第10回：音声インタフェースと UX 設計

小野原彩香｜北星学園大学

Zoom録画を後日公開します。課題はメール/Google Driveでも提出可。

参加スタイルについて

この授業は、**多様な参加方法**で学べる設計になっています。

- 演習・全体共有は**チャット書き込み・共有ドキュメント編集**でも参加可
- 発言は任意です。チャットへの書き込みは発言と同等に評価します
- カメラのオン/オフは自由です
- 質問は Zoom チャット・メールのどちらでも受け付けます
- 録画は後日公開します。自分のペースで視聴してください

本日の到達目標

1. 音声 UI の歴史的発展を概観し、現在の多様な利用文脈を把握できる
2. オンデバイス ASR とクラウド ASR のトレードオフを技術・プライバシー両面から説明できる
3. エッジ推論における小型化手法（量子化・プルーニング・知識蒸留）の概念を理解できる
4. 音声インタフェースの UX 設計原則に基づき、アクセシブルなシナリオを提案できる

授業の流れ

時間	内容
0〜10分	前回（第9回）の振り返り・本日の概要説明
10〜45分	スライド講義（音声 UI の歴史〜UX 手法）
45〜65分	演習（UX 設計シナリオ作成）
65〜80分	全体共有
80〜90分	小課題説明・質疑・まとめ

音声 UI の歴史① — 電話自動応答（IVR）

IVR（Interactive Voice Response）：電話での自動応答

- 「1を押すとお問い合わせ、2を押すとご予約」形式
- プッシュ入力型では**DTMF トーン**を識別。音声認識を使うIVRもある
- 企業の電話窓口・銀行照会・航空券予約に普及
- 問題点：分岐が深くなると迷子になる「**IVR の迷宮**」現象

音声ダイアログシステム（1990年代〜2000年代）

- 音声認識・合成音声などを使って電話で対話
- コールセンターへの導入拡大
- VoiceXML 2.0（2004）：対話・音声・DTMF等を記述する規格

> 考えてみよう：IVR と現代の音声アシスタントの最大の違いは何でしょうか？（チャットに書いてみてください）

音声 UI の歴史② — スマートスピーカー以降

製品	年	特徴
Siri（Apple）	2011	iPhone 4Sに音声アシスタントを統合
Google Now	2012	検索連携・文脈理解
Alexa（Amazon Echo）	2014	ウェイクワードで起動する音声アシスタント
Google Assistant	2016	Allo・Google Home向けの対話アシスタント
ChatGPT 音声モード	2023	当初はASR → LLM → TTSで音声対話

マルチモーダル UI（表の年は発表・導入時点）

- 音声＋タッチ＋視線＋ジェスチャーの統合
- 車載インフォテインメント、AR/VR ヘッドセット
- 音声＋画面表示の組み合わせ（Echo Show、Google Nest Hub）

UX 設計原則① — エラーリカバリ

音声では誤認識や聞き逃しが起こる → **訂正・回復の方法を設計する**

> ユーザー：「明日の天気を教えて」
> システム：「明後日の天気ですね、東京は晴れです」（誤認識）
> ユーザー：「違う、明日」
> システム：「失礼しました。明日の天気ですね。
> 東京は曇りのち晴れ、最高気温18度です」

設計のポイント

- 認識結果の読み上げ確認（コンファメーション）
- 短い訂正インタラクション
- 「もう一度おっしゃってください」フレーズの最適化
- 失敗時の代替入力や**人への引き継ぎ**を、用途に応じて設計

UX 設計原則② — 確認ダイアログとターン設計

確認ダイアログの設計

```
明示的確認（Explicit）：
「○○を削除します。よろしいですか？」
→ 重要・取り消し不可操作に使用

暗黙的確認（Implicit）：
「明日の東京の天気ですね。曇りの予報です」
→ 間違えても訂正しやすい場面で使用
```

ターン設計の原則

原則	理由
簡潔な応答	重要情報は前半に。再生・読み直し・画面表示を用意
選択肢と応答長の調整	覚える量・利用状況・画面併用に応じて設計
プロンプトの明確化	「どうぞ」より「何を検索しますか？」
沈黙のハンドリング	適切なタイムアウトと再プロンプト

クラウド ASR とオンデバイス ASR

クラウド：音声をサーバーに送り認識
オンデバイス：端末内のモデルで認識

指標	クラウド ASR	オンデバイス ASR
応答時間	通信・待ち時間・推論・発話終了検出による	端末性能・推論・発話終了検出による
ストリーミング	モデル・実装により可能	モデル・実装により可能
オフライン認識	通常は不可	必要なモデル等を配置すれば可能
精度・対応言語	モデル・学習データ・入力条件に依存	
計算資源	サーバーの資源	端末のメモリ・電力・演算性能に制約

