3 Streamlit App Integration

A demonstration web application (9_subject_line_ui_1.py) was created using Streamlit to provide an interactive way to explore the fine-tuned model's performance on new, unseen email bodies.

3.1 Purpose:

- Showcase the practical functionality of the trained model.
- Provide a user-friendly interface for testing and exploration.
- Serve as a key deliverable demonstrating the project's outcome.

3.2 Implementation Details:

- **Framework:** Streamlit (streamlit library)
- **Core Libraries:** transformers, torch
- **Model Loading:**
    - The application loads the fine-tuned facebook/bart-large model checkpoint saved during the training phase (specifically, the best checkpoint identified from run ...174424, path defined in MODEL_PATH).
    - Streamlit's @st.cache_resource decorator is used to load the model and tokenizer only once, significantly improving responsiveness.
- **Device Handling:** Automatically detects and uses CUDA GPU if available, otherwise defaults to CPU. The active device is displayed.
- **Generation Logic:**
    - User input text undergoes basic whitespace normalization.
    - The BART tokenizer preprocesses the text (truncation to 512 tokens).
    - The model generates predictions using beam search (num_beams=4) with a maximum subject length constraint (max_length=32+2).
    - Generated token IDs are decoded back to text.
- **User Interface:** Standard Streamlit widgets (st.title, st.markdown, st.text_area, st.button, st.spinner, st.info, st.caption, etc.) are used to create the layout and display information.

3.3 Functionality & Demonstration:

1. The application interface presents a title ("📧 Email Subject Line Generator"), brief instructions, and information about the underlying model (bart-large-finetune-...) and execution device (CPU/GPU).
2. A large text area allows the user to paste or type the body of an email.
3. Clicking the "✨ **Generate Subject**" button triggers the inference process, displaying a spinner while processing.
4. The model generates a subject line based on the input body.
5. The resulting subject line is displayed clearly in a formatted info box (**bold text**) below the button.

Example Outputs:

The following screenshots demonstrate the application generating subjects for different email inputs:

- **Example 1:** Input regarding a wi-fi issue on a Motorola phone. The model generated the succinct subject "**wifi**".

