LFA: Long-speech Forced Alignment with Phoneme-based Reference Assignment

Anonymous submission to Interspeech 2026

Abstract

Forced alignment of short utterances is well-solved, but aligning long speech recordings against reference transcripts remains challenging in practical applications. We propose LFA, a modular divide-and-conquer approach that decomposes long recordings into short chunks via ASR, then assigns reference transcript portions to each chunk using word-level dynamic time warping with edit distance on phonemized word representations. The modular design allows us to use any alignment method on the resulting chunks. We evaluate LFA on noise-augmented TIMIT with file lengths from 5 min to 9.7 h. On these recordings, MFA fails or degrades severely, and NeMo baselines degrade with length. Depending on the per-chunk alignment backend, LFA achieves up to 52 ms median onset error or 68.4% onset accuracy@100 ms on a single 9.7-hour file, maintaining stable quality across all durations.

Index Terms: forced alignment, automatic speech recognition, dynamic time warping, CTC alignment, long speech, reference transcript assignment

1. Introduction

Forced alignment (FA) is the task of determining precise time boundaries for words or phonemes given a speech recording and a transcript. It is fundamental to speech science, corpus linguistics, and language documentation. While FA for short, clean utterances is well-solved, e.g., with the Montreal Forced Aligner (MFA) [1] achieving 97.4% word-level accuracy at 100 ms tolerance on TIMIT [2], alignment of long speech recordings (minutes to hours) remains challenging in practical applications. Early work addressed long recordings using anchor-point methods based on Hidden Markov Models (HMM) [3, 4], but to the best of our knowledge, no prior work has systematically varied recording length to measure how alignment accuracy scales.

Parliamentary corpora span hours per session [5, 6, 7], and oral history and language documentation archives regularly contain multi-hour recordings. These applications require word-level alignment of recorded speech against existing reference transcripts prepared by experts and following specific conventions, rather than using ASR output directly.

Beyond alignment itself, reliable long-speech FA enables large-scale data mining for ASR training: vast quantities of paired speech and text exist as audiobooks, lectures, and parliamentary proceedings, and adding word-level time annotations would turn these into fine-grained training data [8].

Existing FA systems face two scalability problems on long recordings. First, CTC-based exact Viterbi decoding requires dynamic programming over a trellis whose size scales with $O(T \times L)$ (acoustic frames $\times$ label sequence length), which can become impractical for hour-long recordings, motivating linear-memory CTC formulations [9], linear-memory edit distance on decoded phone sequences [10], and divide-and-conquer approaches that keep T small by chunking [3, 4, 11]. Second, HMM-based aligners such as MFA can fail entirely on noisy recordings. In our experiments, standalone MFA produces no useful output on noise-augmented speech.

We address this through a modular divide-and-conquer approach called LFA: we use ASR to decompose the long-speech alignment problem into short, independently solvable chunks, then align each chunk with any FA method of choice. Our contributions are:

1. *Divide-and-conquer reference assignment via phoneme DTW*: A method that converts both ASR hypotheses and reference words to phoneme sequences via grapheme-to-phoneme (G2P) conversion, then uses dynamic time warping (DTW) with the Damerau-Levenshtein distance on phonemized word pairs to partition the reference across audio chunks. This tolerates ASR errors because phonemically similar words yield low cost even when text matching fails; a cumulative assignment strategy further handles word splits and merges (Section 3.3, Figure 1).
2. *Controlled scaling evaluation on noisy speech*: We construct noise-augmented TIMIT length variants spanning 5 min to 9.7 h with identical speech content, providing the first controlled experiment systematically varying recording length on challenging audio.
3. *Flexible use of FA methods*: We show that our decomposition enables us to use any forced alignment method to process recordings of arbitrary length. We evaluate LFA with three different backends — HMM-based (MFA), wav2vec2 CTC, and NeMo Conformer. All variants maintain stable accuracy from 5 min to 9.7 h, with quality determined by the backend's short-segment performance rather than recording duration.

2. Related Work

2.1. Forced Alignment Systems

The Montreal Forced Aligner (MFA) [1] uses Kaldi-based GMM-HMM acoustic models with pronunciation dictionaries and remains one of the most widely used FA tools in phonetics research [2]. However, MFA requires language-specific acoustic models and pronunciation dictionaries, and its behavior on long or noisy recordings has not been systematically studied.

The NeMo toolkit provides two alignment tools. NeMo Forced Aligner (NFA) [13] applies Viterbi decoding to CTC log-probabilities; for long recordings, the model forward pass may need to be chunked to fit memory. NeMo's CTC segmentation [14] uses dynamic programming with optional zero-cost

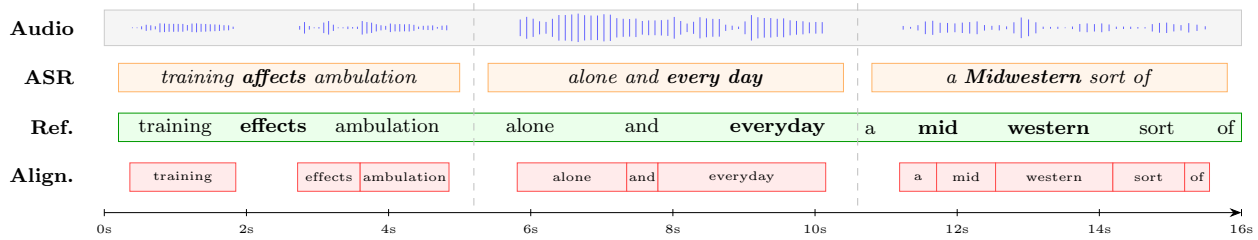

Figure 1: *ASR error types from Buckeye [12]: substitution ("affects" for "effects"; c = 0.17), word split ("every day" for "everyday"), and word merge ("Midwestern" for "mid western"). Word-level DTW assigns reference words despite these errors via low phonemic cost (substitutions) and cumulative assignment (splits/merges). Bottom row: final aligned word boundaries.*

blank transitions, scaling to longer recordings but with potentially reduced accuracy.

WhisperX [15] is not a forced alignment system in the strict sense but is closely related. It combines Whisper ASR with VAD-based segmentation and wav2vec2 alignment. However, WhisperX aligns its own ASR output rather than an external reference transcript, making it primarily an enhanced ASR system with word timestamps rather than a tool for aligning a given reference transcript to audio. Our system builds on the VAD chunking idea but addresses this latter problem.

