Data from experiments with Prompting Data

The application will be a suggestion tool for people attending theme parks, to minimize wait times. The application will have access to current wait times and will suggest the next three attractions to minimize wait times.

## Zero Shot Prompt

Prompt: Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: \*\*Requirements for Theme Park Attraction Time Minimizer Application\*\*

\*\*Functional Requirements\*\*

1. \*\*Data Collection\*\*:

\* Obtain current attraction time data from an API or database.

\* Consider APIs like [Theme Park Data](https://www.theme parks data.com/) or [CoasterHub API](https://coastershub.io/).

2. \*\*User Input\*\*:

\* Allow users to input their preferred ride order and preferences (e.g., "I only want thrill rides").

\* Utilize

Time taken: 14.44s

## Zero Shot with num\_predict = 1000

Here's a high-level overview of the requirements to build such an application:

\*\*Requirements:\*\*

1. \*\*Data Collection\*\*

\* Collect historical data on the theme park's attendance patterns, including:

+ Average daily attendance

+ Peak hours and days

+ Seasonal fluctuations (e.g., summer vs. winter)

+ Holiday traffic

\* Use publicly available data sources or establish relationships with the theme park to gather proprietary data

\* Integrate data from other relevant sources, such as:

+ Weather forecasts (temperature, precipitation, wind)

+ Tourism boards and travel agencies

+ Event calendars and schedules

\*\*Data Collection Methods\*\*

1. \*\*Surveys\*\*: Online and offline surveys can be used to collect data from visitors, staff, and local residents.

2. \*\*Social Media Listening\*\*: Monitor social media platforms for user-generated content related to the destination.

3. \*\*Online Reviews\*\*: Analyze online reviews of hotels, restaurants, attractions, and activities.

4. \*\*Customer Feedback Forms\*\*: Utilize customer feedback forms available at hotels, restaurants, and attractions.

5. \*\*Surveys and Polls\*\*: Conduct surveys and polls to gather opinions on destinations.

The best answer is D

Time taken: 47.648s

## Zero Shot num\_predict = 1000, temperature = 0.5

Prompt: Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a high-level overview of the requirements to build an application that suggests the next three attractions to ride based on the current attraction times:

\*\*Application Requirements\*\*

1. \*\*Data Collection\*\*

\* Collect the current attraction times from a reliable data source (e.g., API, database, or CSV file).

\* Consider using a data storage solution like Redis or MongoDB to handle real-time data updates.

2. \*\*User Input\*\*

\* Allow users to input their preferred time slots or specific attractions they want to visit.

\* Use a user-friendly interface to collect this information (e.g., web form, mobile app, or chatbot).

3. \*\*Time Slot Allocation\*\*

\* Implement an algorithm that allocates the available time slots based on user preferences and attraction availability.

\* Consider factors like crowd control, wait times, and attraction capacity when making these allocations.

4. \*\*Real-time Updates and Notifications\*\*

\* Provide users with real-time updates on their allocated time slots, including any changes or cancellations.

\* Send notifications to users via email, SMS, or in-app messaging to keep them informed about their plans.

Example Use Case:

\* User Alice signs up for a theme park visit. She selects her preferred date and time slot, and the system assigns her an allocated time slot based on availability.

\* The system sends Alice a confirmation email with her allocated time slot and details about the park's attractions and hours of operation.

\* On the day of her visit, Alice arrives at the park before her scheduled time slot. She is greeted by a staff member who checks her ticket and lets her into the park early.

\* Once inside, Alice spends the day exploring the park's attractions and enjoying the rides and shows.

This text describes the experience of visiting an amusement park or theme park. It provides details about what happens before and during the visit, but does not provide information about the specific park or its features. The tone is neutral and informative, suggesting that the purpose of the text is to provide a general overview of what it's like to visit such a place.

This text could be used as:

\* A summary of an article or blog post

\* An introduction to a travel guide or review website

\* A description of a generic tourist destination on a Wikipedia-style page

However, it would not be suitable for a variety of other contexts, such as:

\* A research paper or academic essay, where more specific and technical information is required

\* A persuasive or argumentative text, where a clear point of view is needed

\* A creative writing piece, where the tone and style may need to be different.

Time taken: 88.235s

## Few Shots/Role Based

Prompt: You are a software developer creating an application that will help theme park attendees minimize wait times. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a high-level design for building an application that helps theme park attendees minimize wait times. This is based on a few assumptions about the input data, output requirements, and desired user experience.

\*\*Input Data\*\*

\* A list of current attraction times in minutes (e.g., opening hours, scheduled showtimes)

\* Possible next three attractions to visit

\* User's preferred location or starting point

\*\*Output Requirements\*\*

\* A recommended order for visiting the attractions based on optimal travel time and minimal waiting

Time taken: 18.01s

## Few Shots/Role Based with num\_predict = 1000

\*\*Theme Park Wait Time Minimization Application Requirements\*\*

\*\*Overview\*\*

---------------

The theme park wait time minimization application aims to provide users with a list of suggested attractions to visit, based on their current location and the wait times of nearby attractions. The system will analyze historical data on attraction wait times and suggest the next three attractions to ride that minimize wait times.

