Task 2 Multi-Label Emotion Recognition from Text Report

1. Dataset Preprocessing Steps

In this project, instead of manually preprocessing a new dataset, we are using a pre-trained model fine-tuned on the GoEmotions dataset by Google.
However, based on the nature of the GoEmotions dataset, the general preprocessing involved:

- **Tokenization**
  The input text provided by the user is tokenized using the AutoTokenizer from Hugging Face. It adds necessary padding and truncates text if it's too long to fit into the model's expected input size.
- **Text Formatting**
  The text is converted into a format understandable by the model (tensors) using return_tensors="pt" (PyTorch tensor format).
- **Threshold Filtering**
  After model inference, predicted probabilities are filtered based on a user-selected threshold to show only strong emotion predictions.

2. Model Selection and Rationale

- **Model Selected**
  goemotions_model (custom name here) — a Transformer-based model fine-tuned on the GoEmotions dataset, typically based on architectures like BERT or RoBERTa.
- **Why this model?**
  - **Multi-label Classification:** GoEmotions requires the ability to predict multiple emotions at once, not just one — Transformer-based models handle this well.
  - **Fine-tuning on Emotional Data:** The model was specifically trained on ~58k carefully labeled Reddit comments across 27 emotion labels and a neutral label.
  - **Strong Generalization:** Transformer models like BERT generalize well to unseen text and diverse language styles.
- **Tools Used:**
  - **Hugging Face Transformers** for easy loading and usage.
  - **PyTorch** for inference.

3. Challenges Faced and Solutions

Challenge	Solution Implemented
Mapping raw model labels (LABEL_0, LABEL_1, etc.) to understandable emotion names.	Created a manual dictionary, emotion_mapping and updated mappings from config.json.
Performance issues (reloading model every time on Streamlit reload).	Used @st.cache_resource decorator to cache the model and tokenizer, reducing load time.
Handling multiple emotion predictions simultaneously with different confidence levels.	Applied sigmoid activation to logits to output independent probabilities for each label, allowing multi-label prediction.
No visible output if no prediction crossed the threshold.	Added a friendly info message if no emotions are detected above the threshold.
Dynamic threshold adjustment without reloading the model.	Used Streamlit sliders to allow real-time control of prediction threshold.

4. Results with Visualizations and Interpretations

Results:

- Users input any sentence.
- The model returns multiple emotions with their corresponding confidence scores (between 0 and 1).
- Only emotions scoring above the selected threshold are shown.

Visualization (Suggestion for Improvement)

Currently, the app prints emotions with their confidence values.

You could enhance visualization by:

- Using progress bars (st.progress or st.bar_chart) for each emotion score.
- Showing a horizontal bar graph ranking emotions by probability.
- Example (conceptually):

joy:  0.81

pride:  0.65

love:  0.55