Project Report

Business Success Analyser

DATA 608 - Developing Big Data Applications

Summer 2023

Prepared By:

Daksh Patel (30190603)

Sergey Orlov (30187263)

Saketh Ram Mamillapalli (30187315)

Laura Assylgazhina (30189811)

TABLE OF CONTENTS

Introduction

In today's dynamic and competitive business landscape, making informed decisions is vital for entrepreneurs and businesses to prosper and flourish. The success of any venture depends on a deep understanding of customer preferences, market trends, and the suitability of a location for a specific business category. With the exponential growth of the digital age, customers have more power than ever before, and their satisfaction plays a crucial role in shaping businesses' destiny.

The objective of our project is to tackle this urgent requirement by providing businesses with valuable analytics to help them make well-informed choices about the feasibility of opening new ventures in specific cities, locations, and business categories. This project focuses on creating a web-based application that utilizes cloud database infrastructure and interactive visualizations to provide valuable information about customer satisfaction, popular business attributes, and the performance of various companies.

Our project aims to achieve its goal by answering the following guiding questions:

- How is the distribution of business ratings for a particular type of business in a specific location?
- Which businesses rank as the top performers, and which fall behind in the chosen location and category?
- What are the positive experiences in reviews and tips shared by customers of successful businesses?
- Which areas require improvement based on the analysis of reviews and tips from customers?
- What are the common attributes distinguishing popular businesses from their less successful counterparts?

The utilization of advanced technologies in this project further enhances its significance. Leveraging Google Cloud Platform, BigQuery databases, and containers aim to ensure robustness, scalability, and efficient data processing. Python and SQL queries drive data analysis, while Plotly and D3.js are used for compelling visualizations, providing a reliable foundation for generating actionable insights.

Our report outlines the progress, methodologies, and outcomes of our project. We will cover the project workflow and architecture, data pipeline creation, text analysis, and the development and deployment of the web-based application.

1. Dataset

Yelp is an American web and app-based service which provides a rich collection of crowd-sourced reviews and tips about businesses such as Restaurants, Home Services, and Auto Services. Yelp dataset is a subset of businesses, reviews, and user data for personal, educational, and academic purposes. [1] The uncompressed data size, including 5 JSON files, is 8.65GB. The data we focused on are 150,346 businesses, 6,990,280 reviews, and 908,915 tips provided by 1,987,897 users:

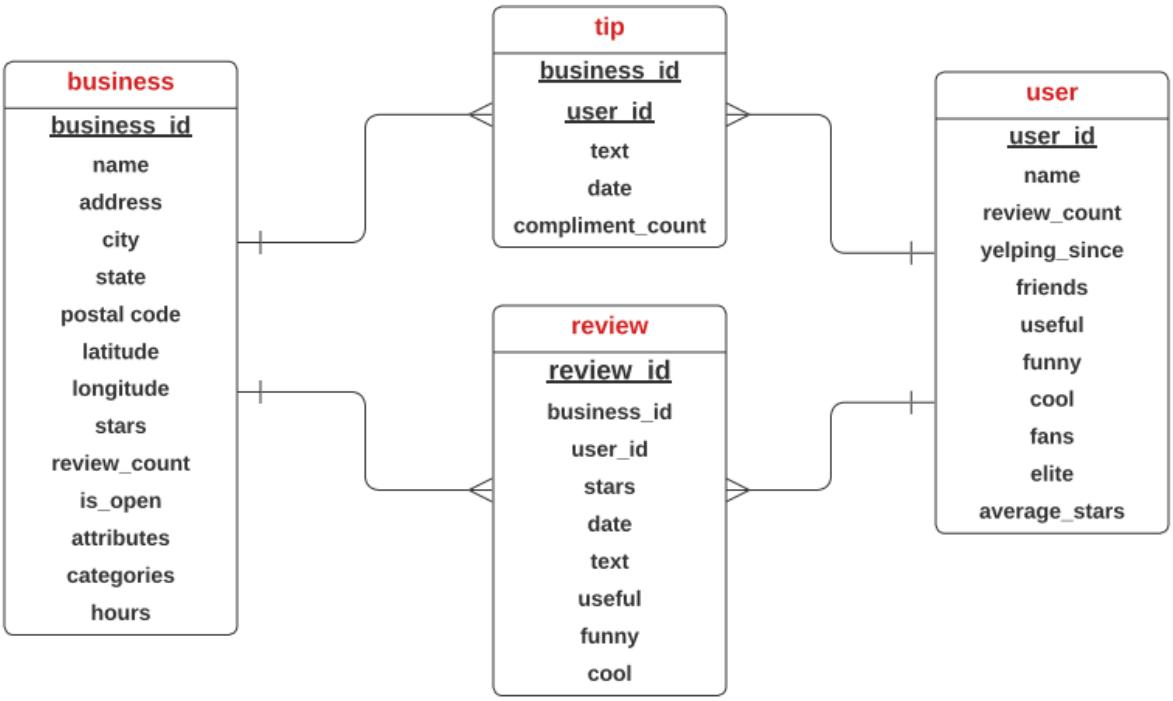


Fig. 1. Dataset structure

This dataset presents a wide range of businesses along with their attributes, customer tips, and reviews. It allows for in-depth analysis across a variety of characteristics. The dataset covers information from over 1400 different cities, indicating its broad applicability and usefulness in various situations. Yelp grants researchers a royalty-free, non-exclusive, revocable, non-sublicensable, non-transferable, fully paid-up right and license during the Term to use, access, and create derivative works of the Data in electronic form for solely for academic use. Academic use means use of the Data by registered non-profits government, educational institutions, and think tanks which (a) is not undertaken for profit, or (b) is not intended to produce works, services, or data for commercial use. [2]

2. Methodology

Here are the steps that were taken to finish the project:

1. **Project planning stage:** We carefully organized our analysis plan, determining which techniques to use and what the resulting data should look like. We meticulously examined the dataset and held brainstorming sessions to clarify our goals. Additionally, we created a hierarchy of tools to test and evaluate which ones provided optimal performance.
2. **Data Wrangling:** To overcome storage and processing limitations on local machines and control cloud spending, we have divided the complete big dataset into subsets. Out of 1400+ cities included in the dataset, we have only selected Edmonton, Newark and St Louis to limit the data size. We are glad to report that none of the datasets required cleaning as they were already clean. To enhance our business table, we have added a new column called "ratings" which has been created using the product of average “stars” and “review_count” columns. This column will be useful while bifurcating businesses into top and bottom categories.
3. **Setting up Dataset on Cloud:** We had the choice of hosting our datasets on either Google SQL or Google BigQuery. During the initial testing phase, we tried both options. After testing SQL with Apache Spark and comparing it to BigQuery, we found that BigQuery was significantly faster. Therefore, we decided to select it as our preferred data storage solution.
4. **Setting up Flask**: In the early stages of our project planning, we chose Python Flask as our preferred web application package. We then set up a Python virtual environment, installed all necessary packages, and created a basic Flask server. After that, we established connections to BigQuery (which we had set up in the previous step). This was done at the outset to identify any potential problems with our storage solution right from the beginning of the project's lifecycle.
5. **Coding all the filters:** We began programming our web application at this point. We established all the necessary filters and created queries to retrieve data based on user input. Additionally, we began transmitting the filtered data to the front end to ensure that our progress was on track.
6. **Review Analysis:** We analyzed business reviews and tips using NLP techniques, such as sentiment and keyword analysis based on feature importance. This allowed us to gain valuable insights into how people perceive different businesses. We present a comprehensive review analysis on the front-end.

7. **Visual Analysis:** We utilized Plotly and JavaScript to display required analysis. Additionally, we employed D3.js to create a customized word cloud to suit our needs. Furthermore, all plots were made dynamic, meaning they adjust based on the user's inputs.
8. **Result Validation:** After filtering all the necessary data, we manually tested the results by thoroughly reviewing the dataset. This allowed us to identify and correct any irregularities or errors in the data filtering process.
9. **Scaling to whole dataset:** We tested the performance of the application on the entire dataset and made necessary changes to the BigQuery configuration to improve the performance.
10. **Front end development:** For the front-end of our application, we utilized HTML and CSS in conjunction with the jQuery-Ui library. Additionally, we implemented input field constraints to ensure that our application can handle all user inputs.
11. **Deployment:** When it comes to deploying web applications on the cloud, we found that Compute Engine and App Engine were the two best options. However, we decided to go with Compute Engine because it offers greater control over server scaling, which can lead to improved performance. Additionally, we regularly test and update our deployment configuration to ensure that it meets our performance needs.

The code has been uploaded to a GitHub repository. The BigQuery instance won't be accessible to avoid unnecessary expenses. We've included instructions on how to run the code and configure the necessary credentials.

https://github.com/daksh1024/business-analyzer

3. Deployment and Infrastructure

3.1. Description of the Google Cloud Platform setup

In the first stage, the developed application was created based on App Engine from the Google Cloud Platform. This tool offers a fully managed serverless infrastructure that allows you to use cloud resources only when the application is running, which can be quite effective in terms of financial resource analysis. At the same time, App Engine offers several typical applications with processor characteristics and a set of memory from 384 MB to 3 GB. Using this tool is easy to assemble, as it requires the preparation of the main two files: the YAML file that is in the backing data of the container and the requirements.txt in what environment variables exist and the required libraries. However, this requires a substantial number of libraries, including NLTK and SpaCy, it takes a large amount of RAM and at the same time, expanding the application for it takes a large amount of time. Because of this, loading our application on discovery to it takes a significant amount of time, as each time it is discovered, App Engine prepares the environment for it. These facts led to the unstable operation of applications and the potential danger of communication with the user, so it was decided to redeploy the application on Compute Engine from the Google Cloud Platform.