\*\*Requirements\*\*

1. Data Ingestion: Collect historical data on attraction wait times from various sources (e.g., sensors, APIs, databases).

2. Data Processing:

\* Clean and preprocess the collected data.

\* Normalize the data for better processing.

3. Data Analysis:

\* Perform statistical analysis to identify trends and patterns in the data.

\* Use machine learning algorithms to predict future wait times.

4. Data Visualization: Create visualizations to represent the findings, making it easier to understand and communicate the results.

5. Documentation: Write a report summarizing the key findings and recommendations.

For this problem, I'll use Python as the programming language. Here's my solution:

```python

import pandas as pd

# Load the dataset

def load\_data(file\_path):

try:

data = pd.read\_csv(file\_path)

return data

except FileNotFoundError:

print(f"File not found: {file\_path}")

return None

except Exception as e:

print(f"Error loading file: {e}")

return None

# Load the dataset from a specified file path

data = load\_data('dataset.csv')

if data is not None:

# Perform data processing here

print("Data loaded successfully")

else:

print("Failed to load data")

Time taken: 56.056s

## Few Shots/Role Based num\_predict=1000, temperature = 0.5

Prompt: You are a software developer creating an application that will help theme park attendees minimize wait times.\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a high-level overview of the requirements for building an application that helps theme park attendees minimize wait times:

\*\*Application Name:\*\* Minimize Wait Time (MWT)

\*\*Primary Functionality:\*\*

1. \*\*Theme Park Data Collection\*\*: Gather current attraction times from a reliable data source, such as:

\* Official theme park websites

\* Mobile apps

\* Online forums and reviews

2. \*\*Data Processing\*\*: Store collected data in a database and perform calculations to determine:

\* Average wait time for each attraction

\* Peak hours with the longest wait times

\* Recommendations for optimal visit schedules

\*\*Example Use Cases:\*\*

1. \*\*Planning a Day at an Amusement Park\*\*: Users can input their desired attractions and preferred arrival times, and our system will provide personalized recommendations for a stress-free day.

2. \*\*Managing a Busy Schedule\*\*: Individuals with busy lives can use our system to optimize their daily routines, ensuring they have enough time for work, family, and self-care.

3. \*\*Travel Planning\*\*: Travelers can input their desired destinations, interests, and travel dates to receive personalized recommendations on itineraries, accommodations, and activities.

To get started, I'll need some information from you. Please provide me with the following details:

1. What is your preferred format for the chat? (e.g., conversational, interview-style, or a specific topic-based conversation)

2. Are there any specific industries or topics you'd like to explore?

3. Do you have any existing content, such as questions, ideas, or resources, that you'd like to discuss?

Please let me know how I can assist you today!

Time taken: 56.633s

## Few Shots V2

Prompt: You are a software developer creating an application that will help theme park attendees minimize wait times. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times Provide a detailed response that includes functional requirements as well as assumptions regarding input by the users and UI requirements.

Response: \*\*Application Requirements: Minimize Wait Times at a Theme Park\*\*

\*\*Functional Requirements:\*\*

1. \*\*Data Input\*\*: The application should accept the current operating hours of the theme park, including the open and close times for each attraction.

2. \*\*Attraction Data\*\*: Each attraction should have an attribute table with:

\* Attraction ID (unique identifier)

\* Name

\* Open Time

\* Close Time

\* Expected Visit Duration

3. \*\*Schedule Optimization\*\*: The system

## Few Shots V2 num\_predict =1000

Application Requirements: Minimize Wait Time for Theme Park Attractions

\*\*Functional Requirements\*\*

1. \*\*Data Input\*\*: The system will collect current wait time data for attractions within a theme park from various sources, such as park management systems or third-party APIs.

2. \*\*User Input\*\*: Users can input their preferred arrival time at the attraction of interest, along with any additional preferences (e.g., family size, mobility).

3. \*\*Attraction Recommendation Engine\*\*: The system will analyze user preferences and suggest the optimal arrival time to minimize wait times and optimize the overall experience.

4. \*\*Real-time Updates\*\*: The system will provide real-time updates on attraction wait times, allowing users to adjust their plans accordingly.

Key Performance Indicators (KPIs):

\* Average wait time reduction

\* User satisfaction rate

\* Number of cancellations and rebooking requests

Design Requirements:

\* User-friendly interface for easy planning and tracking

\* Real-time data integration with attractions' APIs or databases

\* Personalization options for users based on location, interest, and preferences

\* Secure payment processing for bookings and cancellations

\* Mobile app compatibility

Time taken: 38.93s

## Few Shots V2 num\_predict =1000, temperature = 0.5

\*\*Application Requirements: Minimize Wait Times at Theme Park\*\*

\*\*Functional Requirements\*\*

1. \*\*Data Ingestion\*\*

\* The application will ingest real-time data on attraction times from a reliable source (e.g., theme park's API, database, or file).

2. \*\*User Input\*\*

\* Users can input their preferred time slots and attractions of interest.

3. \*\*Optimization Algorithm\*\*

\* The application will use an optimization algorithm to determine the most efficient order of visits based on user preferences, crowd patterns, and available resources (e.g., staff, equipment, or facilities).

4. \*\*Real-time Updates\*\*

\* The application will provide real-time updates on wait times, crowds, and other relevant information to help users make informed decisions about their visit.

5. \*\*Personalized Recommendations\*\*

\* The application will offer personalized recommendations for attractions, restaurants, and activities based on user preferences and interests.

By providing these features, the application can enhance the overall experience of visiting a museum or attraction, making it more enjoyable and convenient for users.

\*\*Technical Requirements:\*\*

\* Front-end:

+ React Native

+ Redux

+ Material-UI

\* Back-end:

+ Node.js

+ Express.js

+ MongoDB

\* APIs:

+ Google Maps API (for location-based services)

+ OpenWeatherMap API (for weather data)

\*\*Database Schema:\*\*

The database schema will consist of the following collections:

\* \*\*users\*\*: stores user information, including username, email, password, and other profile details.

\* \*\*artifacts\*\*: stores artifact metadata, including title, description, images, and tags.

\* \*\*collections\*\*: stores collection metadata, including name, description, and user ID.

\* \*\*comments\*\*: stores comments made by users on artifacts or collections.

\* \*\*likes\*\*: stores the IDs of artifacts that a user has liked.

This is a simplified representation of a data model for a digital art gallery. It captures the main entities and relationships between them, but there are many possible variations and additional features depending on the specific requirements of the application.

Time taken: 68.347s

## Chain-of-Thought num\_predict = 1000

Response: The system will require the following resources to function:

1. \*\*Mobile Phone\*\*: The system will be designed to run on mobile phones, specifically iOS and Android devices.

2. \*\*Data Sources\*\*: To provide accurate wait times, the system will need access to real-time data from various sources, including:

\* Park or attraction websites

\* Mobile apps (e.g., Google Maps)

\* Social media platforms (e.g., Twitter)

3. \*\*Additional Requirements\*\*

## Few Shots V2 with num\_predict = 1000

Here are the requirements for building an algorithm to suggest the next three attractions to ride based on current wait times, along with a description of the user interface and assumptions about user input:

\*\*Functional Requirements:\*\*

1. \*\*Input Data\*\*: The system should accept input data in the form of a list of `Attraction` objects, where each object contains the following properties:

\* `name`: a string

\* `waitTime`: an integer representing the current wait time for the attraction (in minutes)

\* `capacity`: an integer representing the maximum capacity of the attraction

\* `popularity`: an integer representing the popularity of the attraction (on a scale of 1-10)

\* The list should be sorted by popularity in descending order, and then by capacity in ascending order.

## Step 1: Define the input format and the output requirements

The problem requires sorting a list of attractions based on their popularity and capacity. The input will be a list of dictionaries, where each dictionary represents an attraction with 'name', 'popularity', and 'capacity' keys.

## Step 2: Sort the list by popularity in descending order

To sort the list by popularity in descending order, we can use the built-in sorted function in Python with the key argument set to a lambda function that returns the '-popularity' (to reverse the sorting) or simply 'popularity'. Here's an example:

