第11回：自然言語処理との接続

NLUと音声

音声処理と認識

小野原彩香 ／ 北星学園大学

この回は動画回です。Google Formsでテストを受けてください。

この回の学習目標

1. 音声処理パイプライン（ASR → NLU → DM → NLG → TTS）の全体像を説明できる
2. NLUの主要タスク（意図推定・スロット抽出・NER）を区別できる
3. SLUがNLUとどう異なるかを理解できる
4. 対話管理（DM）のアプローチを比較できる
5. LLMと音声処理の統合方法と、構成ごとの違いを説明できる
6. PythonでWhisper・transformersを使った実装ができる

参加方法と配慮事項

この回はCBT・動画形式です。**自分のペースで**進めてください。

- 動画は一時停止・巻き戻し・字幕表示を活用してください
- CBTテストは何度でも挑戦できます（最高点を記録）
- Python実習は「読んで理解すること」でも可
- 不明点・エラーはそのままメールで質問してください

学習の進め方

```
[1] スライド講義を読む（または動画を視聴）
        ↓
[2] Python実習（Google Colab）を実行する
        ↓
[3] CBTテスト（5問）に挑戦する
        ↓
[4] 小課題を提出する（任意・加点）
```

音声処理パイプラインの基本例

```
音声入力
    ↓
[ASR] 音声認識（Automatic Speech Recognition）
    ↓ テキスト
[NLU] 自然言語理解（Natural Language Understanding）
    ↓ 意図・スロット・エンティティ
[DM]  対話管理（Dialogue Management）
    ↓ 行動指示・応答内容
[NLG] 自然言語生成（Natural Language Generation）
    ↓ テキスト応答
[TTS] 音声合成（Text-to-Speech）
    ↓
音声出力
```

各モジュールの役割

モジュール	入力	出力	例
ASR	音声波形	テキスト	「東京から大阪まで新幹線を予約して」
NLU	テキスト	意図・スロット	意図：鉄道予約, 出発地：東京
DM	意図・スロット・履歴	行動計画	日時が未確定 → 日時を質問
NLG	行動計画	テキスト	「出発日時をお教えください」
TTS	テキスト	音声	合成音声で読み上げ

NLU タスク①：意図推定（Intent Classification）

ユーザーが「何をしたいか」を分類するタスク。
例のRoom・Dateは、次のスロット抽出の出力。

```
入力：「リビングの電気を消して」
出力：Intent = TurnOff_Light, Room = リビング

入力：「今日の天気は？」
出力：Intent = GetWeather, Date = 今日
```

アプローチ	手法	特徴
ルールベース	正規表現・キーワードマッチ	高速・解釈可能
機械学習	SVM, ロジスティック回帰	単語特徴や事前学習済み埋め込みを入力
深層学習	BERT, RoBERTa等	文脈を反映した表現で分類。性能は条件による

NLU タスク②：スロット抽出（Slot Filling）

意図を実現するために必要な情報を抽出するタスク。

```
# 飛行機予約の例
入力：「来週の金曜日に東京から福岡まで飛行機を予約したい」

スロット抽出結果：
{
  "departure_city": "東京",
  "destination_city": "福岡",
  "travel_date": "来週の金曜日",
  "transport_mode": "飛行機"
}
# 未充填スロット → システムが追加で質問する
```

NLU タスク③：固有表現認識（NER）

人名・地名などの表現の範囲と種類を推定する。
日時・数値を含めるかなど、対象の種類はデータセットによる。

> 入力：「山田太郎さんが2024年3月に東京大学で発表した論文を探して」
>
> NER結果：
> [山田太郎]PER が [2024年3月]DATE に [東京大学]ORG で発表した論文を探して

タイプ	説明	例
PER	人名	山田太郎
LOC	地名	東京、北海道
ORG	組織名	トヨタ、文部科学省
DATE/TIME	日時表現	3月15日、来週月曜

SLU — 発話音声から意味情報を得る

SLU（Spoken Language Understanding）には、ASR＋テキストNLUを使う構成と、音声から直接意味を推定する構成がある。

構成	処理の流れ
テキストNLU	テキスト → 意図・スロット等
カスケード型SLU	音声 → ASR → テキストNLU → 意味情報
直接型SLU	音声 → 意味情報（書き起こしを必須としない）

- NLUは書き起こし・チャットなども扱う
- 句読点・フィラーの有無は、ASRの出力設定などに依存する
- ASRの誤りや音響条件の変動への対応が必要。韻律等も利用しうる

出典：Lugoschほか（2019）

対話管理（Dialogue Management）

DM は会話の「流れ」をコントロールするモジュール。

```
# ルールベースDMの例
if intent == "鉄道予約":
    if departure_city is None:
        action = "出発地を質問する"
    elif destination_city is None:
        action = "目的地を質問する"
    elif travel_date is None:
        action = "日時を質問する"
    else:
        action = "予約APIを呼び出す"
```

利点：動作が予測可能、デバッグ容易
欠点：ルールが増えると管理困難、想定外の入力に弱い

対話状態トラッキング（DST）

```
ターン1: ユーザー「東京から大阪に行きたい」
  State：{departure: 東京, destination: 大阪, date: ?, count: ?}

ターン2: ユーザー「来週の土曜」
  State：{departure: 東京, destination: 大阪, date: 来週土曜, count: ?}

ターン3: ユーザー「2人で」
  State：{departure: 東京, destination: 大阪, date: 来週土曜, count: 2}
  → この例のスロットが充填 → 空席・料金照会や候補提示へ
```