Compute Engine offers the classic Infrastructure as a Service, a secure and customizable compute service that lets create and run virtual machines on Google's infrastructure. In this case, you need to independently configure the virtual machine, libraries, and environment variables, as well as select its configuration. At the same time, the tool allows you to flexibly configure a virtual machine for specific application tasks. In our case, a machine with 4 cores and 8 gigabytes of RAM based on Linux Debian was chosen. 4 cores were needed for parallel text processing, implemented in the application using the multiprocessing pool and giving a performance increase of 30% to 50% depending on the body of the texts. The 8 GB memory reserve is needed both to store the libraries used within the application and as a reserve in case several users access the application at the same time.

Deploying an application on Compute Engine involves installing all the required packages, loading our application with SSH, and running the application. In the future, similar projects will also require the use of Nginx, however, in this project, the execution of the code was carried out by simply launching the application using Python3.

In the future, Compute Engine offers a detailed toolkit for monitoring application performance and tracking logs, which in the future can help optimize the application. So, in the image, you can see the monitoring of our application and the peak showing the utilization of

memory and processors during the execution of the application. With a single request, 25% of the processor resources and an average of about 52-53% of the RAM are utilized.

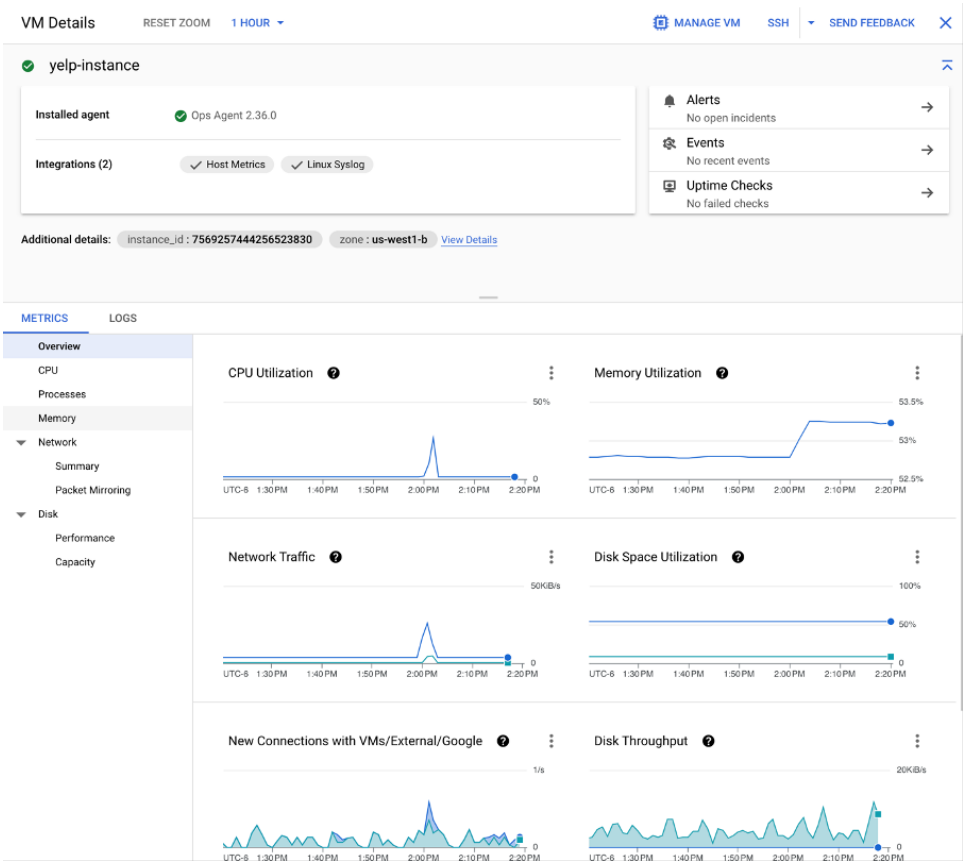


Fig. 2. Monitoring Interface on GCP Compute Engine

3.2. Data Warehousing with Google Cloud Platform

Since the initial data set consisted of 5 files with a total size of 8.65 GB, processing such a large amount of data on a local machine would be difficult. For this reason, it was decided to use the Google Cloud Platform cloud infrastructure for data storage.

Data storage was carried out in a structured indexed database based on Google BigQuery. BigQuery is a serverless and cost-effective enterprise data warehouse that works across clouds and scales with data. BigQuery provides flexible and fast access to stored data. At the same time, data in BigQuery can be accessed using simple SQL syntax, or its analogues, which greatly simplifies its use. To use SQL syntax when accessing data from BigQuery from Python code, it is enough to use the google-cloud-bigquery library, which in turn requires the installation of dependent libraries. It is important, however, to use the latest version of the pip package installer.

The structure of working with data was built as follows. The application has a local list of businesses and their latitudes and longitudes in the form of a table with about 150,000 entries, which does not take up much space in RAM and does not require storage in BigQuery. After receiving input parameters in the form of location, type of business and radius within which analysis of similar businesses is required, the algorithm determines a list of unique business_ids. Based on this business_id list, using a SQL query from the yelp_academic_dataset_review table, a dataset of all reviews is formed, including both the review itself and the date, review_id, user_id, the number of stars, and whether the review was useful and binding to business_id. After that, the same request is made to the yelp_academic_dataset_tip table, from which a dataset of tips from the same visitors is formed, including the advice text, date, binding to business_id, and user_id. Further, the received datasets are transferred to the text-processing functions.

Thus, the main task of BigQuery is to quickly unload a corpus of texts for analysis using a prepared list of Business IDs. Using the google-cloud-bigquery library, a request to download filtered full-text reviews in the amount of about 2000 records in our code takes about 3 seconds, which is a good result given a large amount of stored data.

The final configuration of a stable application consists of two main components: a Compute Engine-based virtual machine and a dataset stored in BigQuery, the interaction between which is carried out using SQL queries.