Aeneas [16] aligns text fragments against audio using TTS synthesis and MFCC-level DTW, handling long recordings at the acoustic level without ASR. It is widely deployed but operates on frame-level features rather than word-level alignment.

Moreno et al. [3] proposed a recursive algorithm that uses ASR to identify anchor points and recursively aligns segments between anchors. Hazen [4] adopted a similar anchor-based approach but replaced the recursion with a fixed three-stage pipeline, using an FST-based pseudo-forced alignment that tolerates errors in approximate transcripts. SailAlign [11] combined iterative ASR with text matching to find reliable anchors, then performed per-chunk FA. DSAlign [17] used DeepSpeech ASR with character-level Smith-Waterman local alignment in a recursive divide-and-conquer strategy: longer ASR fragments are matched first to anchor the alignment, and shorter fragments are resolved recursively within the remaining intervals. Candidate reference windows are ranked by shared character 3-gram count before full Smith-Waterman scoring. DSAlign is the closest predecessor to our work: it shares the same VAD–ASR–text-alignment pipeline structure, but uses orthographic character matching and depends on the discontinued Mozilla DeepSpeech. Bordel et al. [10] aligned 3-hour bilingual parliamentary recordings by running a language-independent HMM phone decoder, then aligning the decoded phone sequence against the reference using Hirschberg's linear-memory edit distance at the individual phone level — explicitly avoiding language models or lexical resources to handle code-switching between Basque and Spanish. Álvarez et al. [18] extended this approach for automatic subtitling, showing that predefined non-binary phone-relatedness matrices outperform binary matching. Both operate at the individual phone level, producing alignment matrices orders of magnitude larger than our word-level approach. In the parliamentary speech domain, Ljubešić et al. [7] aligned over 5,000 hours of Slavic parliamentary speech using word-histogram sliding windows with Levenshtein distance for text-to-text matching, achieving a 74% word yield for Croatian, with the remaining 26% of words unmatched due to ASR errors. Unlike these phone-level predecessors, LFA operates at the word level and separates reference assignment from per-chunk alignment, allowing any aligner on the resulting short chunks.

Rousso et al. [2] compared MFA, WhisperX, and MMS on TIMIT and Buckeye, finding that MFA is superior on short utterances. Deng et al. [19] provided a large-scale FA benchmark including long conversational audio, confirming that long-speech FA remains challenging.

2.2. CTC-based Alignment

CTC [20] provides a natural framework for FA through its frame-level output probabilities. Doras et al. [9] developed a linear-memory CTC alignment algorithm by adapting Hirschberg-style divide-and-conquer to the CTC Viterbi matrix, reducing memory from $O(T \times L)$ to $O(T + L)$ at the cost of additional computation. This solves the memory bottleneck for single-file CTC alignment, but requires the full text to be aligned in one pass. Our approach instead decomposes the problem via ASR-guided chunking, then uses CTC on each short chunk.

Recent work has improved CTC alignment accuracy. Huang et al. [21] showed that label priors reduce CTC's peaky output distributions, improving boundary accuracy by 12–40%. Meta's MMS [22] provides multilingual wav2vec2 models across 1,130 languages with a forced alignment tool. Charsiu [23] offers wav2vec2-based phone-level FA. Mu et al. [24] reformulated FA as a slot-filling task using a Qwen3-based LLM with an audio encoder, predicting discrete timestamps non-autoregressively. Although titled "long-form," their evaluation concatenates short utterances up to 5 min with 80 ms timestamp resolution, whereas our method at 20 ms CTC frame resolution (wav2vec2 FA) works on recordings up to 9.7 h. Rehman et al. [25] presented a CTC-based multilingual aligner focused on phoneme-level temporal boundaries.

2.3. Sequence Alignment with Dynamic Programming

Dynamic time warping (DTW) [26] is a classic algorithm for aligning temporal sequences, extensively used in music information retrieval [27] and speech processing. For text-to-audio alignment, González-Carrasco et al. [28] used word-level Needleman-Wunsch sequence alignment with orthographic word comparison for live subtitle synchronization, later extended by Masiello-Ruiz et al. [29] with adaptive selection among NW, Levenshtein, and Smith-Waterman algorithms. Our approach differs by (1) using DTW rather than alignment based on the edit distance, so that every transition maps an ASR word to a reference word without free gap operations, and (2) operat-

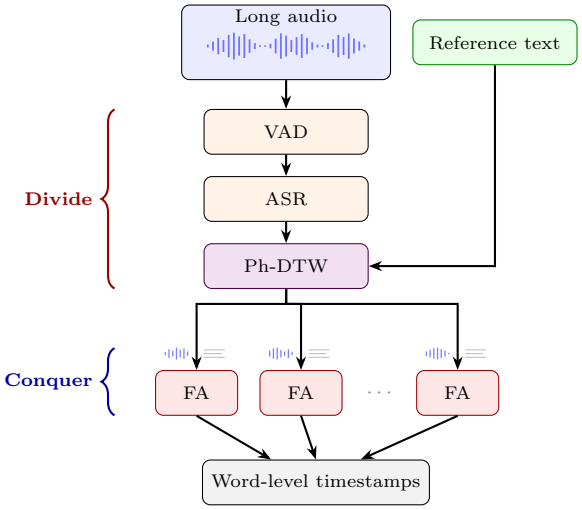

Figure 2: *LFA divide-and-conquer approach. Divide: split audio into chunks via VAD, transcribe each chunk with ASR, assign reference transcript portions to chunks via phoneme DTW. Conquer: Align each chunk independently using forced alignment.*

ing at the word level with a phoneme edit distance cost, rather than aligning individual phone sequences. Jin et al. [30] independently used DTW with phoneme-based costs for on-device forced alignment, but at the frame-to-phoneme level for short utterances rather than word-level reference assignment.

While phoneme similarity has been used at the phone level for long speech alignment [18, 10], our approach operates at the word level: we apply DTW with phoneme-based edit distance on ASR word hypotheses against reference words. This yields an $N_{\text{ASR}} \times K$ cost matrix (ASR words $\times$ reference words), which is orders of magnitude smaller than the $T \times L$ CTC frame-label matrix or phone-level alignment matrices, making it tractable even for multi-hour recordings. The cumulative assignment strategy (Section 3.3.4) then maps phoneme DTW paths to chunk-level reference partitions.