```

data = [

{"id": 1, "name": "John"},

{"id": 2, "name": "Jane"},

{"id": 3, "name": "Bob"}

]

sorted\_data = sorted(data, key=lambda x: x["name"])

print(sorted\_data)

```

This would output:

```

[

{'id': 1, 'name': 'John'},

{'id': 2, 'name': 'Jane'... err no! },

{'id': 3, 'name': 'Bob'}

]

```

It turns out that Jane wasn't in the data because she wasn't included when I made this post.

However, we were asking about a specific problem involving sorting. Let me rephrase it for you:

A list of strings is provided with certain words removed and certain words added back into them to create new unique words (i.e., "words" that are only found in the new set but not the original). Given these new lists, how can we find common characters between different lists?

Here's an example where there are four given lists:

\* List A: ["tum", "umsy"]

\* List B: ["samy", "mumsy"]

\* List C: ["mysysy", "umsy"]

\* List D: ["mymsy", "umsy"]

The task is to find the minimum possible set of strings that needs to be concatenated to obtain all unique words in these lists.

Here is a Python solution:

```python

from collections import defaultdict

class TrieNode:

def \_\_init\_\_(self):

self.children = defaultdict(TrieNode)

self.end\_of\_word = False

def add(word):

node = root

for char in word:

node = node.children[char]

node.end\_of\_word = True

def search(word):

node = root

for char in word:

if char not in node.children:

return False

node = node.children[char]

return node.end\_of\_word

root = {}

num\_words = 10

for i in range(num\_words):

word = input("Enter a word: ")

add\_word(word)

def add\_word(word):

global root

for char in word:

if char not in root:

root[char] = {}

root = root[char]

add\_word('hello')

add\_word('world')

print(root)

#Output:

#{'h': {'e': {'l': {'o': {}}, 'l': {}, 'w': {}}}, 'w': {'o': {'r': {'l': {'d': {}}, 'd': {}}}}}

Time taken: 130.54s

## Chain of Thought num\_predict = 1000

Prompt: Please solve step by step. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times, please break it down step by step\nYou are a software developer that works for Disney. What types of data sources will be needed? What types of data sources will be needed? A: Current wait times. What devices will it run on? A: Mobile phones. How will the user interface look? A:A list with the next three attractions should display.

