iclrfinalcopy

REAL-TIME LAYOUT ADAPTATION USING GENERATIVE AI

Anonymous authors
Paper under double-blind review

AUTHOR NOTE

First Section: An overview about the how the research got instantiated .
Second section: Utilization of AI Models to make more precise analysis
Third section: Our purpose/ goal that we would want to aim via the research.
Final section: A retrospective view of our research and conclusion.

ABSTRACT

In modern web design, ensuring adaptability and user engagement through dynamic layouts is increasingly important. With the growing demand for personalized user experiences, traditional static web layouts are insufficient for meeting user preferences. This paper introduces an innovative approach that leverages generative AI to dynamically adapt web layouts in real-time. With the help of data that is collected under the banner of user interactions through technologies such as java script and node.js, we are able to save those interactions which not only include the click patterns, but also the timestamps, user's name, day and date and number of clicks. These clicks correspond to interactions of users with different react components. This data is being stored as a csv file as it is more easier to read when it comes to parsing it to an AI model. Once every designated cycle, the data is fed to a python script which does an API call to the Chat GPT 4o model which then analyzes the data and re-writes the CSS to create a new web layout which is based on the user's interactions. This successfully gives a web interface that adapts its layout in real-time, which is somewhat similar to many recommended systems of popular applications like netflix, amazon prime etc. Its significance extends across multiple fields, as this approach can enhance user engagement by dynamically displaying components based on user interaction patterns. Additionally, it offers potential revenue growth for companies, allowing them to charge higher rates for ads strategically placed in high-engagement areas of the layout, based on inferred user data.

1 INTRODUCTION

Similar to the recommendation systems used by platforms like Netflix and Amazon Prime, the key motivation for integrating GenAI into FullStack development was to bring that concept to web applications. The goal is to create an immersive and adaptive user experience where behavioral analysis dynamically adjusts the positioning of elements on the webpage.

2 METHODOLOGY

2.1 INITIALIZATION

Before integrating GenAI with the frontend, our primary goal was to extract reusable React components from Figma files to streamline various development tasks. To achieve this, we provided GenAI with three key inputs: a JSON file extracted via Figma's API, a base64-encoded image for enhanced feedback, and a custom prompt to address specific design requirements. This setup allowed the project to generate reusable React components based on Figma designs. This led to a pivotal observation: with the React components generated using the established architecture, we could automate their deployment, effectively eliminating the need for manual intervention by a moderator.

2.2 DEVELOPMENT

4 DATA ANALYSIS VIA GPT4O

After successfully collecting user interaction data, we developed a Python script to analyze this data and update the webpage's CSS dynamically. The script utilizes libraries such as Pandas, OpenAI, and OS to facilitate the analysis and modifications. The process begins by using Pandas to read and analyze the CSV file containing user interaction data. The script focuses on the last row of the CSV file to determine the recently logged in user so as to know if the username in this row exists elsewhere in the data.

4.1 USERNAME IN THE DATA-SET

If the username is found in the contents of CSV except the last row via a function check_username_in_csv, the script invokes the function update_css_from_interactions. This function takes three arguments: the username, the location of the CSS file, and the user query. Next, the CSS file, which contains the web page layout information, is opened and its contents are loaded into a temporary variable named css_file. The function interact_with_csv is then called with the user query, the address of the CSV file, and the contents of the CSS file as arguments. This function performs the following steps:

- Convert CSV to Data-Frame: It uses Pandas to convert the CSV data into a Data-Frame.
- Generate Prompt: A prompt is created based on the user query.
- Generate Custom CSS: The prompt is sent to the generate_text function, which utilizes the GPT-4 model to generate custom CSS.

Here is the generate_text function:

```
def generate_text(prompt):
    chat_completion = client.chat.completions.create(
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        model="gpt-4o"
    )
    return chat_completion.choices[0].message.content
```

The generate_text function communicates with the GPT-4 model to generate a custom CSS file. This CSS file is designed not only to match the existing format for consistency but also to reflect the user's interactions, thereby dynamically adjusting the webpage layout according to user preferences.