通信往復・文字起こし確定・返答音声開始は異なる時間。**計測区間と条件をそろえる。**
端末内で認識しても、天気検索などは通信を必要としうる。

プライバシーとセキュリティ

クラウド ASR で確認すること

- どの音声・文字列が、いつ、どこへ送られるか
- 誤ったウェイクワード検出時の扱い
- 保存・学習利用・削除の条件、アクセス制御

オンデバイス ASR の特徴

- 認識処理に伴う音声の外部送信を避けられる
- 文字起こしの送信・保存・バックアップはアプリの設計による
- 利用分野への適合性は、データの扱い全体で判断する

> マイクの待受、音声の保存、外部送信は別の処理。
> 「マイクが有効」と「音声をすべて常時送信」は同義ではない。

エッジ推論① ― 量子化（Quantization）

概念：重み・活性値を低ビットの表現で近似

```
float32 ：1パラメータ = 32ビット（4バイト）
int8    ：1パラメータ =  8ビット（1バイト）→ 重みの値の保存領域は理論上 1/4
int4    ：1パラメータ =  4ビット（0.5バイト）→ 重みの値の保存領域は理論上 1/8
```

トレードオフ

- 保存領域を削減。スケール値・未量子化部分・実行時メモリも必要
- 速度・精度への影響は、方式・モデル・実行環境による
- Post-Training Quantization（PTQ）vs. Quantization-Aware Training（QAT）

エッジ推論② — プルーニングと知識蒸留

プルーニング（Pruning）：重要度の低いパラメータを削除

```
密なモデル（Dense）→ 疎なモデル（Sparse）
例：100個の重みのうち70個をゼロ化 → 非ゼロの重みが30個
```

- 非構造化プルーニング：個々の重みをゼロ化・削除。密な演算のままでは高速化しない場合もある
- 構造化プルーニング：チャネル・レイヤー全体を削除（ハードウェア最適化しやすい）

知識蒸留（Knowledge Distillation）：大規模モデルの知識を小規模モデルに転移

```
教師モデル（大規模な音声認識モデル）
      ↓ 出力分布・生成した文字列などを学習信号にする
生徒モデル（小規模な音声認識モデル）
```

教師の出力分布から、他候補との相対関係も学ぶ。
公開Whisper tinyがlarge-v3の蒸留モデルという意味ではない。

オンデバイス ASR の代表例

例	特徴・確認する条件
whisper.cpp	CPUに加え、対応GPU・NPU等も利用可能。量子化に対応。MITライセンス
tinyの規模	3900万パラメータ。非量子化ファイル約75 MiB。実行時メモリはさらに必要
Vosk	Kaldiベース。日本語モデルsmall-ja-0.22 / ja-0.22など。語彙の変更可否はモデルの構成による
Apple Siri	iOS 15（2021）で端末内音声認識を紹介。機器・言語・機能に条件がある
Apple Intelligence	端末内モデルとPrivate Cloud Computeのサーバーモデルを併用

速度を固定値で一般化しない：端末・量子化・スレッド数・音声長・復号条件を示してRTFなどで測る。

出典：iOS 15・Apple基盤モデル

音声 UI のアクセシビリティ設計

音声インタフェースはアクセシビリティを高める一方で、**新たなバリアを生む可能性**もある。

音声入力が難しくなる条件の例

- 吃音（Stuttering）
- 失語症（Aphasia）・運動障害性構音障害（Dysarthria）
- 喉頭摘出後
- 緊張・不安による一時的な発話困難
- 場所的制約（図書館・電車内等）

設計上の対策

課題	対策
吃音による誤認識	タイムアウトを長めに設定・繰り返し発話への対応
声量が小さい	感度の自動調整・増幅機能
発話自体が困難	テキスト入力・スイッチアクセスをフォールバックとして**常時**提供

Project Euphonia と多様な利用者への配慮

Google Project Euphonia

- 非標準的な音声への認識精度向上を目指す研究
- 個人の音声を用いた適応により、ASRのWERが改善した報告
- **本人の発話明瞭度の改善ではなく、認識システムの成績の改善**

高齢者への配慮：年齢だけで能力を一括りにしない

- 読み上げ速度・音量・文字サイズを調整できるようにする
- 発話の間・言い直しに対応し、訂正方法を分かりやすく示す
- 音声と画面・文字を併用し、複雑な構文を覚えず操作できるようにする

ユニバーサルデザインは利用の柔軟性・分かりやすい情報・エラーへの許容などを重視する。一つの設計ですべての要求を満たせるとは限らない。

出典：Shorほか（2019）・NC State：7原則

UX リサーチ手法

Wizard of Oz（WoZ）法：人間がシステムの一部・全部を代行し、簡易なプロトタイプで対話や操作を調べる

```
参加者（ユーザー役）  ←音声→ マイク
                                ↓
                        [別室のオペレーター]（応答を模倣）
                                ↓
参加者                ←スピーカー← 応答
```