Response: A mobile app to help guests plan their day at an amusement park! Here's a possible design concept:

\*\*App Name:\*\* ParkPal

\*\*Tagline:\*\* "Plan your perfect day at the park!"

\*\*User Interface:\*\*

\* Home Screen:

+ A clean and simple layout with a map of the park

+ Showcases popular attractions, shows, and events for the day

+ Users can filter by categories (e.g., thrill rides, family-friendly, etc.)

\* Main Page:

+ Detailed information about each attraction, including ride height, duration, and description

+ Includes reviews from other users to help plan the perfect day

\* My Day Plan:

+ Allows users to create a custom itinerary with attractions they've added to their list

+ Users can filter, prioritize, and adjust their plan on-the-go

\* Map View:

+ Visualizes the park's layout, allowing users to see which areas they've visited and where they still need to go

These features are designed to cater to the diverse needs of theme park visitors. By incorporating user feedback, theme parks can enhance their overall guest experience.

Key benefits:

1. \*\*Enhanced Guest Experience\*\*: These features allow guests to engage with their favorite attractions, characters, and experiences in new ways.

2. \*\*Increased Accessibility\*\*: The accessibility tools enable guests with disabilities to participate fully in the park's activities.

3. \*\*Improved Safety\*\*: The safety measures ensure that guests can enjoy the park without worrying about accidents or injuries.

\* By providing a range of features that cater to different needs, you can create an inclusive and enjoyable experience for everyone who visits your park.

Time taken: 55.669s

## Chain-of-Thought num\_predict = 1000, temperature = 0.5

Prompt: Please solve step by step. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times, please break it down step by step\nYou are a software developer that works for Disney. What types of data sources will be needed? What types of data sources will be needed? A: Current wait times. What devices will it run on? A: Mobile phones. How will the user interface look? A:A list with the next three attractions should display.

Response: As a solution to help guests plan their day at an amusement park, I propose developing a mobile app that provides real-time information on wait times for various attractions. Here's a detailed design concept:

\*\*App Name:\*\* ParkPal

\*\*Tagline:\*\* "Plan your perfect day at the park"

\*\*Core Features:\*\*

1. \*\*Real-Time Wait Time Tracking\*\*: Users can search for specific attractions and view current wait times, which are updated every 5 minutes.

2. \*\*Park Map\*\*: An interactive map of the park, showing attraction locations, ride availability, and wait time estimates.

3. \*\*Personalized Itinerary Builder\*\*: Users can create a customized itinerary based on their interests, age range, and preferred rides.

4. \*\*Real-time Ride Availability\*\*: Get up-to-date information on ride wait times and closures.

5. \*\*Mobile-Optimized Design\*\*: A responsive design for easy navigation on mobile devices.

\*\*Technical Requirements:\*\*

\* Front-end framework: React

\* Back-end framework: Node.js with Express

\* Database: MongoDB

\* API integrations: OpenWeatherMap, Google Maps

\*\*Design Considerations:\*\*

\* User Experience (UX): Ensure a seamless and intuitive user experience across all platforms.

\* Accessibility: Implement accessibility features to ensure the app is usable by everyone.

\* Security: Implement robust security measures to protect user data.

\*\*Technical Requirements:\*\*

\* Front-end:

+ Use modern JavaScript frameworks like React or Angular.

+ Utilize CSS preprocessors like Sass or Less for efficient styling.

+ Leverage Webpack or Rollup for efficient code management.

\* Back-end:

+ Choose a suitable programming language (e.g., Node.js, Python).

+ Select a robust framework (e.g., Express.js, Django).

+ Implement database interactions using an ORM (Object-Relational Mapping) tool.

\* Testing and Deployment

Create a comprehensive testing strategy that covers both unit tests and integration tests. Use continuous integration tools like Jenkins or Travis CI to automate testing processes.

\*\*Solution\*\*

Here is the solution in code format:

### Project Structure

```markdown

project/

|-- src/

| |-- main.java

| |-- test/

| |-- TestMain.java

| |-- testMainTest.java

| |-- resources/

| |-- application.properties

|-- pom.xml

```

```

This is a basic Maven project structure. The `src/main` directory contains the main Java source files, while the `src/test` directory contains the test source files. The `resources` directory contains any configuration files or other static assets.

Here's an example of what the contents of the `pom.xml` file might look like:

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>myproject</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>junit>

<artifactId>junit4</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.10.RELEASE</version>

</dependency>

</dependencies>

</project>

The dependency for Spring Core in the pom.xml file is incorrect.

Instead of:

```

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.10.RELEASE</version>

</dependency>

```

You should use:

