-term safety and data privacy. We consulted both leaders from Penn Engineering and OpenAI themselves to create a collaborative product.
Quality Management includes the following: functionality, reliability, usability & re-usability, efficiency/maintainability/portability. Through extensive testing, the model accuracy (measured by both relevance and execution) exceeds 70% which meets the cited performance criteria of over 60%. Viz is lightweight and easily maintainable as a software platform.

XII. Work done since last semester

In our fall minimum viable product (MVP), our website generated functioning code and visualization of user inputs for one specific testing dataset: quantitative data about Tesla's stock price. We chose this dataset since it lent itself to many possible questions that translated well into plain-text English queries.

This spring semester, we refined our fall MVP and added many more features. In terms of refinement, we fine-tuned the OpenAI Davinci text model to improve the accuracy of our results as well as double the number of plot types that we offered. Support for box plots, pie charts, venn diagrams, and word clouds was added to our product. Furthermore, we added the ability for users to upload their own datasets (any .csv file) and check the column names in their dataset prior to plot generation. In the case of poorly formed queries or unintelligible output from GPT-3, we now redirect users to a fail page from which they can modify their query. For cases where a graph does appear, users are able to modify the python code generated in an interactive python console. Our product is now fully functional with support for any csv dataset visualizations for qualitative plot types in addition to our existing quantitative plots. The addition of the interactive console provides the user much more flexibility and customization.

XIII. Discussion and Conclusion

Given the problem statement of making data more accessible to users, Viz is the ideal solution: a fast, easy-use, educational visualization tool. This website application converts plain text English visualization needs on the user's uploaded dataset to working Python code and visualization display. We fine-tune OpenAI's davinci model to support more visualization types and observed user interaction with our site to improve the user experience. Viz is a unique proof of concept that justifies the possibility of data visualization without coding, a world that we hope to continue exploring in the future.