LLM との接続 — アーキテクチャの変化

従来のパイプライン型

```
音声 → ASR → テキスト → NLU → DM → NLG → TTS → 音声
```

テキストLLM統合型（これもカスケード型）

```
音声 → ASR → テキスト → テキストLLM → テキスト → TTS → 音声
                              ↑
                    NLU・DM・NLGの機能の一部または複数を担当
```

Few-shotプロンプトは例を入力に含める方法で、通常は重みを更新しない。
履歴や状態を渡して文脈を扱うが、正確に保持できるかは評価する。

音声を直接扱う例：GPT-4o（2024年発表）

発表では、テキスト・画像・音声を同じモデルで学習し、**音声を直接入出力する構成**を説明。

> カスケード型：音声 → ASR → テキストLLM → TTS → 音声
> 直接音声を扱う構成：音声 → 音声入出力モデル → 音声

- 文字列だけでは渡しにくい韻律・笑い声なども利用できる
- **情報を損失なく保持することや、感情・意味を必ず正しく推定することは保証しない**
- 平均応答時間320ミリ秒は、当時の発表資料の値。利用環境によらず一定の待ち時間という意味ではない

構成を学ぶ歴史的な例。現在の機能一覧ではない。
出典：OpenAI：Hello GPT-4o（2024）

構成を比較するときの観点

観点	NLU・DM・NLGを分ける構成	テキストLLMを統合する構成
役割	各モジュールの入出力を定める	LLMが複数の機能を兼ねられる
検証	個別の処理と接続を検証	モデル・指示・外部処理の組合せを検証
文脈	状態や履歴を渡す	状態や履歴を入力。保持の正確さを確認
コスト・速度	モデル・入出力長・実装・計算資源・通信による	
処理場所	端末内・クラウド・混在の構成がある	

> LLMを使うことと、音声から音声までE2Eで処理することは別の設計軸。構成名だけで精度・価格・遅延の優劣を決めない。

Python実習①：音声認識＋簡易意図分類

```
# 認識後のテキストに適用する簡易ルール（抜粋）
INTENT_PATTERNS = {
    "天気確認": ["天気", "気温", "雨"],
    "アラーム設定": ["アラーム", "目覚まし", "起こして"],
    "電気操作": ["電気", "ライト", "照明"],
}
def classify_intent(text: str) -> dict:
    scores = {intent: sum(word in text for word in words)
              for intent, words in INTENT_PATTERNS.items()}
    best = max(scores.values(), default=0)
    candidates = [intent for intent, score in scores.items()
                  if score == best and score > 0]
    return {"intent": candidates[0] if len(candidates) == 1 else "不明",
            "rule_score": best, "candidates": candidates}
```

一致数は正解確率ではない。同点は候補を返す。

Colabでは録音済みファイルを使用。sr.Microphoneはローカル PC用。

Python実習②：Whisper＋センチメント分析

```
import whisper
from transformers import pipeline
whisper_model = whisper.load_model("tiny", device="cpu")
sentiment_analyzer = pipeline("sentiment-analysis",
    model="lxyuan/distilbert-base-multilingual-cased-sentiments-student")

def speech_to_sentiment(audio_file: str) -> dict:
    transcript = whisper_model.transcribe(
        audio_file, language="ja", fp16=False)["text"].strip()
    if not transcript:
        return {"transcript": "", "sentiment": None, "score": None}
    sentiment = sentiment_analyzer(
        transcript, truncation=True, max_length=512)[0]
    return {"transcript": transcript, "sentiment": sentiment["label"],
            "score": sentiment["score"]}
```

CPU版の抜粋。512は文字数ではなくトークン数。
分類対象は文字起こしの内容。声の感情の推定ではない。

評価指標まとめ

指標	対象	説明
WER	ASR	単語誤り率。評価データ・分割・正規化を統一
Intent Accuracy	NLU	意図分類の正解率
Slot F1	NLU	スロットの適合率・再現率の調和平均
Joint Goal Accuracy	DST	全スロット・値が正解状態と一致するターンの割合
BLEU / ROUGE	NLG	参照文との一致。ROUGE-Lは最長共通部分列
MOS	TTS	人による評点の平均（例：5段階）。自動評価ではない
Task Completion Rate	対話全体	成功したタスク試行数 ÷ 全試行数

WERが0%でも、意図や応答を誤ることがある。各処理と対話全体を評価する。

まとめ：第11回

- **音声対話の基本例**：ASR → NLU → DM → NLG → TTS
- **NLUのタスクの例**：意図推定・スロット抽出・固有表現認識
- **SLU**：音声から意味情報を取り出す。カスケード型・直接型がある
- **対話管理**：ルールベースと機械学習の二大アプローチ
- **LLM統合**：NLU・DM・NLGの機能を担当できる。音声E2Eとは別の設計軸

CBTテスト・次回予告

CBTテスト（Google Forms）

5問・何度でも挑戦可・最高点を記録します。
締め切り：翌週末 23:59

第12回（オンライン同期）：音声処理と認知科学・言語学

- 言語発達と音声知覚（臨界期仮説・音韻発達）
- 神経多様性と音声処理（ディスレクシア・APD・ASD）
- 人間と機械の音声認識の比較
- ディスカッション：「音声インタフェースはバリアかツールか？」