3. Method

Our approach, called LFA, follows a divide-and-conquer strategy (Figure 2): divide the long recording into short chunks using VAD, transcribe each chunk with ASR, assign reference transcript portions to each chunk via phoneme DTW, and solve each chunk independently with forced alignment. The four stages are: (1) VAD, (2) ASR, (3) Phoneme DTW Reference Assignment, and (4) Per-Chunk Forced Alignment.

3.1. Voice Activity Detection

The input audio is segmented into speech regions using Pyannote VAD [31] with onset threshold 0.2 and offset threshold 0.05 (low thresholds to favor recall over precision, since missed speech is more harmful than false alarms for alignment). Long speech regions are further split into chunks of at most C seconds (default $C = 10$) at detected pause boundaries. This produces a sequence of audio chunks $\mathbf{S} = (s_1, \ldots, s_N)$ with corresponding time spans.

3.2. Automatic Speech Recognition

Each chunk s_i is transcribed using Faster Whisper [32, 33] (large-v3). The ASR produces word-level hypotheses $\hat{\mathbf{w}}_i = (\hat{w}_{i,1}, \ldots, \hat{w}_{i,M_i})$ for each chunk with approximate timestamps.

3.3. Phoneme DTW Reference Assignment

The main problem is assigning the correct portion of the reference transcript $\mathbf{R} = (r_1, \ldots, r_K)$ to each audio chunk. The simplest approach would use orthographic matching, but this may fail when ASR errors cause divergence from the reference. We exploit that phonemic forms often differ less than orthographic forms under noisy ASR conditions by comparing words through their phonemic representations (e.g., homophones such as "through"/"threw", both /θruː/).

3.3.1. Grapheme-to-Phoneme Conversion

Both ASR and reference words are converted to IPA phoneme strings using Epitran [34], a rule-based G2P system supporting 61 languages. Rule-based G2P can produce errors on irregular words, but the edit-distance cost function (Eq. 1) tolerates such errors: even imperfect phonemizations produce lower costs for correct word pairs than for unrelated words. For each word w, we obtain its phoneme string $\phi(w)$ — for example, $\phi(\text{"effects"}) = \text{"ifekts"}$.

3.3.2. Word-Level Cost Matrix

We construct an $N_{\text{ASR}} \times K$ cost matrix where N_{ASR} is the total number of ASR words across all chunks and K is the number of reference words. The cost for word pair (i, j) uses the Damerau-Levenshtein (DL) similarity [35] on their phoneme strings:

$$c(i, j) = 1 - \text{DL}_{\text{sim}}(\phi(\hat{w}_i),\ \phi(r_j)) \tag{1}$$

where DL_{sim} returns normalized similarity (0 = completely different, 1 = identical), so $c = 0$ for identical words and $c = 1$ for completely different ones. DL extends Levenshtein distance with transpositions, additionally handling phoneme metathesis. By comparing phonemic forms rather than orthography, homophones and near-homophones receive low cost despite orthographic differences. Figure 3 shows a word-level cost matrix example.

The matrix dimensions are $N_{\text{ASR}} \times K$ words, not $T \times L$ frames as in CTC alignment. For a 2-hour recording with $\sim$11,000 words per side, this is an 11K$\times$11K word matrix, i.e., orders of magnitude smaller than the millions of frames that make full-file CTC alignment infeasible.

3.3.3. DTW Alignment

We use DTW [26] to find the optimal alignment path through the cost matrix. The DTW recurrence is:

$$D(i, j) = c(i, j) + \min\left(D(i-1, j-1),\ D(i-1, j),\ D(i, j-1)\right) \tag{2}$$

We use DTW rather than edit-distance-based alignment (Needleman-Wunsch). In DTW, all three transitions (diagonal, vertical, horizontal) add the local cost $c(i, j)$: there are no free gap operations. This means every step on the path maps an ASR word to a reference word. Diagonal moves give 1:1 alignment; vertical moves handle many-to-one cases where ASR splits a word ("every day" $\rightarrow$ "everyday"); horizontal moves handle

Algorithm 1 Phoneme DTW Reference Assignment

Require: ASR words $\hat{\mathbf{w}}$ with chunk IDs, reference words $\mathbf{R}$, phonemizer ϕ
Ensure: Reference word assignment per chunk

Compute cost $c(i,j) = 1 - \mathrm{DL}_{\mathrm{sim}}(\phi(\hat{w}_i), \phi(r_j))$ for all i, j
Initialize $D(0,0) = 0$; all other $D = \infty$
for $i = 1$ to N_{ASR} **do**
  for $j = 1$ to K **do**
    $D(i,j) \leftarrow c(i,j) + \min(D(i-1,j-1), D(i-1,j), D(i,j-1))$
  end for
end for
$i^* \leftarrow \arg\min_{1 \le i \le N_{\mathrm{ASR}}} D(i,K)$
Backtrack from (i^*, K) to $(1,1)$ to obtain path π
for each reference word r_j **do**
  Collect aligned ASR positions $\mathcal{A}(j) = \{(i, 1-c(i,j)) : (i,j) \in \pi\}$
  Assign r_j to $\arg\max_k \sum_{(i,s)\in\mathcal{A}(j),\,\mathrm{chunk}(i)=k} s$
end for

one-to-many cases where ASR merges words ("Midwestern" → "mid western").

We use phoneme DTW with open-end termination: the path must start at $(1,1)$ but may terminate at any row in the last column, i.e., at (i^*, K) for any $i^* \le N_{\mathrm{ASR}}$. This ensures that all K reference words participate in the alignment while allowing surplus ASR words at the end (e.g., from hallucinations or noise-induced insertions) to remain unmatched. For scalability, we apply a Sakoe-Chiba band constraint [26] with width $B = 0.05 \cdot \max(N_{\mathrm{ASR}}, K)$, restricting computation to a diagonal band and reducing memory from $O(N_{\mathrm{ASR}} \times K)$ to $O(N_{\mathrm{ASR}} \times B)$. Algorithm 1 shows the procedure (banding omitted for clarity).