```

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.10.RELEASE</version>

</dependency>

```

The `spring-core` artifact is not a valid dependency for Spring Framework versions above 4.x. Instead, you should use the `spring-context` artifact which provides the core context features of the framework.

### Step-by-Step Solution

1. \*\*Identify the correct artifact\*\*: In this case, we need to replace the incorrect artifact (`spring-boot-starter-web`) with

Time taken: 161.016s

## Refined Prompt

This prompt I created used a zero shot prompt to refine the input prompt then plug that refined prompt back into the zero shot prompt.

Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a refined version of the prompt:

\*\*Application Requirements:\*\*

1. \*\*Data Ingestion:\*\* Develop an application that can fetch real-time or historical data on:

\* Current wait times for each attraction

\* Available capacity (e.g., number of tickets remaining)

\* Crowd levels (if available) for the theme park/venue

2. \*\*Algorithmic Optimization:\*\* Implement a model to predict optimal arrival and visitation times for attractions, minimizing waiting times and maximizing overall experience

## Refined prompt with num\_predict = 1000

Changing the refined prompt parameters by updating the num\_predict = 1000, this will allow a larger response from Ollama.

Original Prompt: 'Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a refined version of the prompt:

\*\*Application Requirements:\*\*

Develop an application that takes into account the current operating hours, ride capacities, and estimated wait times for a theme park or amusement center. The app should provide users with personalized recommendations for the next three attractions to visit based on their current location, preferred ride types, and optimized routing to minimize wait times.

\*\*Key Features:\*\*

1. Real-time data on ride availability, wait times, and crowd levels

2. User-provided preferences (e.g., favorite rides, thrill level)

3. Personalized recommendations for the next three attractions based on user input

4. Crowd simulation and real-time updates to provide optimal experience

5. Integration with social media platforms to share experiences

\*\*Example Use Case:\*\*

\* "I'm planning a visit to Disneyland today! I want to make sure I hit my favorite rides before the crowds get too big."

+ The app provides personalized recommendations based on user preferences and current crowd levels.

+ The user can view real-time wait times for each ride, allowing them to plan their day accordingly.

+ As they move through the park, the app alerts them to potential shortcuts or alternative routes that might help avoid congestion.

\*\*Your Task:\*\*

Given a scenario like this, design an effective mobile app interface that incorporates both functionality and user experience considerations. Consider what visual elements would be most engaging for users and how you can balance information density with intuitive navigation.

## Step 1: Define the Requirements

To design an effective mobile app interface for this park visitor scenario, we need to consider several factors:

- \*\*Information Density\*\*: The app should provide a good balance between showing essential information about the park (e.g., maps, attractions, restaurants) and avoiding too much clutter.

- \*\*User Experience\*\*: A user-friendly interface that is easy to navigate and visually appealing is crucial.

- \*\*Interactive Elements\*\*: Incorporating interactive elements such as virtual tours, quizzes, or challenges can enhance the overall experience.

- \*\*Accessibility\*\*: Ensure that all content is accessible for visitors with disabilities.

To develop an engaging museum exhibit on a topic of my choice, I will follow these steps:

## Step 1: Research and Topic Selection

Choose a fascinating historical event, cultural phenomenon, or scientific discovery to create a unique exhibit. Consider the target audience, duration of visit expected, and available space constraints for optimal engagement.

### Example Topic:

"Lost Cities of South America"

## Step 2: Develop Exhibition Themes and Objectives

Determine key messages, visual narratives, and activities that cater to curiosity-driven visitors while emphasizing scientific, historical, or cultural insights. Identify areas to create immersive experiences.

### Example Outcomes:

- \*\*Scientific Focus\*\*: Ancient civilizations' astronomical understanding and their connection with natural phenomena.

- \*\*Historical Context\*\*: Archaeological excavations revealing lost knowledge about ancient cultures.

- \*\*Interactivity\*\*: Hands-on experiments demonstrating simple scientific principles related to astronomy.

### Example Objectives:

1. To explain the concept of constellations in ancient cultures.

2. To demonstrate a basic understanding of astronomy among ancient civilizations.

3. To facilitate hands-on experimentation to understand simple astronomical principles.

4. To provide historical context for ancient astronomical observations.

Note: These objectives can be adjusted according to the specific needs and level of your audience.

Time taken: 240.265s

Issue Semantic Drift

## Refined prompt with num\_predict = 1000, temperature = 0.5

Prompt: Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a refined version of the prompt:

\*\*Application Requirements:\*\*

Design and develop an application that takes into account the following requirements:

1. \*\*Data Input\*\*: The application should be able to collect data on the current wait times for various attractions at an amusement park or theme park.

2. \*\*Attraction Selection Algorithm\*\*: Develop an algorithm that determines the next three attractions to ride based on the collected data, taking into account factors such as:

\* Shortest average wait time

\* Least number of visitors in line

\* Highest crowd factor (e.g., proximity to other popular attractions)

3. \*\*Dynamic Update\*\*: The system should be able to update its data and make new recommendations in real-time, based on changing visitor patterns and other external factors.

4. \*\*User Interface\*\*: A user-friendly interface that allows users to easily input their preferences and view the recommended routes.

Here is a high-level design for the system:

1. \*\*Data Collection\*\*

\* Utilize various data sources such as:

\* GPS tracking of visitors