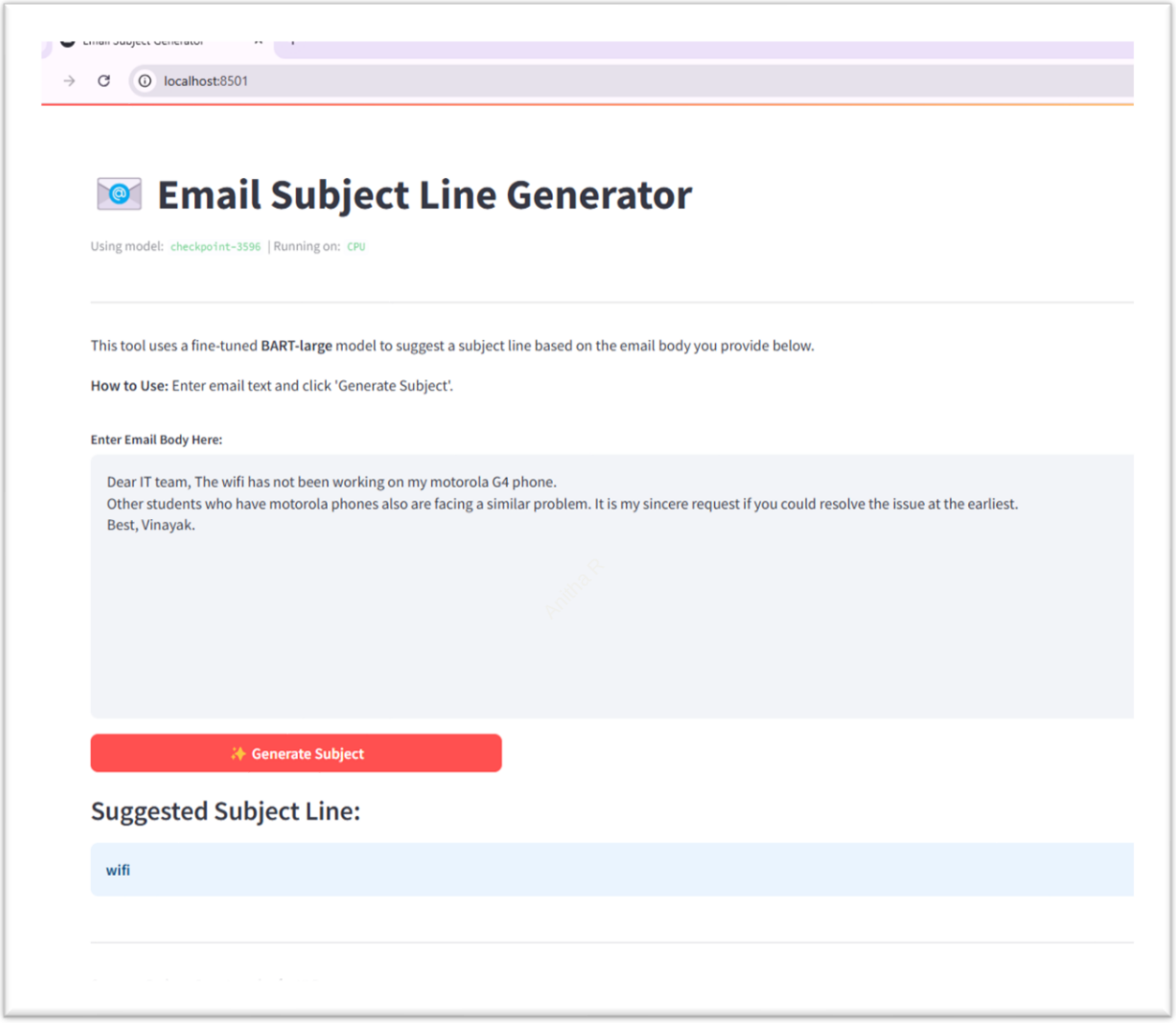
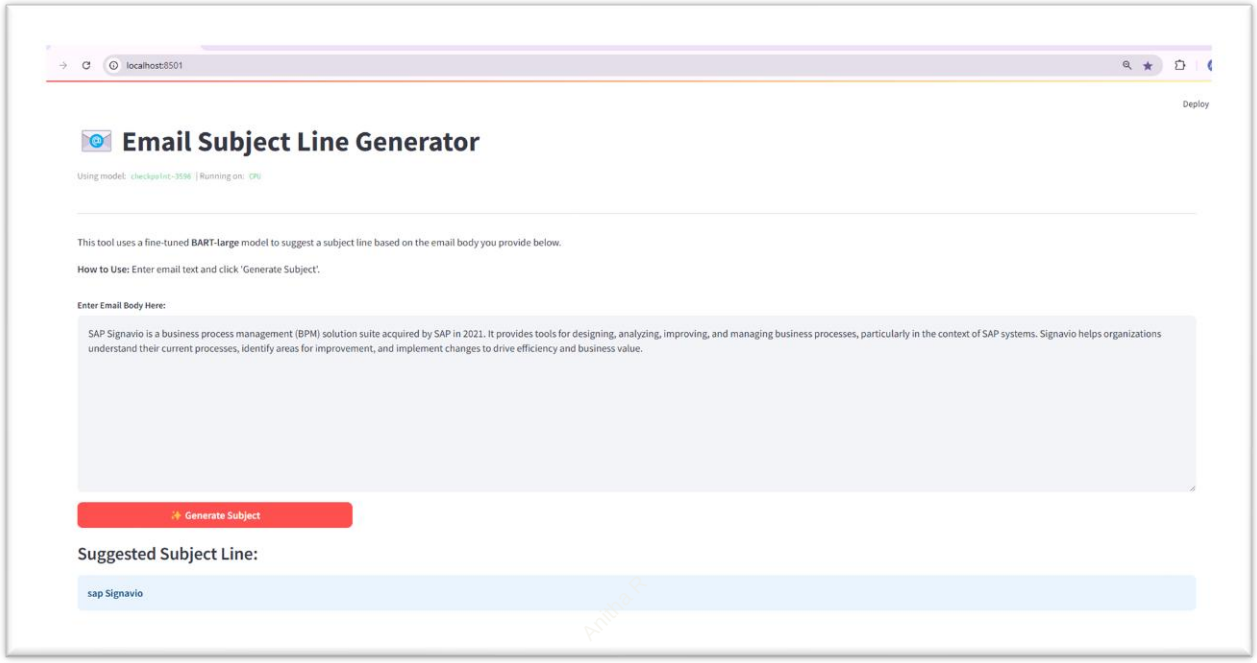


