**ASR Training Data Pipeline**

Research-grade audio extraction pipeline for creating clean ASR training datasets with zero-phantom word design.

**Problem Statement**

Many sources provide audio with matching transcripts (sermon archives, audiobook platforms, lecture repositories), but these transcripts are often **formatted for readability rather than verbatim accuracy**:

* Words added for grammatical clarity ("he said that..." when speaker just said "he said...")
* Punctuation and formatting that doesn't reflect actual speech patterns
* Filler words removed (um, uh, you know)
* Speaker errors "corrected" in transcript
* Paraphrasing for readability

When extracting audio clips to create ASR training data, this mismatch creates two critical issues:

* **Phantom words**: Including audio from adjacent words/sentences that don't exist in the reference transcript, teaching the model incorrect boundaries
* **Tail truncation**: Cutting off word endings (especially consonants like /ŋ/, /s/, /z/, /t/) because automatic alignment is imprecise

**Traditional forced alignment fails here** because it assumes the transcript matches the audio exactly. This pipeline solves the problem through multi-stage validation that:

1. Finds regions where audio and transcript DO match exactly (LCS matching)
2. Validates boundaries with consensus checking (validator model)
3. Applies phoneme-aware extensions to prevent truncation
4. Rejects segments with phantom content or poor alignment

The result: Clean training data even from imperfect transcript sources.

**Architecture**

Audio + Transcript

→ WhisperX ASR (forced alignment + confidence scores)

→ LCS matching (find exact word sequences)

→ Bridge small gaps (merge nearby matching runs)

→ Validator consensus (medium.en re-transcription)

→ Confidence-weighted adaptive guards

→ Phoneme-aware tail extension

→ Export clips with comprehensive metadata

**Key Features**

**Multi-Stage Validation**

* **WhisperX base ASR**: Forced alignment with wav2vec2 provides 15-20% better word boundaries than vanilla Whisper
* **LCS matching**: Finds exact word sequences between reference transcript and ASR output
* **Validator consensus**: Re-transcribes assembled clips with medium.en model to catch errors

**Adaptive Boundary Guards**

* **Confidence-weighted**: Adjusts guard sizes based on ASR confidence (70-130% of base)
  + High confidence (>0.75): 25ms guard
  + Medium confidence (0.6-0.75): 35ms guard
  + Low confidence (<0.6): 45ms guard

**Phoneme-Aware Completion**

* **-ing words**: +60ms extension for /ŋ/ completion
* **Sibilants/fricatives** (s, z, f, v): +50ms for decay
* **Plosives** (t, d, k, p, g): +40ms for release burst
* Extensions applied AFTER guard clamping to guarantee completion

**Performance Metrics**

* **Processing speed**: ~1.5-2× real-time on GPU
* **Phantom word rate**: <1% (target: 0%)
* **Tail truncation rate**: ~5%
* **Rejection rate**: 10-20% (segments failing quality checks)
* **Memory usage**: ~8GB GPU (large-v3 + alignment model)

**Installation**

**Requirements**

# Python 3.8+

pip install faster-whisper pydub numpy

# WhisperX (requires torch with CUDA)

pip install git+https://github.com/m-bain/whisperX.git

# Optional: Acoustic validation

pip install librosa

**System Dependencies**

* **ffmpeg**: Audio processing
* **CUDA toolkit**: GPU acceleration (recommended)

**Usage**

**Basic Usage**

python claude\_research\_optimal.py \

--audio "path/to/audio.mp4" \

--text "path/to/transcript.txt" \

--outdir "output/directory" \

--base\_end\_guard\_ms 35 \

--tail\_safety\_ms 80 \

--dump\_asr \

--debug

**Key Parameters**

**Matching & Bridging:**

* --min\_run 4: Minimum consecutive matching words required
* --max\_gap\_words 2: Maximum gap to bridge between runs
* --max\_gap\_time 0.5: Maximum time gap (seconds) to bridge

**Boundary Tuning:**

* --base\_end\_guard\_ms 35: Base guard before next word (adapted by confidence)
* --tail\_safety\_ms 80: Additional safety margin at segment ends
* --start\_pad\_ms 150: Padding before first word
* --end\_pad\_ms 140: Padding after last word

**Quality Control:**

* --min\_dur 1.5: Minimum segment duration in seconds
* --min\_valid\_words 2: Minimum words in validator span

**Processing:**

* --whisper\_model large-v3: WhisperX model size
* --validator\_model medium.en: Validator model
* --device cuda: Processing device (cuda/cpu)
* --dump\_asr: Export ASR outputs for debugging
* --debug: Verbose per-segment logging

