VOICE BASED UI FOR GEOSPATIAL MAPS

**OVERVIEW**

The **Voice-Activated Geospatial Maps** project is a web application that integrates voice recognition with interactive geospatial maps. The application allows users to perform specific tasks on the map interface via voice commands. It leverages advanced speech-to-text technology to interpret user commands, a FastAPI backend to process commands, and a map visualization library (Leaflet) to display results.

**KEY FEATURES**

* **Voice Recognition**: Uses the Whisper speech-to-text model to capture and interpret user commands.
* **Geospatial Mapping**: Utilizes Leaflet to render interactive maps with location-based overlays.
* **Real-time Command Processing**: Converts voice input into actionable tasks and updates the map interface accordingly.
* **Location Extraction**: Identifies and uses location information within commands to dynamically adjust the map view.

**Technologies Used**:

* **Frontend**: JavaScript, HTML, CSS for UI.
* **Backend**: FastAPI for processing commands and serving data.
* **FFMPEG**: Tool for converting .webm audio files recived to .wav.
* **Speech-to-Text**: Whisper ASR (small-en from huggingface) for handling speech recognition tasks.
* **Named**-Entity-Recognition: We used BERT’s ner pipeline.
* **SentenceTransformers : We used paraphrase-MiniLM-L6-v2(Huggingface) to get encodings of commands and mappings.**
* **Map Visualization**: Leaflet for map displays and overlays.
* **Location Extraction**: OpenCage Geocoder(Free Tier Api) for converting text-based location data into map coordinates.

**System Architecture:**

**FrontEnd: The browser-side application which displays maps, and provides the UI for the user to interact with the map.**

**Backend server: Handles api requests from the front end. All models are hosted and processed on this server**

:

How It Works:

* We use Leaflet.js to display maps and overlays with location-based data.
* We use OpenStreetMap for street map, Googleosm for street and satellite view. We use cartocdn api for the street/road overlay for the satellite view.
* There is a search bar which allows you to search on places. Clicking on the results changes the map view to that location (Also marks the location with a pin).
* The user interacts with the app via a microphone icon that triggers the voice command input. Press the button to start recording, press it again to stop.
* The audio file is send to the Backend in the .webm format.

 **Backend Processing**:

* Receives audio data via FastAPI which it converts to .wav file using ffmpeg before sending it to the model.
* The Whisper model produces a transcript of the audio in English and passes it on to the sentence transformer.
* The sentence transformer is used to encode the given transcript and map to the target\_actions(explained in the next section). [The command is mapped to that target action which has the highest cosine similarity].
* Separately, the speech transcript is used by the NER pipeline to extract the location arguments, if any and uses them to obtain coordinates for the OpenCage Geocoder to resolve any place names or coordinates for map placement.
* The response containing the target action or command along with the necessary arguments is sent to the frontend application.
* The front end handles the response from the server, and makes the map work accordingly.

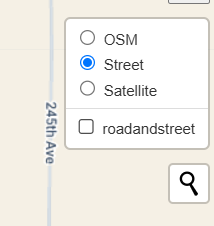
**Functionalities**

* **Voice Commands:**
  + Zoom In: The app zooms to any location mentioned or simply just zooms in on the current location if a specific location is not mentioned. Eg commands: zoom in to Bangalore, zoom in,etc.
  + Zoom Out: Zoom out doesn’t accept location arguments. It simply zooms out from the current location.
  + Directions: Gives the route from current location to another location. If two locatons args are passed, it gives the direction from the former to the latter. Eg commands: Show me directions to hubli, Give me the way from Chennai to Bangalore,etc.

Note: The routing is done by leaflets own OSRM demo server, which is not suitable for production/ high traffic.

* + Search: Just shifts the view to the desired location. Eg: Search for Mumbai
  + Reset: Shifts the view back to your current location.
  + Switch: Change the map type. Example: Switch to street view, switch to satellite map, switch to osm map, etc.
* **Error Handling**: Ensures graceful degradation if speech recognition fails or if the location is unavailable.

Issues:

* Browser asks for permission each time we press the microphone to record.
* There is no limit in place to stop recording after some time.
* Using leaflets’s demo routing engine- may cause issues or slowdowns.
* Sometimes using the word ‘show’ maps the command to switch, which might not be whats intended by the user in all cases.
* Overall accuracy- since we were constrained to using smaller models, observed ccuracy is not high.
*  the road and street overlay checkbox must be hidden to prevent issues with the overlay on the satellite map, if the user clicks on it.

Future Improvements:

* Creating a more user friendly interface (perhaps using react instead of vanilla js).
* Setting up our own routing engine, our own geocoding server, our own map server, etc instead of relying on external API’s.
* Stop recording automatically after the user stops speaking.

Api’s used:

http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}

https://{s}.basemaps.cartocdn.com/light\_only\_labels/{z}/{x}/{y}{r}.png

https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png