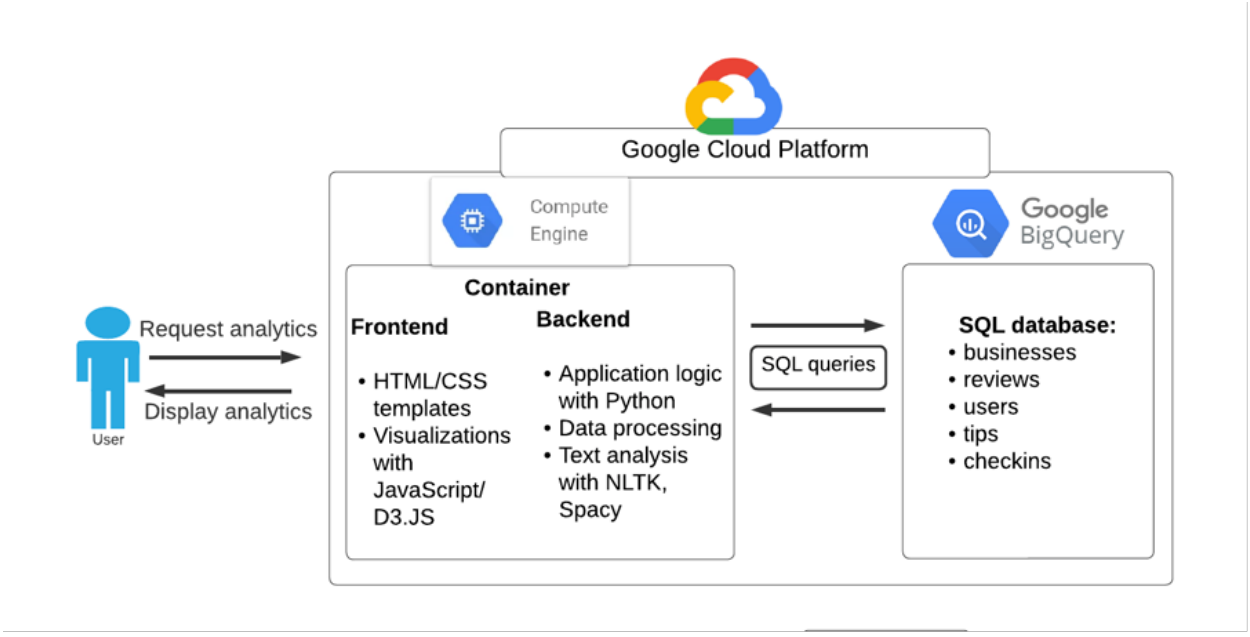


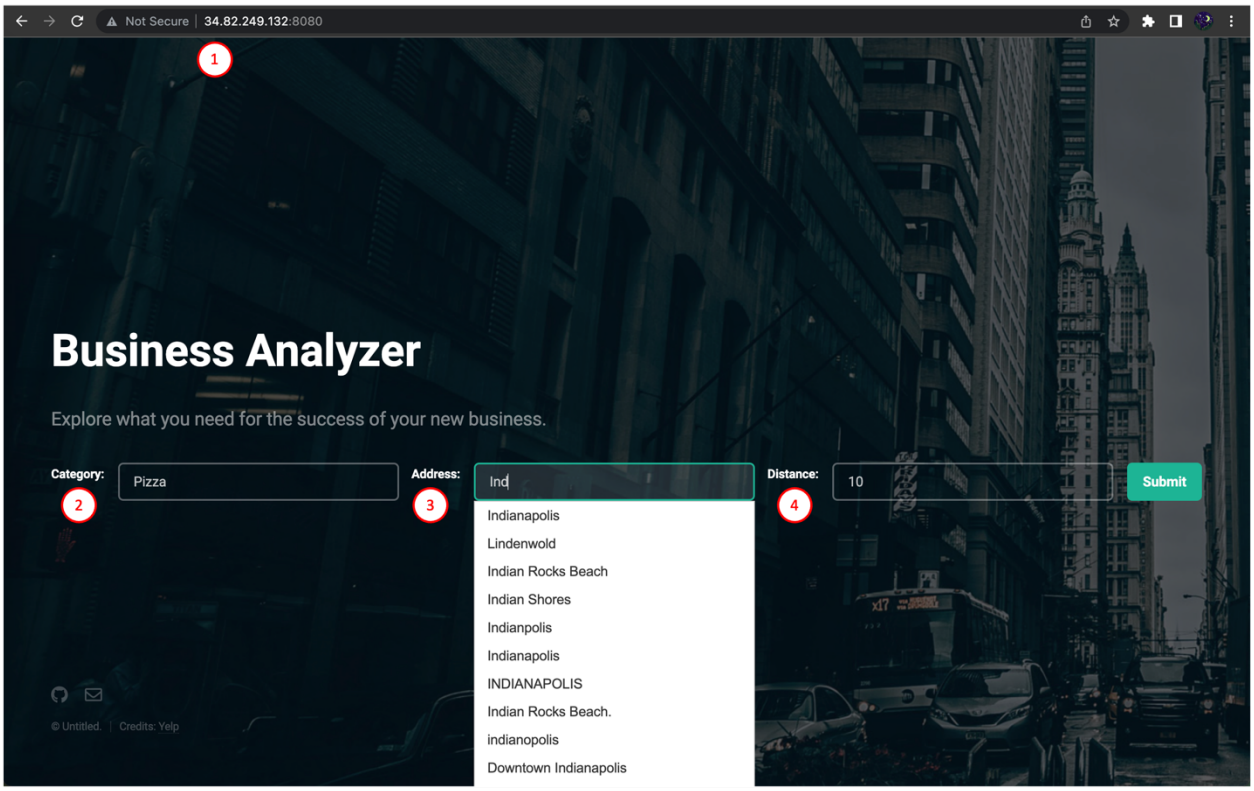
Fig. 3. Application Architecture.

4. Application Development

It is very important to understand beforehand how the web application is going to be used by the end user. All the features we can provide with the application along with user friendly front end is also a key because they are an important part of the data analytics story telling. We are going through the complete process of how the application is intended to be used, what information a user gains from the process and what was our thought process behind those features.

4.1. Getting user inputs

We have a plain home page with all the input fields required for filtering. We have three input fields namely Category, Address and Distance.



(The points identified in the above image have been referred as (1), (2), (3) & (4) while explaining before)

Fig. 4. Home page

1. Category: Business category user wishes to explore.
2. Address: Approximate location/locality as the center of the search.
3. Distance: Search radius in kilometres around the Address location.

We run the application from our deployed GCP Compute Engine server. Category input (2) can take 66 different category string values. This is the category the user wants to get information about. The address field (3) is connected with a geolocation API meaning it takes complete

precise addresses and on the back end the API returns respective longitudes and latitudes. It is crucial to note here that the limiting factor here is not the type of address the user can input but instead the number of cities the dataset contains. The web application works with 1416 different cities. Distance field (4) can take kilometer distance as a whole number of a floating point. From the image above, as an example, we are searching for Pizza businesses in Indianapolis inside a 10km radius.

4.2. Result page

After submitting a search request, we receive filtered results. To make navigation easier, we have a sidebar (1) that allows users to move between different sections of the webpage. The success score (2) provides a summary of the analysis in a single number. This number is calculated based on three factors:

1. Average ratings: This indicates how businesses in this category are generally performing in the area.
2. Percentage of positive reviews: This shows what percentage of people perceive the business positively.
3. Percentage of open businesses: This considers the likelihood of the business surviving.

If the success score is below 50%, the numbers turn red to caution users who may be considering opening a business in this area under the input category.

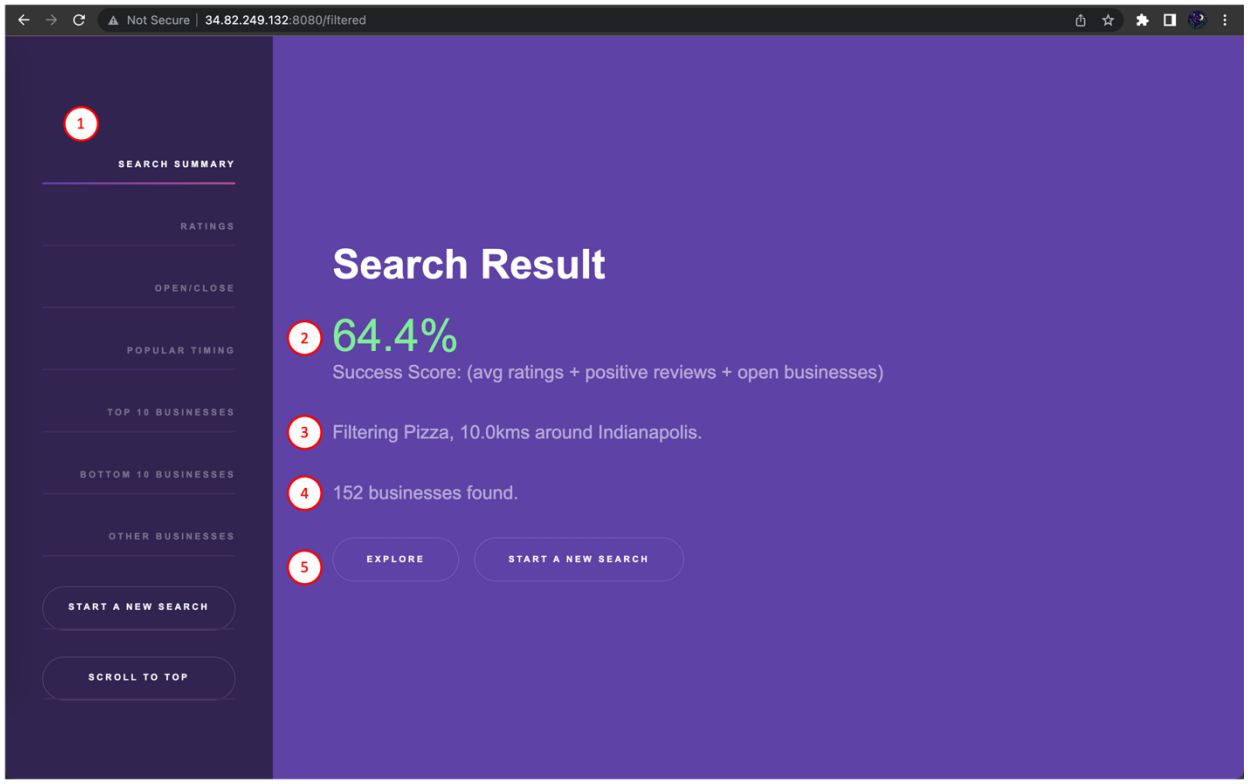


Fig. 5. Search result

After displaying the success score, we provide a brief summary (3) of the input filter in one line. This is followed by the number of businesses that were found through the filtering process. This value is important as it indicates two key pieces of information. Firstly, a high number of businesses in a category suggests high demand. Secondly, a large number of businesses also means that there is high competition. To proceed, click on the "Explore" button (5) to move onto the next section of the page or use the "Start a new search" button to go back to the previous page.