XIV. Appendix

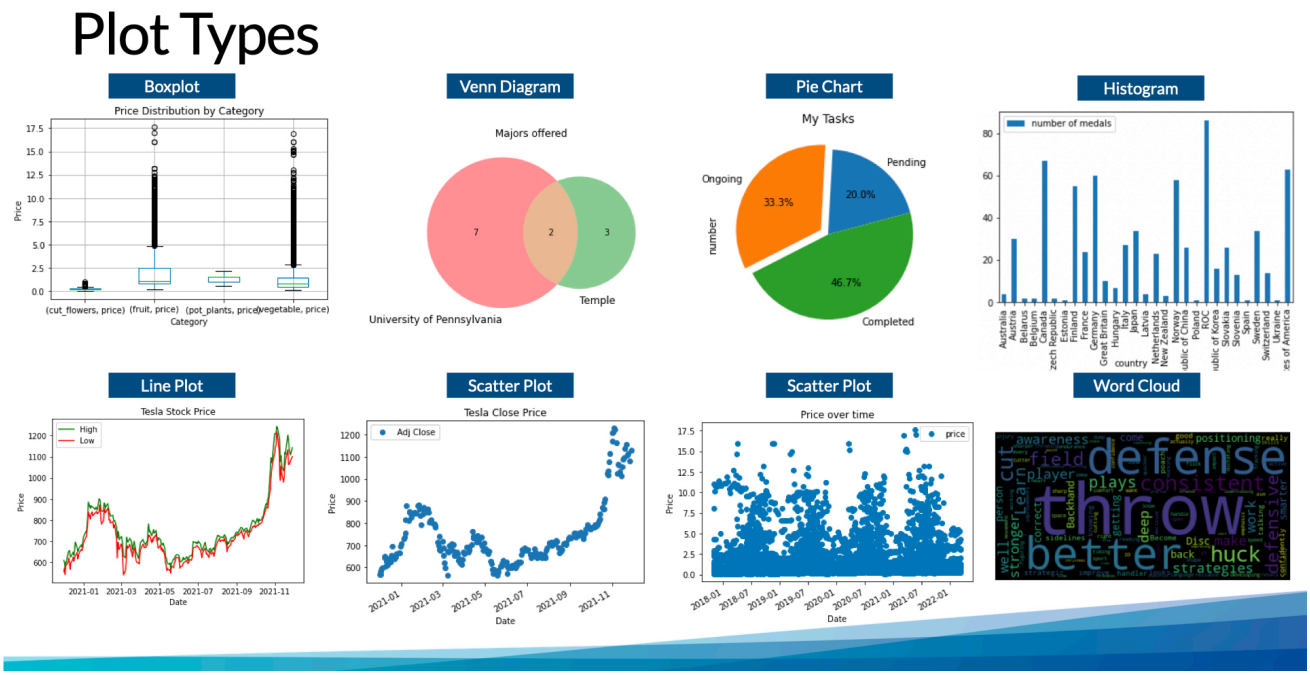
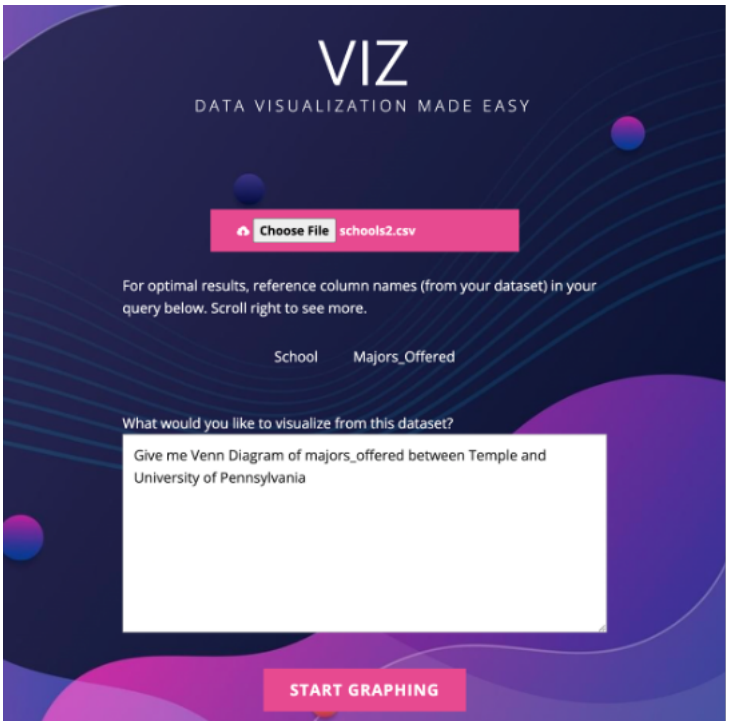
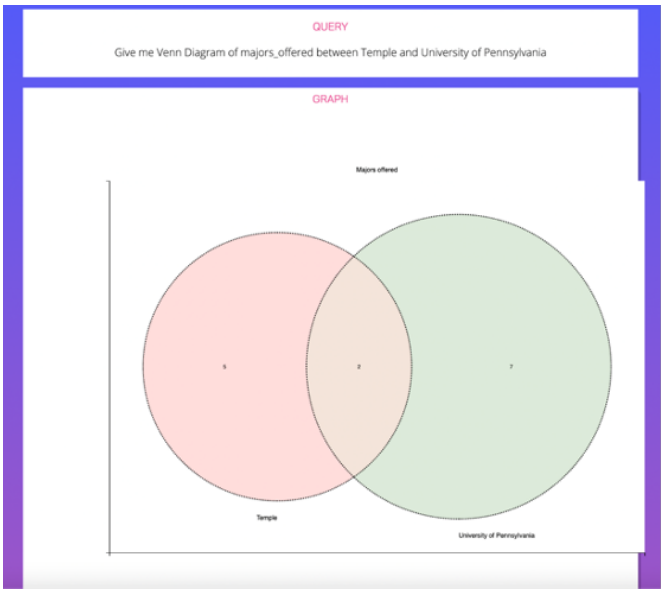


