Lyrics Alignment Model 歌詞の音節と MIDI Onset を高精度にマッチング 頭拍付きカラオケデータ CTC Loss (Wav2Vec2 + MIDI)

以下、**Lyrics Alignment Model** の全体像から技術詳細、入出力、学習プロセス、評価指標までを詳しく解説します。

1. 機能概要

「歌詞をどのタイミングで歌うか」を極めて高精度に推定し、MIDI 上のノートオンセット（頭拍）と一音節ずつ厳密に同期します。

- **対象**：ボーカル音声（生歌／合成）とそれに対応する MIDI シーケンス
- **目的**：カラオケ・字幕生成、歌詞ビジュアライズ、歌唱指導ツールなど

2. 入力データ

1. **音声ストリーム**
   - 録音済み歌唱データ、もしくは合成エンジン出力
   - 前処理で 16 kHz モノラル化、音量正規化
2. **頭拍付きカラオケ MIDI**
   - 各ノートに開始時刻／長さと「頭拍情報」（フレーズ先頭かどうか）をメタデータ化
   - 小節線／拍子情報も含む
3. **歌詞テキスト**
   - 正規化済み → 音素ごとに分割し、モデルの出力ラベル空間を構築
   - （例）「ありがとう」→ [a][ri][ga][to][u]

3. モデルアーキテクチャ

graph LR

A[Raw Audio] -->|feature| B(Wav2Vec2 Encoder)

C[MIDI Onsets] -->|embedding| D(MIDI Encoder)

B --> E[Feature Fusion]

D --> E

E --> F[CTC Decoder]

F --> G[Syllable Timings]

- **Wav2Vec2 Encoder**
  - 自己教師学習済みモデルから抽出した音声特徴（フレームごとの表現）
- **MIDI Encoder**
  - ノートオンセットの時系列を埋め込み →BiLSTM/Transformer で文脈エンコード
- **Feature Fusion**
  - 音声・MIDI 特徴を時系列に合わせて結合（時刻アライメント済み）
- **CTC Decoder**
  - 出力ラベル：各音素＋ blank
  - **CTC Loss** で「いつ何を歌ったか」を学習

4. CTC 学習の要点

- **目的関数**：音声系列（T フレーム）から、音素ラベル系列（N 単位）への可変長マッピングを最適化
- **blank トークン**：歌詞のない瞬間（無音部・ブレスなど）を表現
- **動的計画**：Forward–Backward アルゴリズムで loss と勾配を計算
- **収束条件**：N 歌詞ラベル全出力後に “blank” 連続で終了

5. 学習データ準備

1. **アノテーション**
   - 人手でアライメントされた音声／歌詞時刻データ → 音素単位でタイムスタンプ
2. **Data Augmentation**
   - ±1％ ピッチシフト、±50 ms タイミングジッター、リバーブ付加
3. **バッチ化**
   - 音声長・MIDI 長で可変 → PackedSequence／attention mask で処理

6. 推論フロー

1. 音声 →Wav2Vec2 特徴
2. MIDI→ 時刻エンコード
3. 特徴結合 →CTC デコーダ
4. **最適経路デコード**（Beam Search）で音素列とタイミングを取得
5. ノートオンセット時刻へスナップ（最寄り合わせ）し、**word／phrase レベル**にマージ

7. 評価指標

指標	定義	目標
音素アライメント誤差	平均絶対誤差	≤ 20 ms
音素精度	正解フレーム数/全フレーム数	≥ 95％
音素リコール	正解音素数/全真音素数	≥ 92％
CTC Loss	学習時検証セットの loss	≤ 0.15

8. 応用例

- **自動カラオケ歌詞表示**：歌詞下線／文字色変化を楽譜と同期
- **歌唱レッスン支援**：発声タイミングのズレを可視化
- **字幕・翻訳同期**：多言語歌詞をタイムスタンプ付きで自動生成

この **Lyrics Alignment Model** により、歌詞と音楽が呼吸を合わせたかのように一体化し、視聴／学習体験が飛躍的に向上します。ご不明点や導入イメージについて、ぜひお知らせください！

Lyrics Alignment Model — Patch Review & Codex Prompt (v1.0 🔧)

1. ブラッシュアップ提案

モデル設計

- **Wav2Vec2 Fine-tuning**: 現状は事前学習モデルを frozen ではなく全層微調整。性能向上のため、self.wav2vec.gradient_checkpointing=True を検討し、フリーズ可能な層数を設定できるように。
- **Positional Encoding**: MIDI 時刻埋め込みに絶対／相対位置バイアスを追加し、ノート長やビーコン情報を反映。
- **LayerNorm**: audio_proj 出力後と LSTM 出力前に LayerNorm を挿入し、安定性向上。

設定ファイル

- **Config サンプル**: midi_max_value や vocab_path など周辺設定を追加。
- **デフォルト値の明示**: YAML に freeze_encoder: true や gradient_checkpointing: false を追記。

学習スクリプト

- **早期終了**: EarlyStopping コールバック導入で MAE 50 ms 以下達成後に停止。
- **ログ出力**: 学習進捗・バッチ平均 MAE を TensorBoard ログに追加。
- **LR Scheduler**: cosine annealing や ReduceLROnPlateau の実装をサポート。

CLI & バッチ推論

- **エラーコード**: 推論例外発生時に sys.exit(1) を追加。
- **引数検証**: 入力ファイル存在チェック & フォーマット検証。
- **出力オプション**: --out_json でファイル保存可能に。

リアルタイム WS API

- **例外ハンドリング**: 途中エラー時に await ws.close(code=1011) を呼び予期せぬ切断を防止。
- **バッファ制限**: クライアントが大量送信した場合のフロー制御。
- **ヘルスチェック**: GET /health エンドポイント追加。

テスト強化

- **複数音節テスト**: 3 部音声で隙間なく流すサンプルを生成し、全ラベル抽出を検証。
- **無音部処理**: 無音だけの区間が正しく skip または blank として処理されることをテスト。
- **WS busy ケース**: 並列 /warmup 呼び出し時の status: busy を検証。

ドキュメント改善

- **依存関係**: transformers, librosa, pretty_midi, soundfile のバージョンレンジを列挙。
- **チュートリアル**: CLI → WS → Streamlit 等のフロー図を追加。
- **性能指標**: CTC Loss 値や MAE デモ結果をサンプルグラフで紹介。