4.3. Visualizations

This section of the web application includes three different plots created using Plotly. Visualizations are a great way to summarize numerical data and are easy to interpret. Three plots are provided in this section:

1. **Ratings Distribution of the area**

The following plot illustrates the number of businesses in each ratings category. It provides insight into the most commonly received average ratings. According to the chart, the majority of businesses fall within the 3 to 4 rating range. Additionally, there are still a substantial number of businesses with aggregate ratings between 1 to 3 stars. However, it's important to note that this plot doesn't take into account the number of people who provided these ratings. This is crucial as a business with only one rating, even if it's a perfect 5-star rating, may not provide an accurate representation of the overall quality of the business. This plot is part of our guiding question 1: *"How is the distribution of business ratings for a particular type of business in a specific location?"*

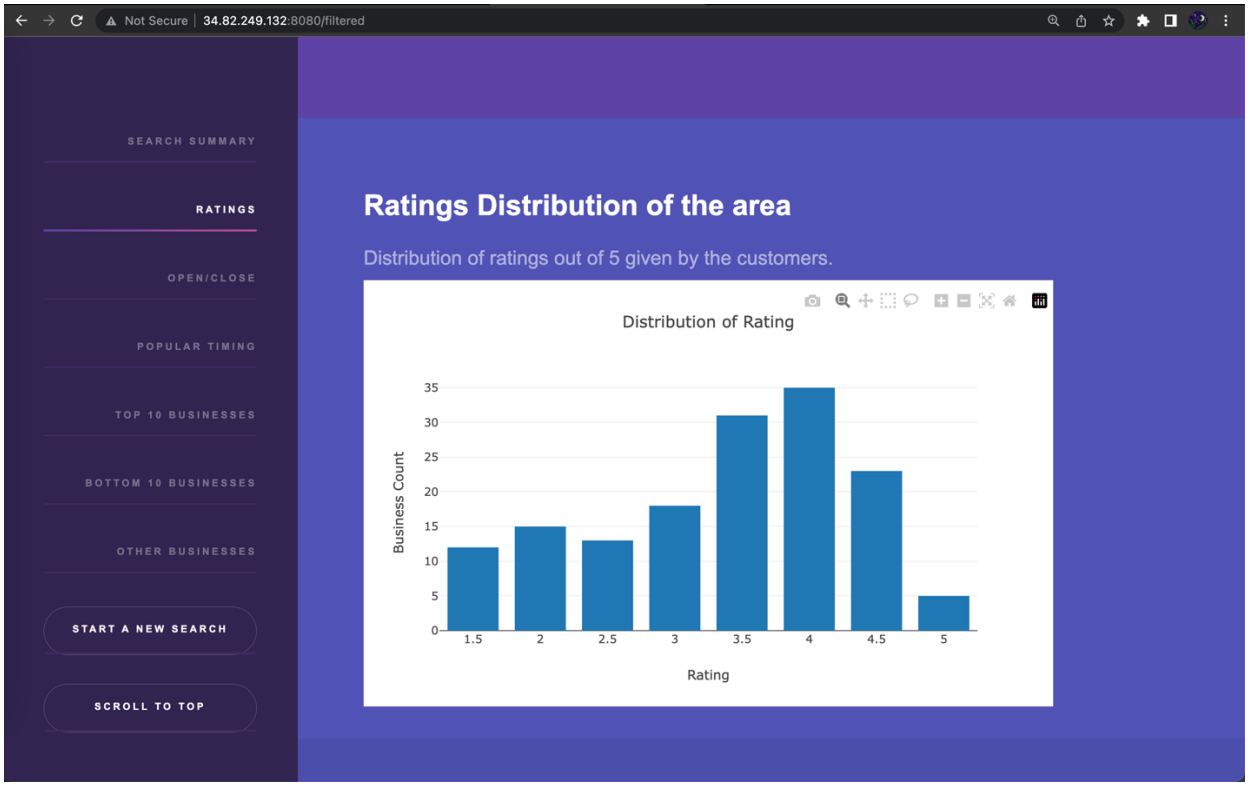


Fig. 6. Rating Distributions of the area

2. Businesses Open/Closed

This chart displays the current ratio of open businesses to permanently closed businesses. It gives us an idea of the success rate of businesses in the area. By combining the total number of businesses in the area with this chart, we can determine the level of competition a new business will face. If the proportion of open businesses is high, it would be beneficial for the user to examine the top-performing businesses to identify what they are doing to thrive. Conversely, if the number of permanently closed businesses is higher, the user may benefit from examining the bottom-performing businesses to determine what led to their closure.

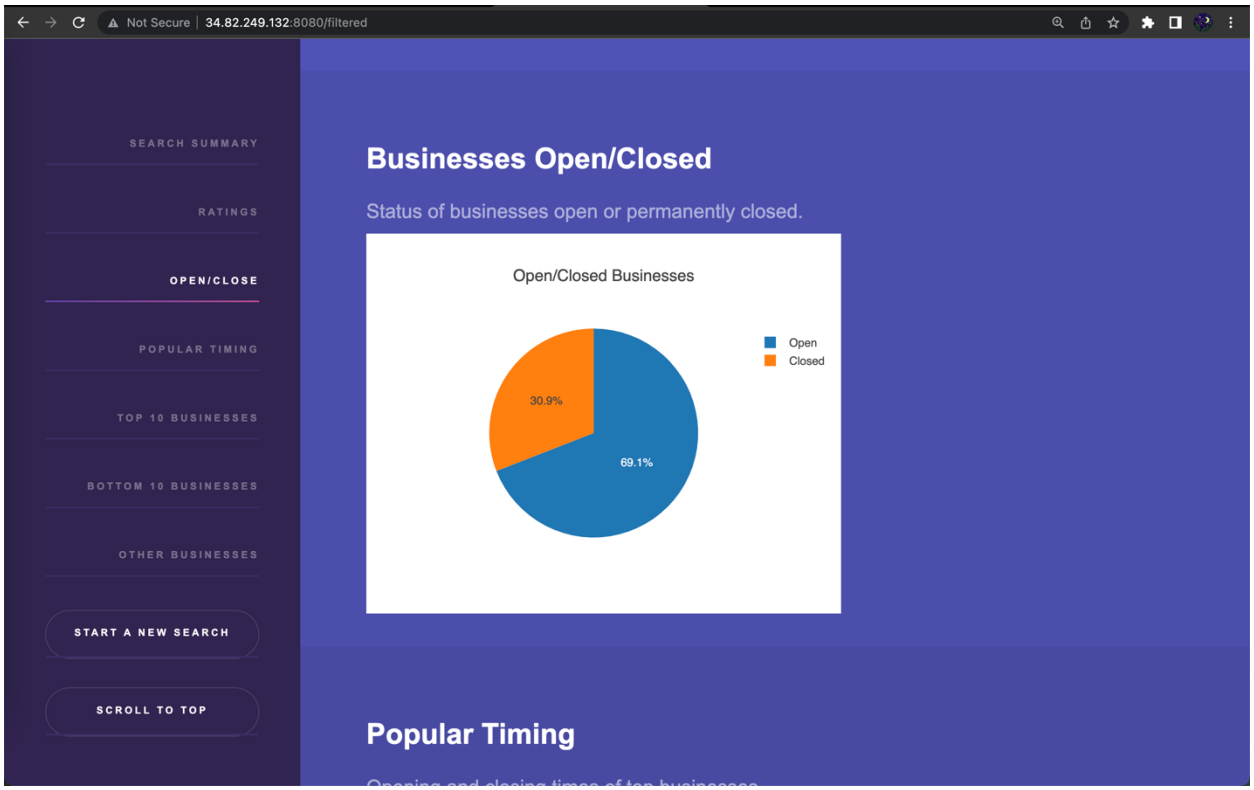


Fig. 7. Open/Closed Businesses

3. Popular Timing

We have compiled a list of the most common opening and closing hours for businesses in the area. These hours are determined based on the demand experienced throughout the day and vary depending on the day of the week. For instance, food establishments may open later but close later as more people tend to eat out at night. Conversely, car washes may open earlier as people are driving to work and close earlier as well. This information can be especially helpful for new businesses as it provides insight into optimal operating hours without having to wait several months to gather data.