3.3.4. Cumulative Assignment

The DTW path aligns each reference word to one or more ASR words. Since each ASR word belongs to a known chunk, we assign each reference word to the chunk whose aligned ASR words contribute the highest cumulative similarity. Formally, for reference word r_j, let $\mathcal{A}(j) = \{(i, s_i) : (i,j) \in \pi\}$ be its aligned ASR positions with similarities $s_i = 1 - c(i,j)$. We assign r_j to chunk $k^* = \arg\max_k \sum_{(i,s_i)\in\mathcal{A}(j),\,\mathrm{chunk}(i)=k} s_i$. This handles word splits and merges: when ASR splits a reference word across chunks, cumulative scoring assigns it to the chunk with the stronger match. When ASR is highly accurate, most reference words align unambiguously to a single chunk but under noisy conditions, the cumulative scoring can resolve ambiguous cross-boundary alignments.

3.4. Per-Chunk Forced Alignment

Each chunk, with its DTW-assigned reference words, is aligned independently using an FA method. Any aligner capable of processing short chunks (typically $\lesssim 20$ s) can be used. We evaluate three options:

CTC alignment with wav2vec2 [36, 37]: The audio is passed through the encoder to obtain frame-level CTC log-probabilities ($\sim$20 ms per frame). Reference words are decomposed into characters, and Viterbi decoding finds the optimal frame-to-label assignment. Character timestamps are merged into word boundaries. We use

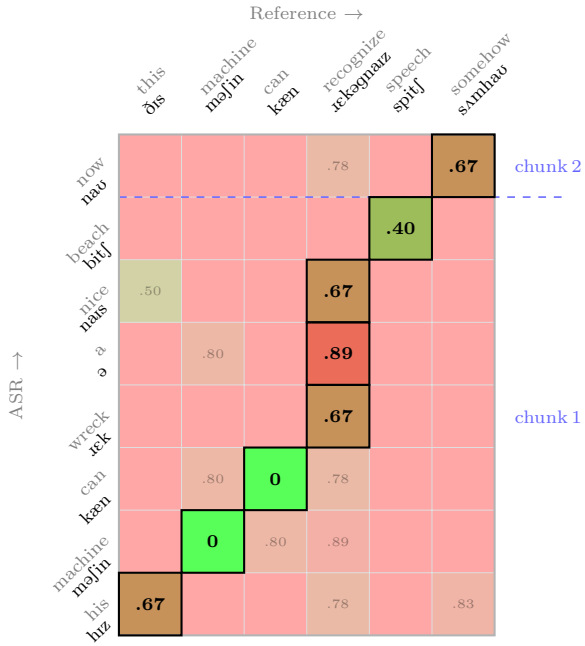

Figure 3: *DTW cost matrix. Cell color encodes distance on IPA bold values mark the optimal path. Dashed lines: Chunk boundaries after ASR step.*

`wav2vec2-xls-r-1b-english`[1] (1B parameters) as default; the CTC model choice is the single largest factor affecting alignment quality (Section 4.5).

MFA alignment [1]: Each chunk is aligned using MFA's GMM-HMM acoustic model with a pronunciation dictionary. MFA provides phoneme-level HMM duration modeling, which often yields finer boundaries than CTC's frame-level resolution.

NeMo Conformer alignment: Each chunk is aligned using NeMo's Conformer CTC model (`stt_en_conformer_ctc_large`) with Viterbi decoding. This provides an additional CTC-based baseline that differs from wav2vec2 in model architecture and training data.

For CTC alignment, several post-processing steps improve quality. The wav2vec2 feature extractor expects zero-mean, unit-variance input. We apply this normalization per-chunk rather than globally, yielding more consistent CTC probabilities across chunks with varying volume levels. CTC alignment tends to extend word boundaries into silence regions at chunk edges. We detect silence by thresholding the RMS energy at chunk boundaries and adjust the first/last word boundaries inward to the speech onset/offset. We prepend a blank column to the CTC trellis before Viterbi decoding, allowing the alignment path to remain in silence before the first token rather than absorbing preceding frames into the first word, and extend word onset boundaries backward through preceding low-blank-probability frames. Each segment is also extended by 0.5 s of acoustic context at its left boundary before alignment. For MFA alignment, we merge tokens that MFA splits at apostrophes and hyphens back to match the reference tokenization.

[1] `https://huggingface.co/jonatasgrosman/wav2vec2-xls-r-1b-english`

Table 1: *TIMIT-LN evaluation datasets derived from TIMIT. All variants contain identical audio content; file length differs.*

Dataset	Files	Words	Mean (min)	Range (min)
5 min	119	457	4.9	1.1–5.2
10 min	59	921	9.9	8.3–10.4
20 min	30	1,812	19.4	6.8–20.8
35 min	17	3,197	34.2	26.2–35.1
1 h	10	5,436	58.3	45.8–60.4
2 h	5	10,871	116.5	104.7–120.2
full	1	54,357	582.0	582.0

4. Experiments

4.1. Datasets

We derive a long and noisy benchmark dataset from the TIMIT corpus [38] (read American English, with manually verified time annotations) which we will refer to as TIMIT-LN. This dataset simulates real-world scenarios such as long parliamentary sessions, lecture recordings, oral history interviews or speeches, where word-precise alignment is necessary. We reorganize TIMIT corpus into acoustically challenging length variants. All 6,300 TIMIT utterances (train + test, 630 speakers) are shuffled so that each file contains a mix of speakers, then concatenated with three per-utterance degradations: room reverberation (RT60 0.3–0.6 s) using pyroomacoustics [39], random gain modulation ($\pm$4–6 dB), and random inter-utterance pauses (0–5 s). Mixed additive noise from the MUSAN corpus [40] is then applied at a global 15 dB SNR (computed over the entire file; representing moderate noise typical of meeting or field recording environments) with three components: babble speech ($\sim$60%), environmental noise ($\sim$25%), and white Gaussian noise ($\sim$15%), with per-file energy proportions drawn from a Dirichlet distribution. Ground truth annotations are shifted by cumulative duration offsets. This procedure yields identical speech content and word sequences across all variants; differences in accuracy are attributable solely to recording length.

Table 1 summarizes the seven evaluation datasets in TIMIT-LN, spanning 5 min to 9.7 h. The longest variant concatenates all 6,300 utterances into a single file.