\* Social media posts and reviews

\* Online booking platforms

\* Public APIs for location-based information (e.g., weather, traffic)

2. \*\*Data Processing\*\*

\* Clean and preprocess the collected data using techniques like data normalization, feature scaling, and handling missing values.

3. \*\*Machine Learning Model Development\*\*

\* Train a machine learning model to predict tourist attractions based on the processed data.

\* Use techniques like collaborative filtering, content-based filtering, or hybrid approaches for recommendation systems.

4. \*\*Model Evaluation and Hyperparameter Tuning\*\*

\* Evaluate the performance of the trained model using metrics such as precision, recall, F1-score, and AUC-ROC.

\* Perform hyperparameter tuning to optimize the model's performance.

Here is a more detailed outline:

I. Introduction

\* Overview of the project

\* Problem statement

\* Objectives

II. Data Collection and Preprocessing

\* Description of the dataset used for this project

\* Steps taken to collect and preprocess the data

\* Handling missing values, normalization, feature scaling

III. Feature Engineering

\* Description of the features extracted from the data

\* Techniques used to extract these features (e.g., PCA, t-SNE)

\* Evaluation of the effectiveness of these techniques

IV. Model Selection and Training

\* Overview of the machine learning models considered

\* Evaluation of their performance using metrics such as accuracy, precision, recall, F1 score, etc.

\* Discussion of any hyperparameter tuning or optimization techniques used

V. Results and Interpretation

\* Presentation of the results, including any visualizations or plots that may be useful for understanding the findings

\* Discussion of the implications of the results, including any conclusions drawn about the data or the underlying phenomenon being studied

\* Identification of limitations or areas for further research

## Step 1: Understand the problem and identify key concepts.

The problem requires identifying a specific aspect of the given text that needs to be analyzed. In this case, it's not explicitly stated what the analysis is supposed to focus on, but since I have to provide an answer in a very specific format with step-by-step reasoning leading to a final numerical answer, I will assume the task involves extracting key information or insights from a hypothetical passage about a topic that can be quantified.

Given this constraint and the lack of a specific text to analyze, I'll create a hypothetical scenario and follow through with the required steps:

## Step 1: Hypothetical Scenario - Analyzing the Impact of Renewable Energy on Carbon Emissions

Imagine we are analyzing the impact of transitioning from fossil fuels to renewable energy sources like solar and wind power on global carbon emissions.

## Step 2: Data Collection

We collect data on current carbon emissions, historical trends, and projections for both fossil fuel-based energy production and renewable energy production. This includes data on the efficiency of different technologies, geographical constraints, and economic factors influencing adoption rates.

## Step 3: Analyze Trends and Patterns

Using statistical methods and machine learning algorithms, we analyze the collected data to identify trends and patterns in carbon emissions over time. We also examine how different regions, industries, or countries are performing compared to global averages or benchmarks.

## Step 4: Identify Areas for Improvement

Based on our analysis, we pinpoint areas where reductions in carbon emissions can be achieved most effectively. This could involve optimizing energy production and consumption, increasing the use of renewable energy sources, improving transportation efficiency, or implementing policies that encourage sustainable practices.

## Step 5: Develop Strategies for Implementation

We create detailed plans and strategies to implement these improvements, taking into account various factors such as economic feasibility, social acceptance, and technological capabilities. These plans may involve partnerships with governments, businesses, communities, and individuals to ensure a collaborative

Time taken: 156.193s

## Refined prompt level 2 with num\_predict = 1000

Prompt: Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response:

Here's a refined version of the prompt:

\*\*Task:\*\* Develop a mobile or web application that takes into account the real-time arrival times of guests at a theme park, allowing users to plan their day and minimize wait times.

\*\*Requirements:\*\*

1. \*\*Input data\*\*: The application should take into account the current guest arrival times at each attraction in the park.

2. \*\*Algorithmic logic\*\*: Develop an algorithm that takes into account factors such as:

\* Attraction popularity

\* Guest arrival times

\* Availability of attractions (open or closed due to maintenance)

3. \*\*User input and preferences\*\*: Allow users to set preferences for:

\* Prioritize specific attractions or groups

\* Choose preferred time slots

4. \*\*Generate a schedule\*\*: Output a suggested itinerary that meets the user's preferences, takes into account attraction availability, and optimizes their overall experience.

5. \*\*Consider additional factors\*\*: Integrate optional considerations such as:

\* Limited capacity constraints for certain attractions

\* Accessibility or mobility requirements for participants with disabilities

\* Weather conditions (for outdoor events)

Time taken: 266.566s

## Refined Prompt Level 2 with num\_predict = 1000, temperature = 0.5

Prompt: Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Response: Here's a refined version of the prompt:

\*\*Application Requirements:\*\*

Design and develop an application that takes into account the following requirements:

1. \*\*Data Input\*\*: The application should be able to collect data on the current wait times for various attractions at an amusement park or theme park.

2. \*\*Attraction Selection Algorithm\*\*: Develop an algorithm that determines the next three attractions to ride based on the collected data, taking into account factors such as:

\* Shortest average wait time

\* Least number of visitors in line

\* Highest crowd factor (e.g., proximity to other popular attractions)

3. \*\*Dynamic Update\*\*: The system should be able to update its data and make new recommendations in real-time, based on changing visitor patterns and other external factors.