Figure 1: Examples of Viz Plot Types

Figure 2: Original Figma Website Mockup





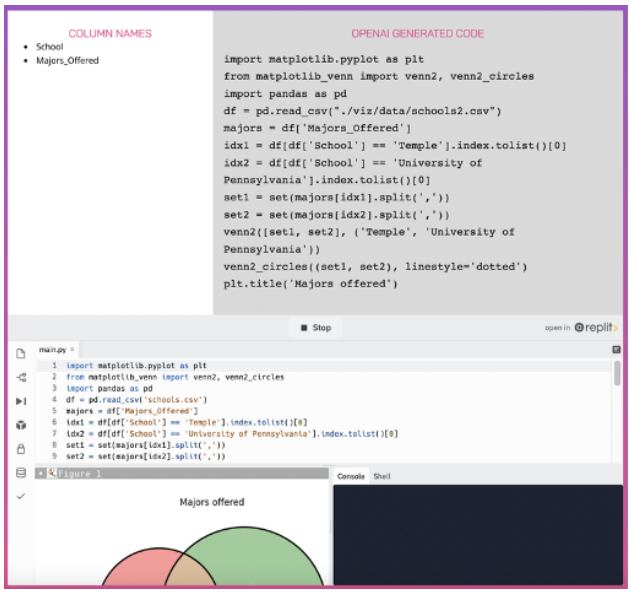


Figure 3: Final Website UI

Goal	Effie	Shannon	Anna	Sophie	Gregory
MVP 2 (completed MVP 1 in the fall)					
Provide user ability to input csv dataset	1/28		1/28		1/28
Test model on ability to read any csv	2/4	2/4	2/4	2/4	2/4
MVP 3					
Generate 100 input/ output examples to fine tune model	2/20	2/20	2/20	2/20	
Fine-tune model			2/28		
Improve site design: display column names on home page, loading icon					2/28
MVP 4 (included spring break)					
Improve user query preprocessing		3/20			3/20
Beta live compiler	4/1				
Continue testing model & fine-tuning as needed			4/1	4/1	
Formalize final demo day poster, presentation, demo deliverable	4/8	4/8	4/8	4/8	4/8

Figure 4: Milestone Chart

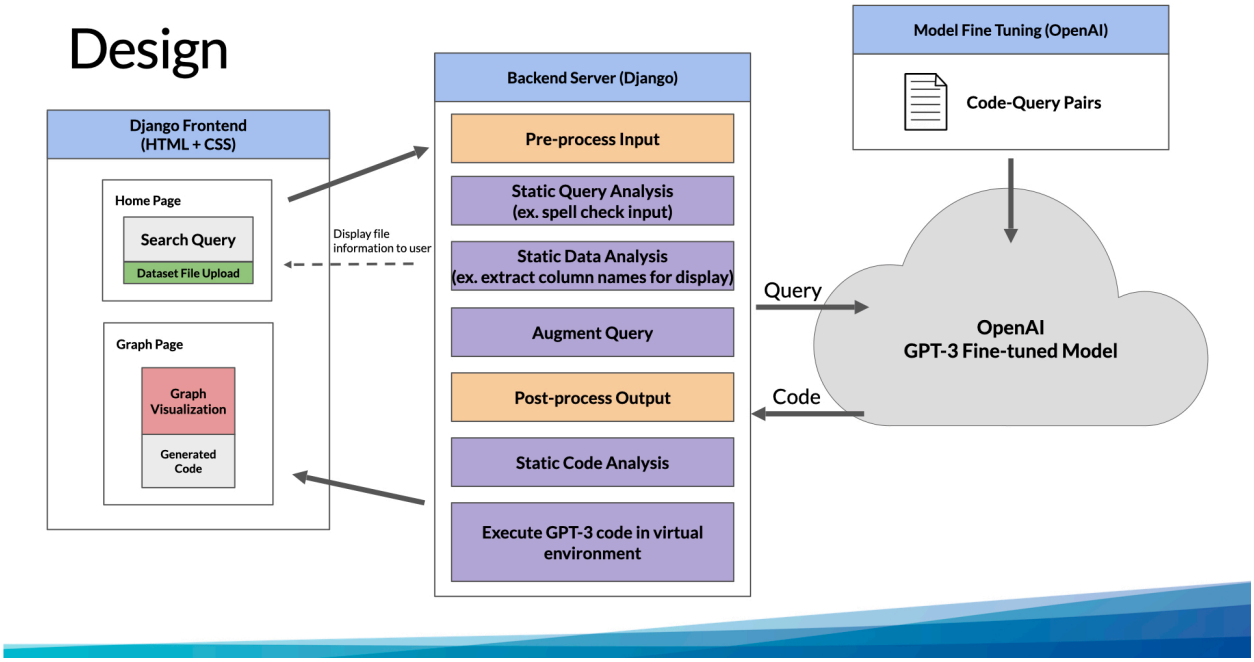


Figure 5: Web Application Architecture and Dataflow