TIMIT-LN is a controlled benchmark dataset under realistic acoustic conditions: at short durations (5 min), most FA systems still perform adequately despite noise; as file length increases through 1 h and 2 h, systems are increasingly stressed by memory growth, error propagation, and the challenge of aligning tens of thousands of words in a single pass.

Since all variants contain the same 54,357 words, differences in accuracy directly measure each system's sensitivity to recording length.

We build on TIMIT rather than using naturally noisy corpora for two reasons. First, TIMIT provides accurate manually verified word-level time annotations, whereas alternatives have significant annotation limitations: Buckeye [12] leaves interviewer speech unannotated, and AMI [41] annotations were themselves generated by forced alignment and contain frequent disfluencies (filled pauses, hesitations). Second, constructing noise-augmented TIMIT variants avoids the cost of manually annotating 10 h of naturally noisy speech while providing a controlled variable (audio length) that cannot be easily isolated in existing corpora.

4.2. Forced Alignment Systems

We compare our approach against three widely-used baselines:

Montreal Forced Aligner (MFA) [1]: MFA v2.2 with pretrained `english_mfa` acoustic model, pronunciation dictionary, G2P for out-of-vocabulary (OOV) handling, and beam width 1000 (increased to 4000 on failure). MFA processes each file as a single unit without internal chunking. On our noisy multi-speaker datasets, MFA in its default mode fails to produce any alignments across all beam widths tested. In single-speaker mode, MFA produces output but with rapidly degrading accuracy (on@100 drops from 51% at 5 min to 32% at 10 min, to effectively unusable 5.5% at 20 min).

NeMo CTC Segmentation [14]: CTC-based forced alignment with QuartzNet (`stt_en_quartznet15x5`) and Conformer (`stt_en_conformer_ctc_large`). Uses dynamic programming over CTC log-probabilities where blank transitions between words have zero cost, finding the most likely segmentation boundaries.

NeMo Forced Aligner (NFA) [13]: CTC-based forced alignment with Citrinet (`stt_en_citrinet_1024`) and Conformer models. Uses Viterbi decoding (also dynamic programming, but over the full CTC state space including blank probabilities). For long recordings we chunk audio into 10 min segments for the model forward pass.

LFA (our approach): Pyannote VAD[2] (onset=0.2, offset=0.05), Faster Whisper large-v3[3], open-end DTW with cumulative assignment (Epitran G2P, Damerau-Levenshtein phoneme distance), and per-chunk forced alignment with chunk size C=10 s. We evaluate with three per-chunk aligners: MFA, wav2vec2 CTC (`xlsr-1b-en`, 1B parameters), and NeMo Conformer (Conformer).

4.3. Evaluation Metrics

Following standard practice [13, 2], we report mean and median absolute onset and offset error in milliseconds, and accuracy at thresholds $\tau \in \{25, 50, 100, 200\}$ ms (on@τ for onset, off@τ for offset), defined as the fraction of reference words with error $\leq \tau$.

4.4. Results

Figure 4 and Table 2 show how alignment accuracy scales with recording length on TIMIT-LN variants.

Standalone MFA fails on noisy audio: in default mode, it produces no usable output at any beam width. In single-speaker mode, accuracy degrades from 51% on@100 at 5 min to 32% at 10 min and 5.5% at 20 min. Among CTC-based baselines, NFA Conformer and NFA Citrinet range from 38–50% on@100, with NFA Citrinet maintaining 50.3% at 5 min but dropping to 42.9% at full; NFA Conformer shows a non-monotonic dip at 1 h (38.2%) driven by catastrophic drift in a subset of files (3 of 10 files exhibit onset means above 10 s), partially recovering at 2 h (44.5%), though still below its 35 min level (47.4%). CTC-Seg Conformer performs similarly (48% at 5 min, 41% at full). CTC-Seg QuartzNet degrades catastrophically from 16.4% at 5 min to 1.1% at full, as its weaker acoustic model accumulates alignment drift over long recordings.

In contrast, all three LFA variants maintain stable accuracy across the full range from 5 min to 9.7 h. LFA with wav2vec2

[2] `https://huggingface.co/pyannote/voice-activity-detection`

[3] `https://huggingface.co/openai/whisper-large-v3`

Table 2: *Onset accuracy@100 ms (%) across noisy TIMIT-LN length variants. All datasets contain the same 54,357 words.*

	On@100 ms (%) by dataset						
System	**5 min**	**10 min**	**20 min**	**35 min**	**1 h**	**2 h**	**full**
Baselines							
CTC-Seg (QuartzNet)	16.4	12.1	10.4	4.4	7.2	0.6	1.1
CTC-Seg (Conformer)	48.0	47.1	44.6	43.6	43.3	40.7	40.7
NFA (Citrinet)	50.3	50.4	50.0	49.4	47.9	46.7	42.9
NFA (Conformer)	49.4	49.0	46.9	47.4	38.2	44.5	45.3
MFA[†]	51.1	31.5	5.5	1.8		—	
LFA (ours)							
LFA (NeMo Conf., C=10)	45.4	45.6	45.5	44.7	45.1	44.4	46.5
LFA (MFA, C=10)	63.1	64.0	64.1	62.9	63.2	62.7	63.5
LFA (wav2vec2, C=10)	**67.7**	**68.9**	**68.9**	**68.1**	**66.8**	**65.6**	**68.4**

[†]Single-speaker mode; default mode produces no output. At 5 min, MFA produces matching word counts on only 9 of 119 files (3,801 words); accuracy is computed over these files only, not the full 54,357 words. No results beyond 35 min.