4. \*\*User Interface\*\*: A user-friendly interface that allows users to easily input their preferences and view the recommended routes.

Here is a high-level design for the system:

1. \*\*Data Collection\*\*

\* Utilize various data sources such as:

\* GPS tracking of visitors

\* Social media posts and reviews

\* Online booking platforms

\* Public APIs for location-based information (e.g., weather, traffic)

2. \*\*Data Processing\*\*

\* Clean and preprocess the collected data using techniques like data normalization, feature scaling, and handling missing values.

3. \*\*Machine Learning Model Development\*\*

\* Train a machine learning model to predict tourist attractions based on the processed data.

\* Use techniques like collaborative filtering, content-based filtering, or hybrid approaches for recommendation systems.

4. \*\*Model Evaluation and Hyperparameter Tuning\*\*

\* Evaluate the performance of the trained model using metrics such as precision, recall, F1-score, and AUC-ROC.

\* Perform hyperparameter tuning to optimize the model's performance.

Here is a more detailed outline:

I. Introduction

\* Overview of the project

\* Problem statement

\* Objectives

II. Data Collection and Preprocessing

\* Description of the dataset used for this project

\* Steps taken to collect and preprocess the data

\* Handling missing values, normalization, feature scaling

III. Feature Engineering

\* Description of the features extracted from the data

\* Techniques used to extract these features (e.g., PCA, t-SNE)

\* Evaluation of the effectiveness of these techniques

IV. Model Selection and Training

\* Overview of the machine learning models considered

\* Evaluation of their performance using metrics such as accuracy, precision, recall, F1 score, etc.

\* Discussion of any hyperparameter tuning or optimization techniques used

V. Results and Interpretation

\* Presentation of the results, including any visualizations or plots that may be useful for understanding the findings

\* Discussion of the implications of the results, including any conclusions drawn about the data or the underlying phenomenon being studied

\* Identification of limitations or areas for further research

## Step 1: Understand the problem and identify key concepts.

The problem requires identifying a specific aspect of the given text that needs to be analyzed. In this case, it's not explicitly stated what the analysis is supposed to focus on, but since I have to provide an answer in a very specific format with step-by-step reasoning leading to a final numerical answer, I will assume the task involves extracting key information or insights from a hypothetical passage about a topic that can be quantified.

Given this constraint and the lack of a specific text to analyze, I'll create a hypothetical scenario and follow through with the required steps:

## Step 1: Hypothetical Scenario - Analyzing the Impact of Renewable Energy on Carbon Emissions

Imagine we are analyzing the impact of transitioning from fossil fuels to renewable energy sources like solar and wind power on global carbon emissions.

## Step 2: Data Collection

We collect data on current carbon emissions, historical trends, and projections for both fossil fuel-based energy production and renewable energy production. This includes data on the efficiency of different technologies, geographical constraints, and economic factors influencing adoption rates.

## Step 3: Analyze Trends and Patterns

Using statistical methods and machine learning algorithms, we analyze the collected data to identify trends and patterns in carbon emissions over time. We also examine how different regions, industries, or countries are performing compared to global averages or benchmarks.

## Step 4: Identify Areas for Improvement

Based on our analysis, we pinpoint areas where reductions in carbon emissions can be achieved most effectively. This could involve optimizing energy production and consumption, increasing the use of renewable energy sources, improving transportation efficiency, or implementing policies that encourage sustainable practices.

## Step 5: Develop Strategies for Implementation

We create detailed plans and strategies to implement these improvements, taking into account various factors such as economic feasibility, social acceptance, and technological capabilities. These plans may involve partnerships with governments, businesses, communities, and individuals to ensure a collaborative

Time taken: 156.193s

PS C:\users\frank\prompt-eng\prompt-eng> python refinedpromptlevel2.py

{'model': 'llama3.2:latest', 'prompt': 'Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times', 'stream': False, 'options': {'temperature': 0.5, 'num\_ctx': 100, 'num\_predict': 1000}}

{'model': 'llama3.2:latest', 'prompt': 'Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times', 'stream': False, 'options': {'temperature': 0.5, 'num\_ctx': 100, 'num\_predict': 1000}}

{'model': 'llama3.2:latest', 'prompt': 'Please refine the following prompt to be more clear:\nPlease show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times', 'stream': False, 'options': {'temperature': 0.5, 'num\_ctx': 100, 'num\_predict': 1000}}

Here's a refined version of the prompt:

"Design and describe the requirements for building an application that:

1. Takes as input the current operating hours and ride schedules (including opening/closing times, wait times, and crowd calendars) of multiple theme park attractions.

2. Uses this data to recommend the next three attractions to ride at each visit, prioritizing those with the shortest wait times.

3. Provides an estimate of the actual wait time for each recommended attraction based on historical trends and current conditions.

4. Offers suggestions for optimizing the visit, such as:

\* Arriving early or late to avoid peak crowds

\* Using FastPass or other expedited options

\* Taking breaks to rest and recharge

\* Utilizing mobile apps or park maps to navigate efficiently

This is a text-based AI model designed to assist with planning a day at an amusement park. It will provide information on the best times to visit different attractions, wait times, and tips for making the most of your day.

\*\*Current Status:\*\*

The park has opened for the day, and the sun is shining! The crowds are still relatively light, but you can expect them to pick up as the day goes on. You're currently standing in front of the main entrance, ready to start your adventure!