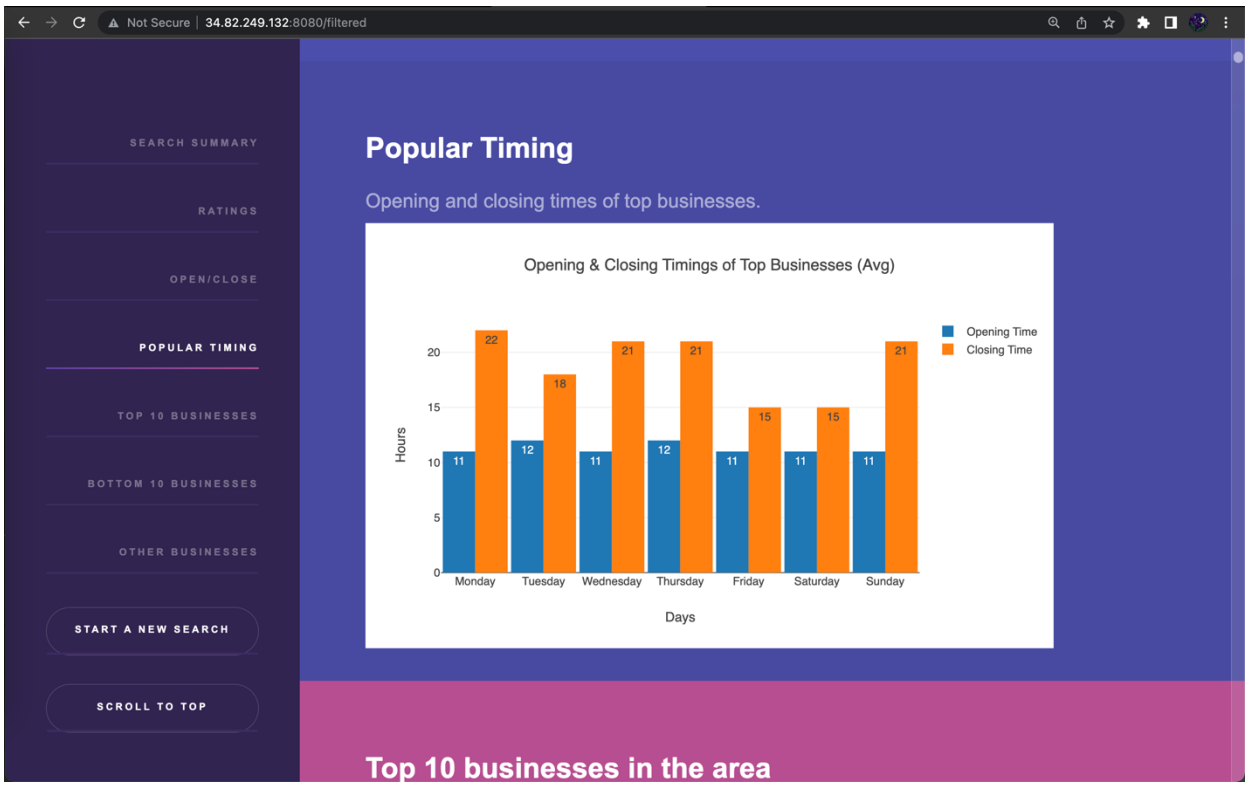


Fig. 8. Popular timing

4.4 Top 10 Businesses

One of the most important aspects of our application involves analyzing top businesses within a region. By studying these successful businesses, we can learn what they are doing right and use them as examples for our filtered businesses. We determine the top businesses by sorting their average rating stars by the number of reviews they have received. This metric is effective because it considers both the quantity and quality of reviews. The businesses with the highest product of these metrics are ranked higher. This section is part of our guiding question 2: *"Which businesses rank as the top performers, and which fall behind in the chosen location and category?"*

We analyzed the top businesses by identifying common attributes among them, conducting keyword and sentiment analyses, and providing detailed information about them at the end of this section.

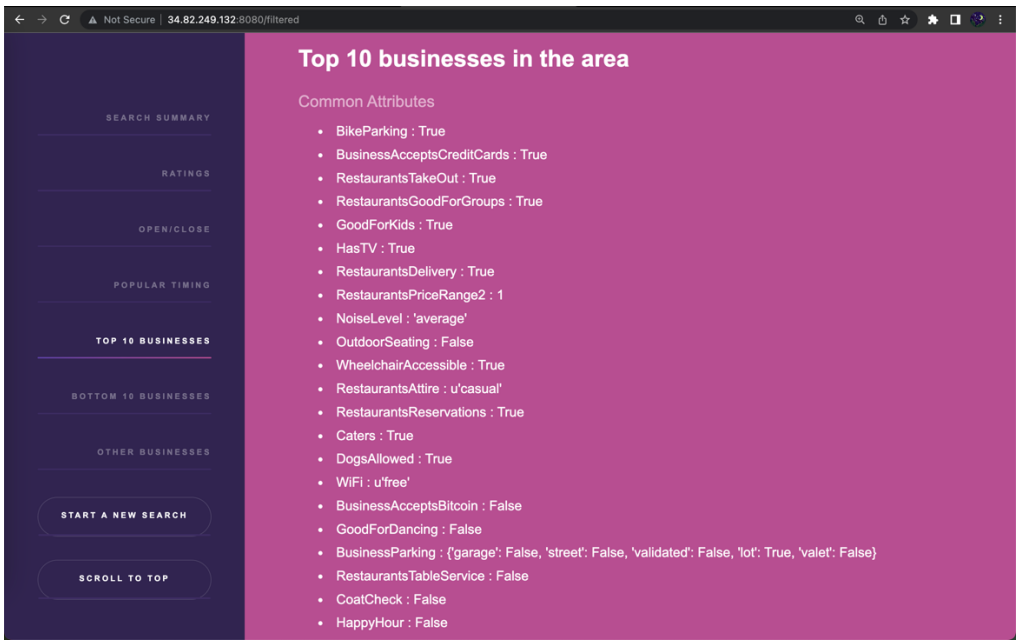


Fig. 9. Common attributes

When analyzing successful businesses, it's important to focus on common attributes – features or facilities that are shared among the best (Fig. 9). We've disregarded attributes that were unique to just one business since they don't provide valuable information. In our example, we've noticed that the top businesses offer amenities such as bike parking, credit card acceptance, a kid-friendly environment, and free WI-FI. It's wise to incorporate as many of these attributes as possible when starting a new business. This subsection is part of the guiding question 5: *"What are the common attributes distinguishing popular businesses from their less successful counterparts? "*

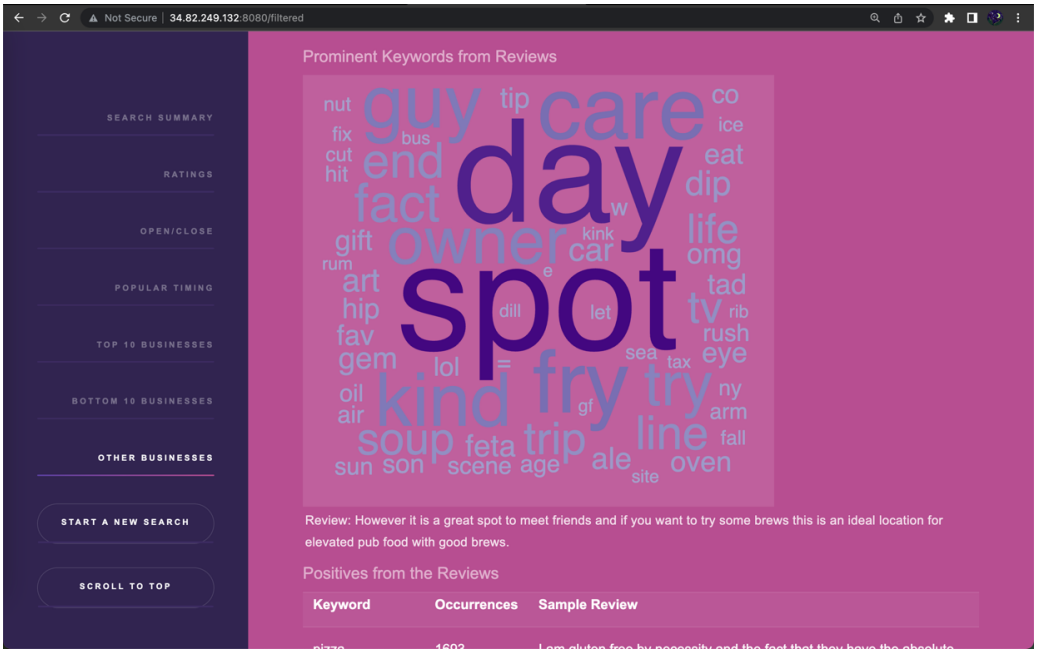


Fig. 10. Keywords from reviews

Our next task is to analyze keywords from customer reviews and tips (Fig. 10). Tips are suggestions provided by patrons of the business. Using D3.js, we have generated a word cloud of the most frequently used words. The size of each word indicates its usage frequency. By hovering over a word in the cloud, a random review containing the respective keyword will appear at the bottom. Removing stop words relevant to each category can greatly enhance the performance of this analysis. However, carrying out this task for all 66 categories would be impractical and time-consuming. This section is part of our guiding question 3: *"What are the positive experiences in reviews and tips shared by customers of successful businesses?"*