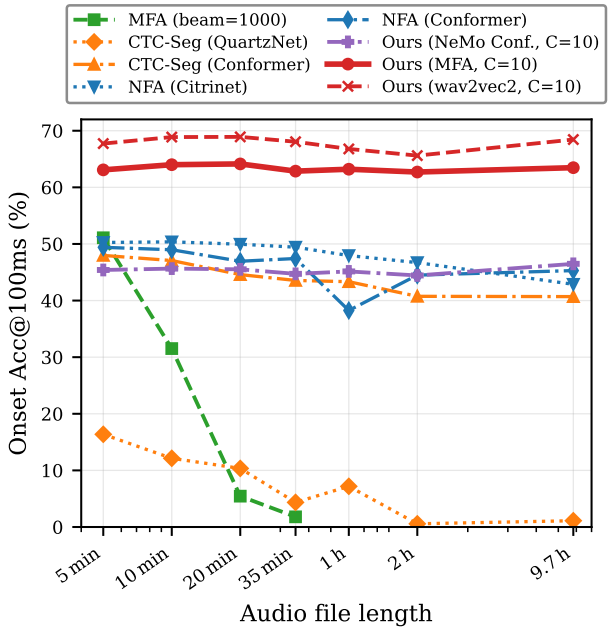

Figure 4: *Onset accuracy@100 ms vs. file length (log scale) on TIMIT-LN. LFA configurations maintain stable accuracy from 5 min to 9.7 h, while baselines generally decline at longer durations.*

achieves 65.6–68.9% on@100 (median onset 62–66 ms), LFA with MFA achieves 62.7–64.1% (median onset 52–54 ms), and LFA with NeMo achieves 44.4–46.5%. There is no scale degradation: LFA wav2vec2 scores 68.4% on the 9.7-hour file, within 0.7 pp of its 5-minute score (67.7%).

Table 3 provides a detailed breakdown on *TIMIT-LN-full* (1 file, 9.7 h). LFA with MFA per-chunk alignment achieves the best median onset (52 ms) with 63.5% on@100. LFA with wav2vec2 achieves the best on@100 (68.4%) with slightly higher median onset (62 ms). LFA NeMo exceeds standalone NFA Conformer on on@100 (46.5% vs. 45.3%) and achieves lower mean onset error (504 vs. 5,797 ms), confirming that phoneme DTW assignment improves alignment stability regardless of the per-chunk aligner.

4.5. Ablation Studies

We ablate two design choices: maximum chunk size (Table 4) and CTC alignment model (Table 5).

Table 4 shows the effect of maximum chunk size on *TIMIT-LN-1h* and *TIMIT-LN-2h*. All three aligners perform best with C=10, with the largest drop for wav2vec2 (-2.6 pp on@100 from C=10 to C=20 at 1 h) and smallest for NeMo (-0.7 pp at 1 h). The effect is moderate across all aligners ($\leq$2.6 pp), indicating low sensitivity to chunk size within the 10–20 s range. We use C=10 as the default for all experiments.

Table 5 compares four wav2vec2 CTC models at C=10 on *TIMIT-LN-2h*. The 1B-parameter xlsr-1b-en achieves 65.6% on@100 with 66 ms median onset, outperforming xlsr-53-en (300M) by 5.5 pp. Among 300M-parameter models, cross-lingually pretrained xlsr-53 outperforms English-only wav2vec2-960h (60.1% vs. 55.9% on@100). We use xlsr-1b-en as the default for all other experiments.

Table 6 shows per-stage timing for LFA with wav2vec2 alignment. The real-time factor (RTF) is approximately 0.054 on the full 9.7-hour file. On the full file, ASR dominates the runtime (48%), followed by alignment (38%) and DTW including G2P conversion (11%). The DTW reference assignment step adds moderate overhead, with G2P conversion accounting for roughly two-thirds of that time.

5. Discussion

The core idea behind LFA is that ASR, despite being imperfect, provides enough information to decompose the long-speech alignment problem into locally solvable subproblems. Each short chunk can then be aligned with standard CTC-based or HMM-based methods, which are typically more stable on short segments than on very long recordings. Per-chunk alignment quality is primarily determined by the aligner's short-segment performance, while any degradation on long files tends to arise from boundary effects and transcript-to-chunk assignment errors.

We chose DTW over Needleman-Wunsch or Smith-Waterman because all three DTW transitions (diagonal, horizontal, vertical) consume the local cell cost — there are no free gap operations. Every path position pays for mismatches, and ASR word splits/merges naturally correspond to many-to-one and one-to-many moves (cf. Figure 1). In Needleman-Wunsch,

Table 3: *Detailed results on TIMIT-LN-full (1 file, 9 h 42 min, 54,357 words). CTC-Seg QuartzNet and MFA omitted.*

	Onset (ms)		Offset (ms)		On@τ				Off@τ		
System	Mean	Med	Mean	Med	@25	@50	@100	@200	@50	@100	@200
Baselines (standalone)											
CTC-Seg (Conformer)	546.9	119	512.5	111	7.1	16.7	40.7	71.7	17.6	44.2	78.3
NFA (Citrinet)	6,500	117	6,490	87	10.5	21.4	42.9	74.0	31.1	55.0	75.9
NFA (Conformer)	5,797	108	5,789	68	7.9	18.1	45.3	80.5	40.1	61.5	78.3
LFA (ours)											
LFA (MFA, C=10)	738.9	**52**	746.8	**58**	**32.9**	**49.3**	63.5	70.9	**46.2**	62.0	69.7
LFA (wav2vec2, C=10)	673.5	62	693.2	61	17.3	39.5	**68.4**	74.7	40.0	**69.8**	74.2
LFA (NeMo Conf., C=10)	**504.4**	106	**491.9**	64	8.3	18.7	46.5	**82.7**	42.3	63.4	**80.0**

Table 4: *Effect of maximum chunk size C on onset accuracy (%) on TIMIT-LN-1h and TIMIT-LN-2h.*

		1 h		2 h	
Aligner	C	**@50**	**@100**	**@50**	**@100**
MFA	10	**48.5**	**63.2**	**48.2**	**62.7**
	15	47.2	61.3	47.0	61.0
	20	47.5	61.8	47.0	61.1
wav2vec2	10	**37.7**	**66.8**	**36.8**	**65.6**
	15	36.5	64.7	36.1	64.4
	20	36.0	64.2	35.4	63.6
NeMo Conf.	10	**17.6**	**45.1**	**16.8**	**44.4**
	15	16.9	44.3	16.4	43.3
	20	17.0	44.4	16.2	43.0

Table 5: *CTC alignment model comparison on TIMIT-LN-2h (C=10).*

		Onset (ms)		On@τ		
CTC Model	**Params**	**Med**	**Mean**	**@25**	**@50**	**@100**
xlsr-1b-en	1B	**66**	**690**	**16.1**	**36.8**	**65.6**
xlsr-53-en	300M	79	723	12.9	29.0	60.1
fb-960h (base)	95M	83	735	14.0	29.8	57.2
fb-960h (large)	300M	87	745	10.4	25.7	55.9

gap penalties allow skipping words entirely — appropriate for sequence alignment but problematic when every reference word must be assigned to some chunk. True ASR insertions and deletions are handled by the cumulative assignment strategy, which aggregates reference words between chunk boundaries regardless of path structure.