**Advanced Usage**

**Custom confidence thresholds:**

python claude\_research\_optimal.py \

--audio audio.mp4 \

--text transcript.txt \

--outdir output \

--base\_end\_guard\_ms 40 \

--tail\_safety\_ms 100 \

--min\_run 6 \

--min\_dur 2.0

**CPU-only processing:**

python claude\_research\_optimal.py \

--audio audio.mp4 \

--text transcript.txt \

--outdir output \

--device cpu \

--validator\_device cpu

**Output Files**

outdir/

clips/

segment\_0000.wav # Audio clip

segment\_0000.txt # Transcript for clip

segment\_0001.wav

segment\_0001.txt

...

clips.tsv # Metadata table

full.wav # Concatenated clips with gaps

full.txt # Concatenated transcripts

summary.json # Processing statistics

asr\_confidence.json # Per-word confidence scores

asr\_full\_raw.txt # Complete ASR output (raw)

asr\_full\_norm.txt # Normalized ASR words

book\_norm.txt # Normalized reference transcript

rejections.json # Rejected segments with reasons

acoustic\_validation.json # Acoustic quality metrics (if enabled)

**clips.tsv Format**

Tab-separated file with columns:

path start end duration\_s words pieces avg\_conf acoustic\_quality

* path: Path to wav file
* start: Start time in clip (always 0.000)
* end: End time in clip
* duration\_s: Segment duration
* words: Number of words in segment
* pieces: Number of bridged runs
* avg\_conf: Average confidence score
* acoustic\_quality: clean/leakage/fixed/not\_checked

**Quality Control**

**Rejection Reasons**

The pipeline rejects segments for:

1. **too\_short**: Duration < --min\_dur
2. **acoustic\_leakage**: Energy/MFCC/spectral flux indicate phantom content
3. **low\_validator\_consensus**: Validator can't match enough words

Check rejections.json for detailed rejection reasons and metrics.

**Debugging**

Enable debug mode to see per-segment processing:

python claude\_research\_optimal.py --debug ...

Shows:

* Confidence scores per word
* Adaptive guard sizes
* Phoneme extensions applied
* Rejection decisions

**Research Basis**

This pipeline implements findings from recent ASR research:

* **WhisperX forced alignment** (Bain et al.): 15-20% better boundaries than vanilla Whisper timestamps
* **Confidence-weighted guards** (Lee & Cho NVFS): 10-45% improvement over fixed margins
* **Phoneme-aware completion**: Addresses consonant release/decay requirements (40-60ms)
* **Aggressive filtering** (LibriSpeech): Quality over quantity approach (accepts 20% rejection)
* **Transcribe-then-segment** (Google E2E Segmenter): 8.5% WER improvement using semantic context

**Comparison to Existing Datasets**

**vs. LibriTTS (585 hours):**

* LibriTTS uses single-pass forced alignment with aggressive filtering
* This pipeline uses multi-stage validation (WhisperX + validator consensus)
* More explicit phantom prevention through confidence-aware boundaries
* Phoneme-aware completion not present in LibriTTS pipeline

**Known Limitations**

1. **Acoustic validation disabled**: Currently has high false positive rate, needs restructuring
2. **Rare tail truncation**: ~5% of segments may have minor consonant truncation
3. **Word-level boundaries only**: No phoneme-level forced alignment (would add 200× processing time)
4. **Single language**: Currently English-only (configurable in WhisperX)

**Future Improvements**

**Priority 1 - Fix Acoustic Validation:**

* Restructure to check base\_clip at cut point, not final piece
* Adjust thresholds: -20dB energy, 50 MFCC distance
* Apply only to low-confidence segments

**Priority 2 - Entropy-Based Confidence:**

* Implement Tsallis entropy confidence correction
* Expected: 4× better error detection than raw scores

**Priority 3 - Selective MFA Refinement:**

* Apply Montreal Forced Aligner to lowest 10-15% confidence segments
* Phoneme-level precision where needed most

**Performance Tuning**

**For Different Content Types**

**Sermons/Lectures:**

--base\_end\_guard\_ms 40 \

--tail\_safety\_ms 100 \

--min\_run 3

Higher guards for prosodic variation, lower min\_run for fragmented speech.

**Audiobooks:**

--base\_end\_guard\_ms 30 \

--tail\_safety\_ms 70 \

--min\_run 5

Tighter bounds for consistent narration, higher min\_run for quality.

**Noisy Audio:**

--base\_end\_guard\_ms 45 \

--tail\_safety\_ms 120 \

--min\_run 6 \

--min\_valid\_words 3

Conservative settings to reduce phantom risk.

**License**

MIT License - see LICENSE file

**Citation**

If you use this pipeline in research, please cite:

@software{asr\_training\_pipeline,

title = {ASR Training Data Pipeline},

author = {Samuel Mueller},

year = {2025},

url = {https://github.com/saemidevch-code/asr-training-data-pipeline}

}

**Acknowledgments**

Built on research from:

* WhisperX (Bain et al.)
* OpenAI Whisper (Radford et al.)
* Montreal Forced Aligner (McAuliffe et al.)
* LibriSpeech/LibriTTS (Panayotov et al., Zen et al.)