Model Monitoring and Drift Detection in ASR Systems

As automatic speech recognition (ASR) systems become increasingly integrated into real-world applications, ensuring sustained model performance post-deployment is critical. In this project, we implemented an ASR pipeline using Facebook's `wav2vec2-large-960h` model, integrated with a frontend search interface and Flask backend for querying transcribed audio. While this solution performs well in development settings, robust model monitoring is essential to maintain reliability in dynamic production environments such as on AWS.

Why Monitor ASR Models?

ASR systems are prone to model drift—changes in the statistical properties of input data that degrade model performance. We used Mozilla's Common Voice dataset (`cv-valid-dev.csv`), which includes audio samples with metadata such as age, gender, accent, and text. Evaluating the ASR system on this dataset revealed performance discrepancies across demographics:

- For `sample-000029.mp3`, the actual text was "are you gonna throw a rock", but the model transcribed it as "A GINTO THROW A ROCK", likely due to the speaker's accent (Canadian) and age (twenties).
- For `sample-000018.mp3`, the correct line "to nourish the falcon" was rendered as "TO ROURISH THE FALCON", suggesting confusion from background noise or phonetics.

Accents like US and England dominate the dataset (637 and 327 samples), but accents such as African (23), Philippines (6), and Southatlandtic (3) are underrepresented—correlating with higher error rates. This highlights the need for active drift detection to ensure inclusivity and accuracy across diverse speakers.

Proposed Monitoring Pipeline

To address these challenges, we propose a **multi-layered model monitoring pipeline** consisting of the following components:

1. **Data Logging**
   Each ASR request is logged with:
   - Audio metadata (duration, accent, age group, etc.)
   - Transcribed text (generated_text)
   - Actual text
   - User corrections (optional feedback loop)

   These logs serve as the foundation for all downstream monitoring.

2. **Performance Monitoring**
   For labeled data, we calculate **Word Error Rate (WER)** and **Character Error Rate (CER)** periodically. This is critical when we integrate newer datasets or real-world user input. Performance metrics are visualized in dashboards to detect performance regression early.

3. **Input Drift Detection**
   We track input features such as:
   - Accent distribution
   - Audio quality (SNR, volume range)
   - Duration and silence ratios

4. **Embedding-based Drift Detection**
   Using wav2vec2 embeddings, we cluster inputs and apply statistical tests (e.g., KL divergence) to detect subtle distribution changes over time.

5. **Feedback Loop & Alerts**
   User-reported errors or manual corrections can serve as soft labels. Integrating them into a feedback loop enables continuous model evaluation and potential fine-tuning. Threshold-based alerts notify developers of performance drops or abnormal input patterns.

Mitigating Drift

Upon detecting drift, the following remediation strategies can be employed:

- Fine-tune the model on new data (accent-specific or domain-specific)
- Use ensemble methods or fallback models for edge cases
- Augment the dataset with synthetic or collected data representing the drifted distribution

Conclusion

Monitoring ASR systems is a continuous process. For models like **wav2vec2**, exposed to diverse and evolving audio, tracking drift is essential. This pipeline ensures transparency, adaptability, and sustained performance—critical for deploying inclusive and trustworthy ASR in production.