On noisy audio, MFA produces no output in default mode and degrades from 51% to near-random accuracy (5.5%) within 20 min in single-speaker mode. The best CTC-based baselines (NFA Conformer, NFA Citrinet) range from 38–50% on@100 and degrade with file length. LFA avoids both problems by operating on short, well-segmented chunks where any standard aligner performs well.

Since the decomposition is independent of the per-chunk aligner, any FA backend can be used on long recordings without modification. All three backends we evaluate — HMM-based MFA, wav2vec2 CTC, and NeMo Conformer Viterbi — maintain stable accuracy from 5 min to 9.7 h, with quality differences reflecting each backend's short-segment performance rather than any interaction with recording duration. MFA gives the best median onset (52 ms) and wav2vec2 gives better accuracy at coarser thresholds (68.4% on@100).

Table 6: *Per-stage timing (LFA wav2vec2, C=10, A100 GPU). Per-file average in seconds. RTF = total / audio duration. Total includes I/O and initialization overhead not shown in individual stages.*

Dataset	**VAD**	**ASR**	**DTW**	**Align**	**Total**	**RTF**
5 min	0.3	8.0	1.9	6.2	19.1	0.066
10 min	0.6	15.9	4.9	12.6	36.8	0.062
20 min	1.3	30.8	13.1	24.4	72.4	0.062
35 min	2.2	53.7	9.3	42.7	111.1	0.054
1 h	3.7	92.5	16.4	72.5	189.0	0.054
2 h	9.3	322.6	34.1	148.8	519.6	0.074
full	37.6	910.2	209.6	727.4	1,895	0.054

The LFA NeMo vs. standalone NFA Conformer comparison is the cleanest ablation of chunking, since both use the same CTC model (`stt_en_conformer_ctc_large`) — LFA NeMo processes 10 s chunks while NFA runs on the full recording. On *TIMIT-LN-full*, LFA NeMo achieves 11× lower mean onset error (504 vs. 5,797 ms) with consistent gains at all thresholds: on@100 (+1.2 pp), on@200 (+2.2 pp), off@50 (+2.2 pp), and off@100 (+1.9 pp). Chunking eliminates large-scale drift while also improving per-word alignment at every threshold.

Our *TIMIT-LN-full* result (68.4% on@100 on a single 9.7 h file with 54,357 words) demonstrates that LFA scales to very long recordings. We use banded DTW with a band width of 5% of the matrix dimension, reducing memory from $O(N_{\text{ASR}} \times K)$ to $O(N_{\text{ASR}} \times B)$ where $B \ll K$. The DTW step (including G2P conversion) takes 1.9 s on 5-minute files, 16 s on 1-hour files, and 210 s on the 9.7-hour file — typically under 15% of total runtime. The system's overall runtime is dominated by ASR and per-chunk alignment, both scaling linearly with audio duration.

LFA requires three language-dependent components: ASR, CTC alignment, and G2P. When MFA is used as the per-chunk aligner, a language-specific acoustic model and dictionary are additionally required. The CTC-only configuration avoids this dependency. LFA supports any language for which all three components are available; the overlap of Whisper (99 languages), XLS-R wav2vec2 (over 50 fine-tuned), and Epitran (61 languages) covers several dozen languages, though we have

only evaluated on English.

LFA's reliance on ASR as an intermediate step introduces a structural limitation: words present in the reference transcript but absent from ASR output cannot be reliably assigned or aligned. Whisper (our ASR backend) systematically omits filled pauses and hesitations both by training convention (subtitle-derived data) and explicit text normalization [33]. On conversational speech such as Buckeye [12], where fillers constitute a substantial portion of annotated words, these tokens have no corresponding ASR hypothesis; DTW assigns them based on positional context alone, and CTC alignment is forced to place them without acoustic evidence. The same mechanism affects partial words, disfluencies, and interviewer speech that appears in the audio but not in the ASR output. Whole-file acoustic aligners such as NFA and MFA do not share this limitation, as they align the reference directly against the audio signal using models trained on such phenomena. Our TIMIT-LN benchmark avoids this issue because TIMIT consists of read speech without fillers, but applying LFA to conversational corpora would require a verbatim ASR system (e.g., CrisperWhisper [42]). For aligning parliamentary and broadcast speech, where reference transcripts are typically edited to remove disfluencies, this limitation has minimal practical impact.

6. Conclusion

We presented LFA, a modular divide-and-conquer approach to long-speech forced alignment that uses ASR to decompose the problem into independently solvable chunks, with phoneme DTW for reference transcript assignment. On noisy TIMIT recordings spanning 5 min to 9.7 h, standalone MFA fails or degrades severely while the best CTC-based baselines range from 38–50% on@100. LFA maintains stable accuracy across all durations: 68.4% on@100 (62 ms median onset) on a single 9.7-hour file with wav2vec2, and 63.5% (52 ms median onset) with MFA as per-chunk aligner. Any aligner capable of aligning 10 s chunks can be used to process very long recordings by operating on short, well-segmented chunks.

We release the LFA code as open-source and the benchmark dataset.[4]

Future work includes evaluating LFA on conversational corpora with overlapping speech and disfluencies and integrating speech enhancement as a preprocessing stage to improve VAD on severely degraded audio.

7. References

[1] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using Kaldi," in *Proc. Interspeech*, 2017, pp. 498–502.

[2] R. Rousso, E. Cohen, J. Keshet, and E. Chodroff, "Tradition or innovation: A comparison of modern ASR methods for forced alignment," in *Proc. Interspeech*, 2024, pp. 1525–1529.

[3] P. J. Moreno, C. Joerg, J.-M. Van Thong, and O. Glickman, "A recursive algorithm for the forced alignment of very long audio segments," in *Proc. ICSLP*, 1998.

[4] T. J. Hazen, "Automatic alignment and error correction of human generated transcripts for long speech recordings," in *Proc. Interspeech*, 2006.

