Model Monitoring Pipeline for ASR Microservice: Tracking Model Drift

This essay proposes a comprehensive model monitoring pipeline for the repository's Wav2Vec2-based ASR microservice, focusing on drift detection mechanisms to ensure sustained performance.

Monitoring Objectives

The ASR system uses `cv-decode.py` to batch-transcribe Common Voice datasets, indexing results for later search. Since inference is not real-time like in interactive systems, model drift detection must prioritize transcription quality and accuracy over latency metrics. Processing speed and throughput become secondary metrics since batch jobs can run overnight without user impact.

Monitoring Flow:

Audio Input → Preprocessing → ASR Inference → Metrics Collection (Pre + Post Inference) → Human Review (sampled or interactive feedback) → Drift Detection → Alerts → Response Actions

Data Collection Strategy

To maintain transcription quality and detect model drift, the monitoring system should focus on indicators tied to transcription accuracy and input data characteristics.

Hence, the key metrics to track include:

- Input data characteristics, such as duration and sample rate distributions
- Audio spectral properties and quality markers (e.g. noise level, signal-to-noise ratio)
- Confidence scores and logit distribution profiles
- Transcription length and vocabulary usage trends

These metrics are collected at various stages of the pipeline and stored in a time-series database (e.g. Prometheus, InfluxDB) for historical analysis. Input characteristics and spectral features are captured during audio preprocessing in `_process_audio_file()`. Logit distributions and confidence scores are recorded by adding monitoring hooks to `_run_inference()`, while transcription patterns are derived after inference from the model's outputs and the decoded text.

Drift Detection Methods

Model drift can be detected by continuously monitoring the key metrics for shifts in their statistical distributions over time. Specifically:

- Confidence scores and logits: A drop in average confidence or a change in the distribution (e.g., more flat or skewed logits) may signal reduced model certainty or changes in input patterns.
- Transcription patterns: Increases in average transcription length, out-of-vocabulary terms, or inconsistent formatting can indicate that the model is encountering unfamiliar or degraded inputs.
- Audio features and quality markers: Significant shifts in spectral characteristics, noise levels, or input durations suggest that the nature of the input data is changing, which may challenge model robustness.

By comparing these metrics against historical baselines using statistical tests (e.g., KL divergence, Wasserstein distance) or time-series anomaly detection, we can identify when and how the model's behavior begins to deviate — enabling timely retraining or adaptation.

Human-in-the-Loop Quality Assurance

While automated metrics can detect statistical changes, they cannot validate whether transcriptions remain semantically accurate or contextually appropriate for search indexing. To cover that gap, the pipeline includes a

two-layer human validation process: expert reviews and real-time user feedback. Expert reviewers can evaluate 1-2% of daily transcriptions through stratified sampling, built into the `cv-decode.py` workflows. Meanwhile, users provide feedback directly through the search platform (assuming audio playback is available) using:

- Thumbs up/down ratings
- Error reporting forms
- Quick correction suggestions

Together, these feedback channels help fine-tune automated thresholds and uncover blind spots, feeding back into a continuous loop that sharpens drift detection over time.

Alerting and Response

The system triggers tiered alerts when metrics drift beyond configurable thresholds (e.g., 1.5 standard deviations (warning) or 2.5 (critical) from the baseline). Each alert includes diagnostics and recommended next steps. Depending on severity, the response may include automated reports or suggestions to initiate model retraining. All alerts and actions are logged to support continuous improvement in detection and response.