Summary-attack website Documentation

Li Jui Tseng

August 12, 2024

1 Introduction

This document provides a comprehensive overview of the summary-attack website built using Next.js for the front end and Flask for the back end. The application is structured to leverage modern web development practices, including server-side rendering, responsive design, and RESTful API services.

2 Front End: Next.js with Tailwind CSS

2.1 Overview

The front end of the application is built using Next.js, a React-based framework that enables server-side rendering and static site generation. Tailwind CSS is used for styling, providing utility-first CSS classes to create a responsive and modern UI without writing custom CSS.

2.2 Technologies Used

- **Next.js:** A React framework that enables server-side rendering and static site generation.
- **React:** A JavaScript library for building user interfaces.
- **Tailwind CSS:** A utility-first CSS framework that allows rapid UI development with minimal custom CSS.
- **Daisyui:** A tailwindcss library with custom made components to facilitate the creation and utilization of components

2.3 Project Structure

The front-end code is organized into a series of pages and components. Below is a breakdown of the main pages:

2.3.1 /

The `/` page serves as the landing page of the website. It includes:

- A title for the website
- Links to perturbations and how it work pages
- A table introducing the command line functions.

2.3.2 /how-it-works

The `/how-it-works` page allows users to understand the main attacks described in the paper.

- Description of the setup for poisoning attack and influence functions
- Link to datasets

2.3.3 /perturbations

The `/perturbations` page allows users to understand the main attacks described in the paper.

- Explains the 11 different attacks in detail with examples for each one.

2.3.4 /demo

The `/demo` page allows the user to try the demo of summary-attack.

- A dropdown to let the user select the different options for different combination of attacks.
- Connected to the backend for POST.
- Get's a response back from server as JSON and displays it to the end user in a custom component.

2.3.5 pictures and others

Components such as pictures are located inside the folder /public/images/. Images and videos can be added as desired in the previosuly described folder.

2.4 Styling with Tailwind CSS

Tailwind CSS is employed to ensure the application is both responsive and visually appealing. Some key features include:

- **Responsive Design:** Tailwind's responsive utility classes (`sm, md, lg, xl`) are used to make the application adapt to different screen sizes.
- **Custom Components:** Custom components such as the display text in demo are created using Tailwind's utility classes to maintain consistency throughout the app.
- **Dark Mode:** Can be implemented in future versions.

3 Back End: Flask API

3.1 Overview

The back end is built using Flask, a lightweight Python web framework. Flask provides the necessary API endpoints for the front end to interact with. It runs on port 5000 and handles all the server-side logic, including data processing and conversion from ui to the command line for the /demo page in the front end.

3.2 Technologies Used

- **Python:** The programming language used for back-end logic.
- **Flask:** A micro-framework for Python used to create the web server and API endpoints.

3.3 API Endpoints

The back end consists of a series of RESTful API endpoints that the front end communicates with. Below is a breakdown of the main endpoints:

3.3.1 POST /predict

This endpoint receives data from the front end. It expects a JSON payload with the necessary parameters (described below).

```
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    model = data['model']
    dataset = data['dataset']
    split = data['split']
    size = data['size']
    perturbation = data['perturbation']
```

3.3.2 Server Configuration

The Flask server is configured to run on port 5000. The server can be started using the following command:

```
python3 server.py or python server.py
```

3.4 Notes

- Server may return an error if when running the command line, the prompt "Are you sure you want to run the custom hugging face" appears, and since there's no pre response hard coded, it won't be able to run the command.
- Problem stated above can be fixed by hard-coding a response for the cmd.

4 Deployment

4.1 Front End Deployment

The front-end Next.js application can be deployed on platforms like Vercel, Netlify, or any other static hosting service free of charge.

4.2 Back End Deployment

The Flask API can be deployed on a variety of platforms such as Heroku, AWS, or DigitalOcean. Deployment steps include:

- Setting up the server environment with Python and Flask.
- Configuring the server to run on port 5000.
- Ensuring the server is accessible from the internet and can handle API requests.

4.3 Challenges

- It could be difficult to host the backend since the command line requires hugginface, therefore taking a lot of space and computing power, might not find a free hosting option while maintening a smooth operation.

5 Upcoming version

It is expected that new functions and sections will be coming in the next release version.