[5] O. Aubert and J. Jäger, "Making parliamentary debates more accessible: Aligning video recordings with text proceedings in OpenParliamentTV," in *Proc. ParlaCLARIN IV Workshop*, 2024, pp. 77–83.

[4] Code and data will be made available upon publication.

[6] J. Wirth and R. Peinl, "ASR Bundestag: A large-scale political debate dataset in German," in *Proc. IntelliSys*. Springer, 2023, pp. 190–202.

[7] N. Ljubešić, P. Rupnik, and D. Koržinek, "The ParlaSpeech collection of automatically generated speech and text datasets from parliamentary proceedings," in *Proc. SPECOM*. Springer, 2024, pp. 137–150.

[8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[9] G. Doras, Y. Teytaut, and A. Roebel, "A linear memory CTC-based algorithm for text-to-voice alignment of very long audio recordings," *Applied Sciences*, vol. 13, no. 3, p. 1854, 2023.

[10] G. Bordel, S. Nieto, M. Penagarikano, L. J. Rodriguez-Fuentes, and A. Varona, "A simple and efficient method to align very long speech signals to acoustically imperfect transcriptions," in *Proc. Interspeech*, 2012, pp. 1840–1843.

[11] A. Katsamanis, M. P. Black, P. G. Georgiou, L. Goldstein, and S. Narayanan, "SailAlign: Robust long speech-text alignment," in *Proc. Workshop on New Tools and Methods for Very-Large Scale Phonetics Research*, 2011.

[12] M. A. Pitt, K. Johnson, E. Hume, S. Kiesling, and W. Raymond, "The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability," *Speech Communication*, vol. 45, no. 1, pp. 89–95, 2005.

[13] E. Rastorgueva, V. Lavrukhin, and B. Ginsburg, "NeMo forced aligner and its application to word alignment for subtitle generation," in *Proc. Interspeech*, 2023, pp. 5257–5258.

[14] L. Kürzinger, D. Winkelbauer, L. Li, T. Watzel, and G. Rigoll, "CTC-segmentation of large corpora for German end-to-end speech recognition," in *Proc. Speech and Computer (SPECOM)*. Springer, 2020, pp. 267–278.

[15] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-accurate speech transcription of long-form audio," in *Proc. Interspeech*, 2023, pp. 4489–4493.

[16] A. Pettarin, "aeneas: automagically synchronize audio and text," https://www.readbeyond.it/aeneas/, 2017, version 1.7.3, open-source tool.

[17] T. Kamp, "DSAlign: DeepSpeech based forced alignment tool," https://github.com/mozilla/DSAlign, 2020, mozilla, open-source tool (unmaintained).

[18] A. Álvarez, H. Arzelus, and P. Ruiz, "Long audio alignment for automatic subtitling using different phone-relatedness measures," in *Proc. ICASSP*, 2014, pp. 6280–6284.

[19] Y. Deng, F. Richardson, J. Steinberg, and P. Torres-Carrasquillo, "Speech-to-text forced alignment benchmark," in *Proc. ISPA*, 2025.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[21] R. Huang, X. Zhang, Z. Ni, L. Sun, M. Hira, J. Hwang, V. Manohar, V. Pratap, M. Wiesner, S. Watanabe, D. Povey, and S. Khudanpur, "Less peaky and more accurate CTC forced alignment by label priors," in *Proc. ICASSP*, 2024, pp. 11 831–11 835.

[22] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi, A. Baevski, Y. Adi, X. Zhang, W.-N. Hsu, A. Conneau, and M. Auli, "Scaling speech technology to 1,000+ languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

[23] J. Zhu, C. Zhang, and D. Jurgens, "Phone-to-audio alignment without text: A semi-supervised approach," in *Proc. ICASSP*, 2022, pp. 8167–8171.

[24] B. Mu, X. Shi, X. Wang, H. Liu, J. Xu, and L. Xie, "LLM-ForcedAligner: A non-autoregressive and accurate LLM-based forced aligner for multilingual and long-form speech," 2026.

[25] A. Rehman, J. Cai, J.-J. Zhang, and X. Yang, "BFA: Real-time multilingual text-to-speech forced alignment," 2025.

[26] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[27] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015, vol. 5.

[28] I. González-Carrasco, L. Puente, B. Ruiz-Mezcua, and J. L. López-Cuadrado, "Sub-Sync: Automatic synchronization of subtitles in the broadcasting of true live programs in Spanish," *IEEE Access*, vol. 7, pp. 60 968–60 983, 2019.

[29] J. M. Masiello-Ruiz, B. Ruiz-Mezcua, P. Martinez, and I. González-Carrasco, "Synchro-sub, an adaptive multi-algorithm framework for real-time subtitling synchronisation of multi-type TV programmes," *Computing*, vol. 105, pp. 1467–1495, 2023.

[30] K. Jin, S. Penke, and S. Algubelli, "VoiceNet: Multilingual on-device phoneme-to-audio alignment," in *Proc. Interspeech*, 2025, pp. 2260–2264.

[31] H. Bredin, "pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe," in *Proc. Interspeech*, 2023, pp. 1983–1987.

[32] G. Guillou, "Faster whisper transcription with CTranslate2," https://github.com/SYSTRAN/faster-whisper, 2023, software.

[33] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023, pp. 28 492–28 518.

[34] D. R. Mortensen, S. Dalmia, and P. Littell, "Epitran: Precision G2P for many languages," in *Proc. LREC*, 2018, pp. 2710–2714.

[35] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[36] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[37] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," in *Proc. Interspeech*, 2021, pp. 2426–2430.

[38] J. S. Garofolo, L. F. Lamel, W. M. Fisher, D. S. Pallett, N. L. Dahlgren, V. Zue, and J. G. Fiscus, "TIMIT acoustic-phonetic continuous speech corpus," Linguistic Data Consortium, LDC93S1, 1993.

[39] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: A Python package for audio room simulation and array processing algorithms," in *Proc. ICASSP*, 2018, pp. 351–355.

[40] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," 2015.

[41] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus: A pre-announcement," in *Proc. Machine Learning for Multimodal Interaction (MLMI)*. Springer, 2005, pp. 28–39.

[42] L. Wagner, B. Thallinger, and M. Zusag, "Crisperwhisper: Accurate timestamps on verbatim speech transcriptions," in *Proc. Interspeech 2024*, 2024, pp. 1265–1269.