4.2 USERNAME NOT IN THE DATA-SET

If the username from the last row of the CSV file is not found in the dataset, indicating that the current active user is visiting the website for the first time, the load_original_css function is called. This function accepts two arguments: the path to the original CSS file (og_css_file) and the path to the actual CSS file used by the webpage. The contents of the (og_css_file) are read and then written to the actual CSS file. This ensures that new users, who have never visited the website before, are presented with the default layout. As they interact with the page, the layout can be customized based on their interactions, providing a personalized experience over time.

5 GOAL

5.1 REINFORCEMENT LEARNING (RL) FRAMEWORK FOR DYNAMIC WEBSITE LAYOUT OPTIMIZATION

In this problem, the goal is to optimize the layout of a website to maximize user engagement and specific business objectives (such as increasing user time on the site, clicks, or revenue). The layout changes are dynamically applied based on user interactions.

5.2 STATE (S)

The state s_t at time t represents the current layout of the webpage along with the user's interaction data. This data can include:

- Component positions and sizes (as defined by CSS),
- User clicks on components,
- Time spent on different sections of the webpage,
- User engagement metrics like scroll depth, bounce rate, and interaction frequencies.

$$s(t) = \{\text{current layout, user clicks, time spent, engagement metrics}\}$$

5.3 ACTIONS (A)

The actions a_t represent the potential modifications to the webpage's CSS layout. These actions can include:

- Repositioning components (e.g., moving a button or banner),
- Resizing components (e.g., changing the size of images or text areas),
- Recoloring or restyling elements (e.g., adjusting font size or colors to improve visibility).

$$a(t) = \{\text{layout modification: component resizing, repositioning, etc.}\}$$

5.4 REWARD (R)

The reward $R(s_t, a_t)$ is a function that quantifies the success of an action based on the change in user engagement. This reward can be constructed by considering the following factors:

- Maximizing user time on the website,
- Encouraging specific actions, such as clicks on ads or buttons,
- Minimizing the bounce rate (i.e., keeping users engaged and preventing them from leaving),
- Increasing overall revenue generated from user interactions.

You can define the reward function as:

$$R(s_t, a_t) = \alpha\cdot(\text{user time on website})+\beta\cdot(\text{click-through rate})-\gamma\cdot(\text{bounce rate})+\delta\cdot(\text{revenue from ads})$$

where α, β, γ, and δ are tunable coefficients that represent the relative importance of each metric.

6 OBJECTIVE

The objective is to learn an optimal policy π that maximizes the cumulative reward over time. The goal is to find the policy that tells the system which layout changes should be made to maximize long-term user engagement and business metrics.

The cumulative reward is mathematically defined as:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$$

Where:

- Π is the policy (the strategy of adjusting the layout based on user interactions),
- $R(s_t, a_t)$ is the reward at time t,
- $\gamma \in [0, 1]$ is the discount factor that controls the trade-off between immediate and future rewards (e.g., a higher γ means that long-term rewards are prioritized over short-term gains).

7 CONCLUSION

The problem is cast as a Markov Decision Process (MDP) where:

- States represent the layout and user interactions,
- Actions are layout modifications,
- Rewards are computed based on user engagement and business goals.

The RL algorithm's objective is to learn a policy π that optimally adjusts the layout in real-time to maximize the cumulative reward by improving user engagement and achieving specific business outcomes.

8 RESULT

Reinforcement learning (RL), a subfield of unsupervised learning, provides a goal-driven framework ideal for optimizing dynamic website layouts. By integrating supervised learning, using existing user data for predictions, this approach enhances the accuracy of layout optimization. The combination allows the system to not only adapt based on user interactions but also predict user preferences, moving toward more effective outcomes.

While human behavioral patterns serve as core inputs to the model, the reward system plays a crucial role in guiding layout adjustments. It addresses the challenge of accurate predictions, ensuring that the layout aligns with user behavior. This cyclical process continuously refines the user experience, making the website more responsive to individual preferences.

Beyond increasing engagement and revenue, this method offers valuable insights into how users interact with components, allowing developers to design layouts that cater more effectively to user attention. As layouts dynamically evolve in response to real-time data, this system fosters a personalized user experience, advancing both design quality and user satisfaction.