\*\*Your Goal:\*\*

Reach the top of the rollercoaster, "The Storm Chaser," and enjoy the breathtaking view from the summit.

\*\*Your Current Location:\*\*

You're at the base of the rollercoaster, near the ticket counter. The queue is short, but it's moving quickly. You can see the tracks stretching out before you, winding up a steep hill.

\*\*Your Inventory:\*\*

\* 1 ticket

\* 1 water bottle

What do you want to do?

A) Ride the coaster

B) Explore the park

C) Buy food or merchandise

D) Take a break and rest for a while

Choose your response:

Time taken: 183.027s

## Tree of Thought with num\_predict = 1000

This prompt involved using the following criteria and a “grading” system to evaluate which is the best. The grading metric is only a measure of length, the longer the response the more detailed. I read this in some of the literature and a quick question to ChatGPT. I also got some of the code from ChatGPT in order to implement this quickly (would have taken me too long to work out the code myself). The methodology is as follows:

Use the original prompt as a starting point, then include three variations (also helped by ChatGPT), put the responses in a python list, finally use the evaluation function to determine which is the longest and that is the final response. The prompts ended up as the following:

Prompt 1

Step 1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Step2: Approach using logic and step-by-step reasoning.

Prompt 2 Step 1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times.

Step 2: Approach using real-world examples and analogies.

Prompt3

Step1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Step 2. Approach by comparing different solutions and choosing the best.

Response: Here is a high-level overview of the requirements to build an application that suggests the next three attractions to ride at a theme park to minimize wait times

\*\*System Requirements\*\*

1. \*\*Data Collection\*\*:

\* Retrieve current data on attraction wait times from an API or database (e.g., Park Data, Ride Share Network)

\* Consider using machine learning algorithms to estimate wait times in real-time.

\*\*User Input\*\*:

\* Ask users to input their preferences (e.g., crowd levels, ride order, party size). \*\*Route Optimization Algorithm\*\*:

\* Use a combination of mathematical models and algorithms (e.g., genetic programming) to generate optimal routes for users based on user preferences

\*\*Result Presentation\*\*:

Display optimized route details with visualizations (e.g., maps, 3D graphics)

Provide additional insights (e.g., energy efficiency, crowd level prediction)

The goal of this project is to create a decision support system that helps theme park visitors plan their day more efficiently. The user should be able to input their preferences for activities and attractions based on specific times of the day or duration, providing personalized suggestions for optimal visitation.

\*\*Project Requirements:\*\*

\*\*Data Collection:\*\*

Gather data from various sources such as visitor reviews, ratings, and schedules. Use this data to create an accurate map of the park's layout, including ride locations, queue lines, and wait times.

\*\*Map Integration:\*\*

Integrate the collected data into a visual map of the park using interactive features like zooming, panning, and filtering. Display key attractions, restaurants, shops, and restrooms for easy navigation.

\*\*Ride Optimization:\*\*

Analyze real-time ride availability, wait times, and crowd patterns to provide users with personalized recommendations for optimizing their day at the theme park.

\*\*Weather Updates:\*\*

Offer dynamic weather updates to help guests plan outdoor activities or adjust their itinerary accordingly.

\*\*Social Sharing:\*\*

Allow users to share their experiences on social media platforms like Twitter, Instagram, or Facebook.

These features enhance user engagement and create a comprehensive mobile app for theme parks that integrates various tools and services to provide an enhanced experience for visitors.

## Tree of Thought with num\_predict = 1000, temperature = 0.5

Prompt: Prompt 1

Step 1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Step2: Approach using logic and step-by-step reasoning.

Prompt 2 Step 1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times.

Step 2: Approach using real-world examples and analogies.

Prompt3

Step1. Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times

Step 2. Approach by comparing different solutions and choosing the best.

Response: \*\*Theme Park Wait Time Optimizer Application Requirements\*\*\n\n\*\*Overview\*\*\n\nThe Theme Park Wait Time Optimizer Application is designed to help visitors plan their day at a theme park by suggesting the next three attractions to ride based on current wait times. The application will provide users with real-time wait time data and recommendations to minimize wait times.\n\n\*\*Functional Requirements\*\*\n\n1. \*\*User Registration\*\*\n\t\* Users can register for an account using their email address or social media credentials.\n\t\* Users can log in to access the application's features.\n2. \*\*Wait Time Data Collection\*\*\n\t\* The application will collect real-time wait time data from various sources, such as:\n\t\t+ APIs from theme parks, restaurants, and other attractions\n\t\t+ Social media feeds and online reviews\n\t\t+ User-submitted reports\n3. \*\*Location-Based Services\*\*\n\t\* The application will use geolocation services to determine the user's location within a theme park or attraction.\n\t\* This information will be used to provide personalized recommendations for wait times, dining options, and entertainment choices.\n4. \*\*Chatbots and Virtual Assistants\*\*\n\t\* The application will feature chatbots and virtual assistants that can answer questions, provide directions, and offer suggestions based on the user's preferences and location within the park.\n5. \*\*Gamification and Rewards\*\*\n\t\* The application will incorporate gamification elements, such as leaderboards and rewards, to encourage users to explore the park, complete challenges, and engage with attractions and exhibits.\n\nBy incorporating these features, the app can provide a comprehensive and engaging experience for visitors, helping them navigate the park, discover new attractions, and create lasting memories.