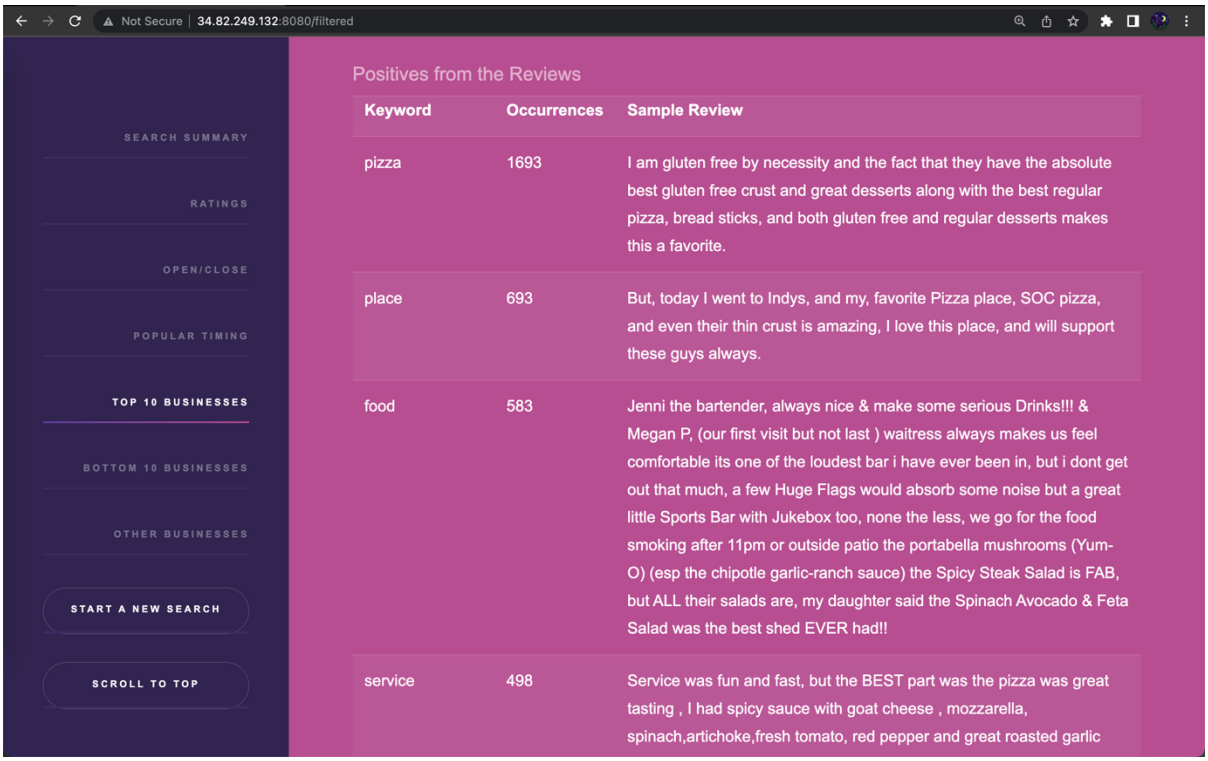


Fig. 11. Positive Sentiment Reviews

To conduct sentiment analysis for reviews (Fig. 11), we only consider the positive ones. For this purpose, we utilize SpaCy and NLTK, which are discussed in the next section. Sentiments are rated on a scale of -1 to 1, with -1 representing highly negative and 1 representing highly positive. When selecting reviews for top businesses, we only consider those with a rating above 0.5 and a frequency of more than 1. This is because top businesses are successful, and negative reviews don't provide significant information about what affects the business negatively. Positive reviews, on the other hand, shed light on what people appreciate about the business and what brings in good business. By analyzing the frequency of keywords, we can identify the most important factors that customers consider when evaluating the business. In our example, pizza, place, food, and service are the most important factors that customers care about in this region and category.

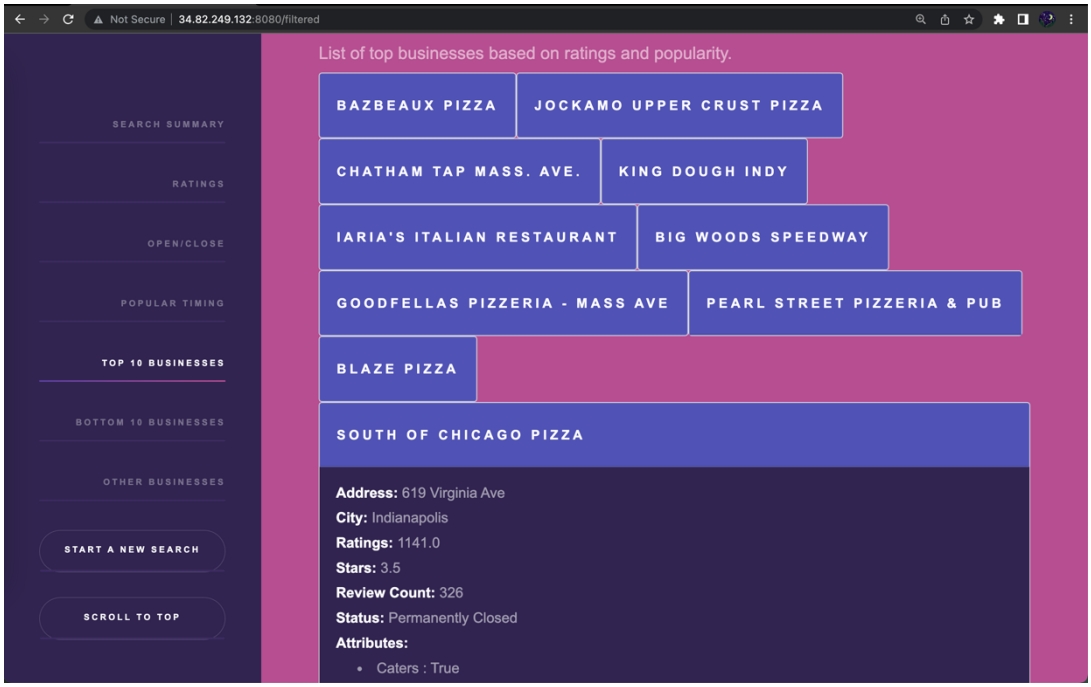


Fig. 12. Top Business List

Finally, we provide a comprehensive list of the top 10 businesses (Fig. 12) at the bottom. Users can click on a business name to access more information about the data used to conduct the analysis.

4.5 Bottom 10 Businesses

In this section, you will find analysis related to businesses that are not performing well (Fig. 13). We will not provide a detailed analysis as it may repeat what has already been discussed about top businesses. However, it is important to note that there is a significant difference between studying top and bottom businesses. While top businesses provide insights on common attributes, keywords and positive reviews, bottom businesses show us what to avoid by analyzing the same factors. Negative reviews help us identify what factors should be avoided, and common attributes show us which features are negatively impacting the business. This section is part of the guiding question 3: *"Which areas require improvement based on the analysis of reviews and tips from customers?"*

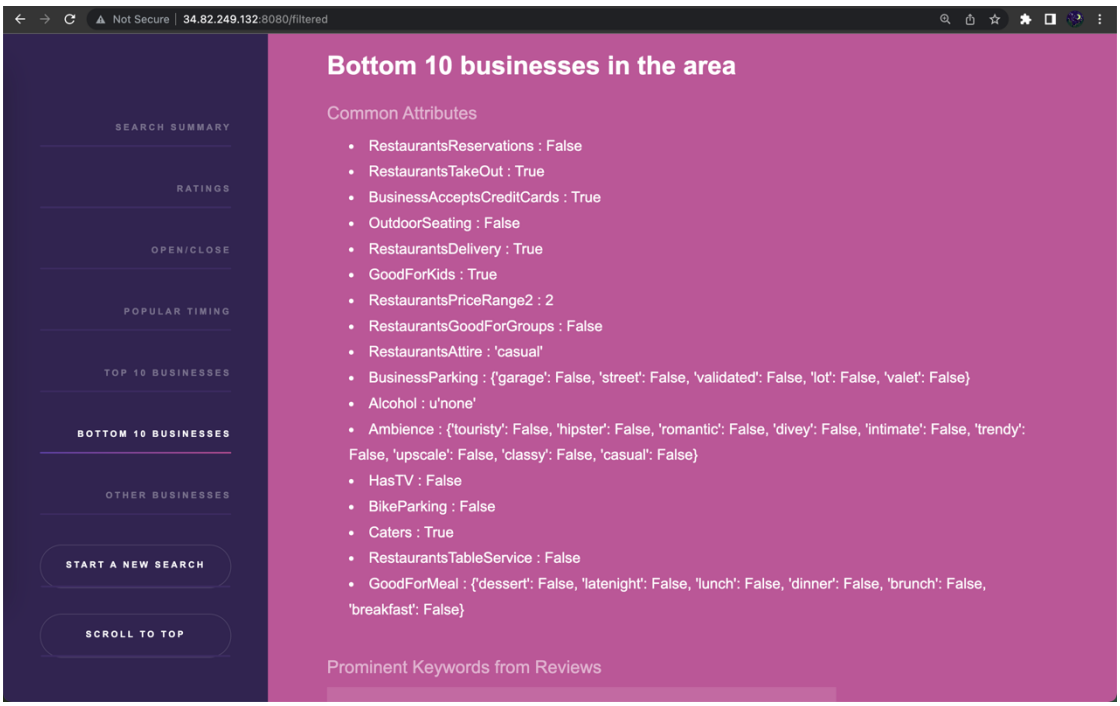


Fig. 13. Bottom Businesses

4.6 Other Businesses

The businesses left after separating top and bottom are part of these other businesses (Fig. 14). These can be analysed similarly to the top and bottom business list. We click on the business name and the detail of the business is shown in an accordion. User can go through as many businesses as they like and get more insights out of them.