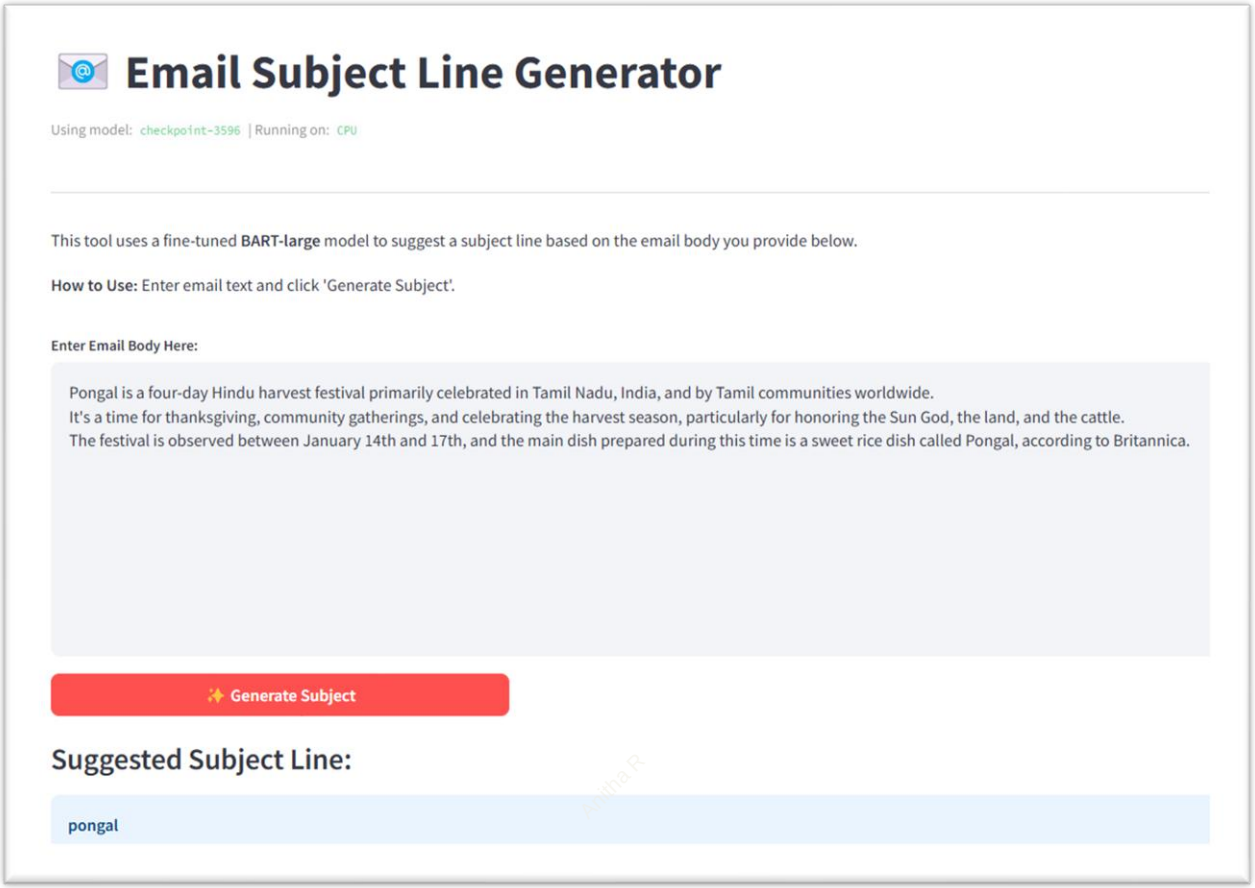
Figure 1: Streamlit app generating subject for wi-fi issue email.

- **Example 2:** Input describing the SAP Signavio business process management suite. The model generated "**sap signavio**".



Caption Figure 2: Streamlit app generating subject for SAP Signavio description.

- **Example 3:** Input describing the Pongal festival. The model generated "**pongal**".



Caption Figure Z: Streamlit app generating subject for Pongal festival description.

Demo Video Link:

https://github.com/AR-Version2/AUTOMATED-EMAIL-SUBJECT-LINE-GENERATION-USING-DEEP-LEARNING/blob/main/Output/Demo%20Video.mp4

These examples show the model's ability to identify key terms or topics from the input text and generate relevant, albeit sometimes very brief, subject lines.

3.4 How to Run:

- **Prerequisites:**
    - Python environment with necessary libraries installed: pip install streamlit transformers torch pandas numpy.
    - Access to the saved fine-tuned model directory.
- **Configuration:** Ensure the MODEL_PATH variable within the script points correctly to the saved fine-tuned model directory.
- **Execution:** Open a terminal, navigate to the script's directory, and run:
- streamlit run 9_subject_line_ui_1.py
- Access the app via the local URL provided (usually http://localhost:8501).

3.5 Deployment:

The application was tested locally. As a standard Streamlit application, it is easily deployable to various cloud platforms (Streamlit Community Cloud, Heroku, Google Cloud Run, AWS, etc.) by providing the script, a requirements.txt file, and making the model files accessible to the deployed instance.