本実装の前に対話を探索できる。人が代行した成功率・速度を完成システムの性能とみなさない。

その他の手法

手法	音声 UI での活用
Think Aloud 法	音声操作への干渉に注意。操作後の振り返りも検討
発話ログ分析	頻出コマンド・失敗パターン発見
A/B テスト	無作為割付などでプロンプト変更の効果を比較
ヒューリスティック評価	設計初期の問題発見

音声 UI のユーザビリティ評価指標

指標	定義
Task Completion Rate	成功したタスク試行数 ÷ 全タスク試行数
Turn Count	対話ターン数（発話か一往復か、数え方を統一）
WER（Word Error Rate）	音声認識誤り率（第9回で学習）
Time on Task	タスク完了に要した時間
Recovery Rate	回復したエラー事例数 ÷ 回復対象のエラー事例数
主観的ユーザビリティ（SUS 等）	ユーザーが感じる使いやすさ

> WERが低くても、下流処理・対話設計・タスク条件などでタスク完了率は低くなりうる。
> → 技術的な指標とユーザー体験指標を組み合わせて評価する。

演習：音声インタフェースを UX 視点で設計する（20分）

目標：UX 設計原則とアクセシビリティの知識を活かし、具体的なシナリオを作成する。
自分のペースで取り組んでください。Zoom 参加またはメール提出でも可。

STEP 1：ペルソナの決定（5分）

> ペルソナ名（仮名でOK）：
> 年齢・職業：
> 音声UI利用シーン：
> 使用デバイス：
> 特別な配慮が必要な点（あれば）：

ペルソナ例のヒント：

- 視覚障害のある 30 代会社員が通勤中に使う音声メモアプリ
- 80 代の祖父母が使う薬の服用管理アシスタント
- 吃音のある大学生が講義中に使う静音テキスト変換ツール
- 調理中の料理人向けレシピナビ

演習：シナリオ作成（STEP 2〜3）

STEP 2：シナリオの作成（10分）

> 【場面設定】
> （どこで・何をしているときに音声UIを使うか）
>
> 【タスク】
> （達成したいこと）
>
> 【対話シナリオ（3〜5ターン）】
> ユーザー：「〇〇」
> システム：「〇〇」
> ユーザー：「〇〇」
> システム：「〇〇」
>
> 【エラーケース（1つ）】
> （誤認識が起きたとき、どう回復するか）
>
> 【アクセシビリティ配慮】
> （どんな多様なユーザーに対応しているか）

STEP 3：共有（5分）

チャットまたは共有ドキュメントにシナリオを貼り付けてください。
Zoom 参加中の方は口頭での紹介も歓迎。

全体共有（15分）

投稿いただいたシナリオへの**講評のポイント**：

1. エラーリカバリは考慮されているか？
2. アクセシビリティへの配慮はあるか？
3. 第8〜9回までの技術（Whisper・CER等）で実現できる設計か？

小課題

提出方法：Google Drive または メール添付
締め切り：翌週末 23:59
形式：400字程度（超えても問題ありません）
備考：音声録音での提出・箇条書きでの提出も可。分量より内容を重視します。

課題内容：

> 音声インタフェースの**良い点・問題点をそれぞれ3つ**挙げ、UX 改善案を提案せよ。

解答の観点（必ずしも従う必要はありません）：

- **技術的改善**（ASR 精度向上・ノイズ除去等）
- **デザイン的改善**（UI 設計・フィードバック方法等）
- **社会的・倫理的改善**（データ利用ポリシー・アクセシビリティ等）

本日のまとめ

1. **音声 UI の歴史**：IVR（DTMF）→ 限定語彙 ASR → スマートアシスタント → LLM ベース音声 UI へ進化。
2. **UX 設計原則**：エラーリカバリ・確認ダイアログ・ターン設計・マルチターン対話の文脈保持。
3. **クラウド vs オンデバイス**：精度・遅延・プライバシー・オフライン動作のトレードオフ。
4. **エッジ推論の小型化**：量子化・プルーニング・知識蒸留の3手法。
5. **アクセシビリティ設計**：多様な発話困難者・高齢者への配慮。フォールバック手段を常時提供。
6. **UX リサーチ**：WoZ 法・Think Aloud・発話ログ分析など。技術指標（WER）とユーザー体験指標を組み合わせる。

次回予告（第11回：動画回）

自然言語処理との接続（NLU と音声）

- 音声処理パイプライン全体（ASR → NLU → DM → NLG → TTS）
- 意図推定・スロット抽出・NER
- LLM との接続（Whisper + ChatGPT）
- Python 実習（Google Colab）

事前準備（任意）：Google Colab で Whisper を動かしてみる：

```
!pip install openai-whisper
import whisper
model = whisper.load_model("tiny")
```