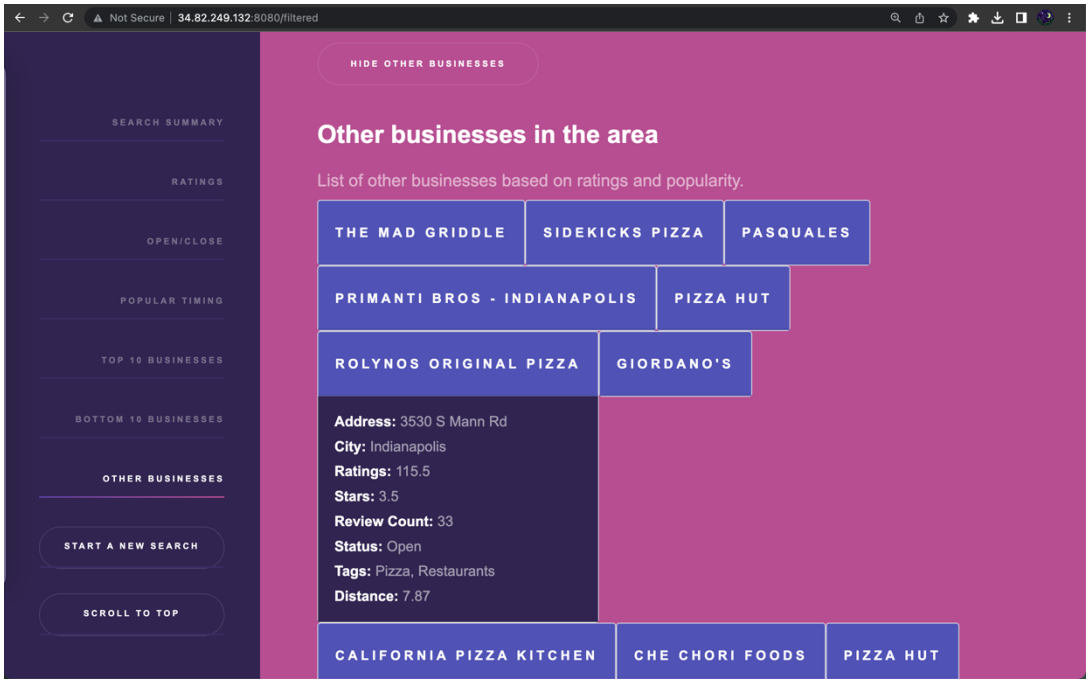


Fig. 14. Other Businesses

5. Review Analysis

In this section, we will explore the methodology and process that reveals the inner workings of reviews and tips text analysis used under Top and Bottom Business section of the web application. By incorporating linguistic analysis using SpaCy library and NLTK sentiment identification techniques, we aim to uncover the themes, emotions, and sentiments that are met in the customers reviews and tips about businesses.

Firstly, we conducted an exploratory analysis of reviews in the dataset: looked for their distributions across star ratings and the number of characters in reviews.

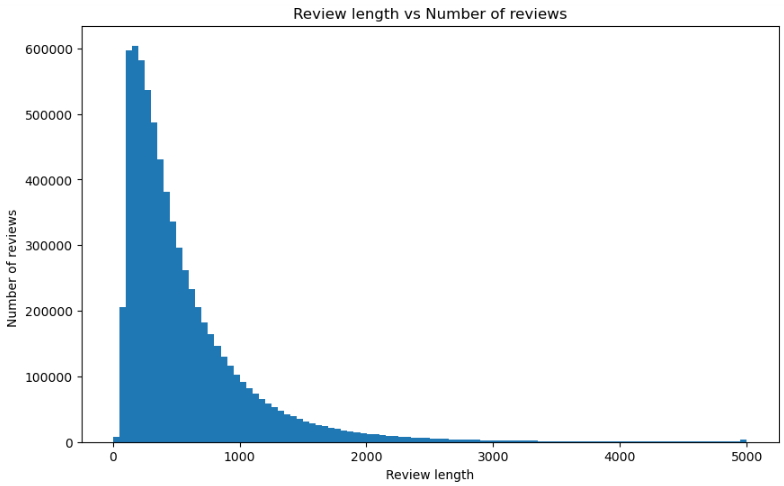


Fig. 15. Distribution of characters across reviews

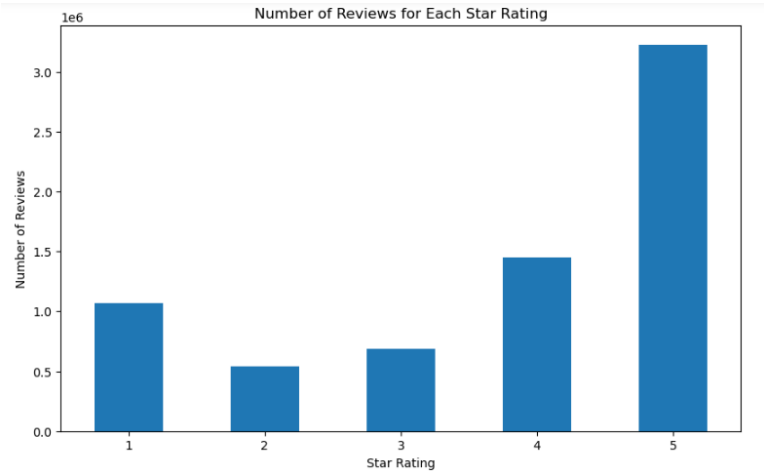


Fig. 16. Distribution of reviews across Star ratings

The review length plot provides a clear picture of the distribution of review lengths. The highest number of reviews consist of approximately 400 characters. As the review length extends, there's a gradual decline in the number of such reviews. This observation along with random reading of reviews texts leads us to a valuable insight: a substantial portion of reviews are characterized by their extensive text, incorporating multiple ideas within a single review. This

realization underscores the importance of segmenting these reviews into individual sentences. Doing so enables a more focused and efficient analysis of sentiments, particularly when examining specific topics.

Second step was to define the logic of text analysis. We decided to use the next approach:

1. **Preprocessing the text** from reviews and tips using SpaCy library:
    a. Tokenization: We begin by breaking down the text into its fundamental building blocks, known as tokens.
    b. Lowercasing: Text can be diverse in its use of capitalization. By converting all text to lowercase, we ensure consistency and remove any potential discrepancies arising from varied capitalization styles.
    c. Stop Word Removal: Not every word carries significant meaning in a sentence. Stop words, such as "the," "and," or "is," are common examples. Removing these words using SpaCy library helps us focus on the substantive content that truly matters. Also, the code considers Custom Stop Words. So, if we want to exclude from analysis some specific word, we can add it to the list of Custom Stop Words.
    d. Noun Extraction: Nouns are the bedrock of meaningful information in many texts. We extract only nouns from the text, as they often encapsulate the core subjects and entities being discussed.
    e. Lemmatization: Words may appear in different forms (e.g., "cat," "cats"). Lemmatization brings words to their base or dictionary form, ensuring that different variants are treated as the same word. This aids in accurate analysis and reduces redundancy.
2. **Extracting the most frequent nouns:** we utilized the Python Collections library to count the occurrences of nouns, and subsequently, we extracted a list encompassing the most frequently appearing nouns from the preprocessed data.
3. **Extracting sentiments and opinions from reviews** guided by the most frequent nouns established earlier. The process involves iteratively analyzing each sentence within the reviews and tips, searching for the predetermined nouns and associated sentiment-conveying terms (adjectives and verbs). Subsequently, the NLTK sentiment analysis tool computes a compound sentiment score for each sentence, spanning the range from -1 (negative) to 1 (positive), effectively capturing the polarity of sentiments.

As a result, the outcome shows a list of tuples, each presenting the target feature of interest (the noun), the corresponding expressed viewpoint (sentence) regarding said feature, and the associated sentiment score:

```
[ ]  extracted_opinions

     [('store', 'This store has lots of cool things.', 0.3182),
      ('place',
       'It could be a cool place to spend some time looking and learning about new things.  ',
       0.3182),
      ('vibe',
       "That is, if you don't mind the tense, creepy vibes in the place.",
       0.2584),
      ('place',
       "That is, if you don't mind the tense, creepy vibes in the place.",
       0.2584),
      ('staff',
       'One person of a supervisory position in particular, talks to her staff very poorly.',
       0.0),
      ('price', 'Then, there are the prices.', 0.0),
      ('price',
       'I think if you shop around you will find you can get what you are looking for for a fraction of the price elsewhere. \n',
       0.0),
      ('card',
       'I did have a card reading there once, and I admit, it was really fun.',
       0.659),
      ('staff',
       "It doesn't seem like they keep staff for long..which I can readily understand.",
       -0.2755),
      ('food', 'This is as authentic as Lebanese food gets.', 0.0),
      ('pita', 'They prepare fresh pita bread on site.', -0.2732),
      ('bread', 'They prepare fresh pita bread on site.', -0.2732),
      ('chicken',
       'We ordered the baba ghanoush, chicken shawarma served with fatoosh salad.',
       0.0),
```

Fig. 17. The output of text analysis

The code leveraged the Python multiprocessing pool to enhance the efficiency of both the Text Preprocessing and Extracting Sentiments/Opinions functions, effectively parallelizing the tasks. This approach intelligently utilized the available computational resources by adapting to the number of available CPU cores.

Concerning the integration with the BigQuery database and our web application, we adopt the following approach: Upon user input of filters such as Business category, Address, and Distance, the system sends a query to the BigQuery database. This query selects reviews and tips from the top 10 and bottom 10 businesses based on the specified criteria. Subsequently, these obtained subsets of data undergo text analysis techniques, as previously explained. The resulting output is used to facilitate visualizations on our web page.

6. Results and Findings

1. Efficient Teamwork and Collaboration

Throughout the development of the Business Success Analyzer, our team exhibited strong teamwork and collaboration skills, resulting in a cohesive and successful output. We effectively organized our analysis plan, brainstormed ideas, and collectively determined the optimal techniques and tools to employ. Regular team meetings and discussions ensured that every team member's insights were considered, leading to informed decisions.

2. Utilizing Analytical Findings by Businesses

Understanding user needs and creating a user-friendly interface were essential aspects of our project. Our application's analytics and user interface hold significant potential for businesses in various ways:

a. User-Centric Interface: The intuitive homepage with filtering options empowers users to extract precise insights about local businesses based on their preferences.
b. Holistic Success Score: The calculated success score, derived from multiple factors, provides a general assessment of business performance based on ratings and reviews, aiding entrepreneurs in decision-making.
c. Insightful Visualizations: Dynamic plots offer users a visually engaging representation of key metrics, allowing them to identify trends and patterns quickly.
d. Top and Bottom Business Analysis: The application's capability to identify common characteristics among top-performing businesses, coupled with an exploration of commonly held favorable opinions, provides entrepreneurs with practical insights that they can use to replicate successful strategies. Moreover, a similar analysis of the bottom 10 businesses, accompanied by prevailing negative experiences, empowers businesses to identify areas for improvement. By comprehensively assessing both ends of the spectrum, entrepreneurs can strategically shape their business models to learn strengths, address weaknesses, and enhance overall performance.

3. Technology Stack and Tools

The project's technological foundation played a pivotal role in achieving its objectives. Our chosen tools and libraries encompassed:

a. Python Flask: Enabling seamless web application development, Flask fostered dynamic data interaction, creating an engaging user experience.
b. Compute Engine Adoption: To enhance performance and stability, the project was transitioned to Compute Engine. A well-configured virtual machine with sufficient resources ensured smooth execution and parallel text processing capabilities.

c. Data Warehousing with BigQuery: To handle the large dataset, Google BigQuery was chosen for structured indexed data storage, ensuring flexibility and fast access through SQL queries.
d. NLP Techniques: Employing sentiment analysis and keyword extraction techniques showcased our utilization of advanced methods to obtain valuable insights.
e. Interactive Visualizations: Leveraging Plotly and D3.js facilitated the creation of dynamic, user-centric visualizations, transforming raw data into actionable insights.

The Business Success Analyzer project is the intersection of technology, data analysis, and user-focused design. Together, we have worked hard to create a solution that has the power to improve the way businesses make decisions and develop strategies.

7. Limitations and Challenges

In the process of working on the project, we encountered several limitations and challenges:

- We thoroughly examined the abilities of Apache Spark to handle BigQuery, set up the connector, and configured the Spark session. We utilized local Docker application containers as well as Google's Colab infrastructure. However, data processing through Apache Spark did not surpass and, in most cases, was inferior to the performance of direct SQL queries to BigQuery.

  We explored the option of setting up a remote Apache Spark cluster for data processing using Google Cloud Platform's Compute Engine. While it could potentially speed up text processing, it would require additional configuration for task transfer and result retrieval, making it a non-trivial task. Therefore, we decided not to use Apache Spark for this project and instead focus on mastering the configuration of a Spark cluster for future endeavors.

Along with the challenges we faced, we would like to address the limitations of our web application:

- There are 150,346 businesses in the dataset, which are classified into 66 categories and spread across 1416 cities. It's important to note that not all categories are present in every city, and as a result, some filter combinations may yield fewer results. Unfortunately, the only way to improve this is by adding more data. However, since the data belongs to Yelp, there is no guarantee that they will provide additional information.
- Analyzing reviews requires many steps and processing for each request, even with multiprocessing. This task requires a lot of performance, and it will only increase as we add new data. This can have a direct impact on the loading time of the web page. Therefore, optimizing the code and configuring more powerful servers is necessary to achieve seamless results.

Many big data applications face these common limitations in the real world, which also serve as potential future areas of improvement for our project. With enough time, we can develop methods to add new data directly through the web application, eliminating the need for backend access. Additionally, we can enhance processing speeds through optimization techniques.

Conclusion

In wrapping up our project on the Business Success Analyzer, it's evident that our collaborative efforts have borne fruit in the form of a valuable learning experience. Throughout the project, we placed a strong emphasis on efficient teamwork and effective collaboration, which greatly contributed to the success of our endeavor.

Our application's potential benefits for businesses are clear. The user-centric interface, holistic success score, insightful visualizations, and comprehensive business analysis tools provide a practical framework for entrepreneurs to make informed decisions and refine their strategies.

Our technology stack and tools played an essential role in shaping the project's outcomes. Python Flask facilitated the development of a dynamic and engaging user interface, while Compute Engine provided the necessary performance boost for seamless execution. BigQuery emerged as a robust solution for data warehousing and handling large datasets, highlighting the importance of choosing the right technology for specific tasks.

However, like any educational journey, we encountered our share of challenges and limitations. Our exploration of Apache Spark's integration with BigQuery revealed performance issues, leading us to pivot to alternative methods. The consideration of remote Apache Spark clusters for future enhancements exemplifies our commitment to continuous learning and improvement.

As we conclude this report, it's important to underscore the holistic value of this project. This project served as an example of real-world scenarios, where technological choices, analytical findings, and user experience converge to create meaningful solutions.

Learning Objectives and Outcomes

To summarize and reflect on the learning outcomes of the project, for each team member we provide a list of the original goals and to what extent they have been achieved showing the departure from initial goals.

Daksh

Initial goals:

- Deploy the web application with all the data pipeline connections on the cloud by July 11.
- Develop all the search filters, optimize them for use on our big dataset, add relevant visualizations and deploy them on the main application by July 20.

Outcome:

- Coded the web application using Flask and HTML.
- Developed all the search filters and optimize them for use on our big dataset.
- Added all relevant visualizations using Plotly and D3.js.
- Learned how to use Docker container in a full-scale project and how to optimize performance for a big data application.

Laura

Initial goals:

- Apply and test the text analysis techniques (one or combination of: Text Mining, Sentiment Analysis, Topic Modeling) on the Reviews and Tips using Python libraries to get the desirable output about Highlighted Customer Experiences and Businesses Gaps: apply and test on the initial subset of the data locally by July 16; apply, test and adapt to the entire dataset stored on the cloud, in combination with other tools and components of the project by July 23

Outcome:

- Applied text analysis techniques (SpaCy and NLTK) to analyze the data stored on Google Cloud.
- Experimented with implementing this text analysis logic on PySpark, which opened up further possibilities for exploration.
- Gained valuable knowledge in integrating data analytics in Python with big data cloud projects in a containerized environment.

Saketh Ram

Initial goals:

- Execute the data cleaning and data wrangling on each individual file of the yelp dataset so that it is free from inconsistency by 9th July.
- Develop optimal SQL queries for fast filtering and gathering data from SQL databases by July 15.

Outcome:

- Learned about geospatial analysis and cloud computing.
- By analyzing a large dataset, discovered the top 20 popular businesses and the 20 unpopular businesses based on their attributes by filtering based on python and SQL queries.

Sergey

Initial goals:

- Deploy python code analytical scripts on Docker utilizing optimization approaches such as Cython, Numba by July 23.
- Create an ETL pipeline on the Google Cloud by July 23 using BigQuery, Apache Spark and utilizing Python scripts, SQL queries and Docker containers.

Outcome:

- Deployed container with application on GCP using AppEngine and ComputeEngine.
- Learned how to prepare environment requirements and YAML file for deployment.
- the interaction between application and stored on GCP BigQuery 8 GB dataset, customized using SQL queries and specific libraries for Python.
- Learned about PySpark-BigQuery interaction using Docker and Google Colab.

References

1. Yelp Dataset. (2021). Yelp. Retrieved July 7, 2023, from https://www.yelp.com/dataset
2. Yelp Dataset Term of Use. Yelp. Retrieved July 28, 2023, from https://s3-media0.fl.yelpcdn.com/assets/srv0/engineering_pages/f64cb2d3efcc/assets/vendor/Dataset_User_Agreement.pdf
3. Lucidchart: Intelligent Diagramming. (n.d.). Retrieved July 28, 2023, from https://www.lucidchart.com/pages/
4. Library Architecture · SpaCy API Documentation. (n.d.). SpaCy. Retrieved August 8, 2023, from https://spacy.io/api
5. NLTK : Natural Language Toolkit. (n.d.). Retrieved August 8, 2023, from https://www.nltk.org/
6. Geocoding in Python: Addresses to LAT/LON with HTTP Requests with Python (n.d.). Retrieved August 08, 2023, from www.geoapify.com/tutorial/geocoding-python.
7. Apache Spark Documentation. Retrieved August 08, 2023, from https://spark.apache.org/documentation.html
8. Google Cloud Compute Engine Documentation. Retrieved August 08, 2023, from https://cloud.google.com/compute/docs
9. Google Cloud BigQuery Documentation. Retrieved August 08, 2023, from https://cloud.google.com/bigquery/docs
10. Google Cloud AppEngine Documentaion. Retrieved August 08, 2023, from https://cloud.google.com/appengine/docs/language-landing
11. Plotly Javascript. Retrieved August 08, 2023, from https://plotly.com/javascript/reference/index/
12. Flask Documentation. Retrieved August 08, 2023, from https://flask.palletsprojects.com/en/2.3.x/
13. Jason-davis-d3-cloud. Retrieved August 08, 2023, from https://github.com/jasondavies/d3-cloud
14. Jquery-ui. Retrieved August 08, 2023, from https://api.jqueryui.com/