#対話理解モジュール ユーザー発話をテキスト／音声から解析（最後に組み込む予定） Gemini API（LLM）＋ Whisper API（音声→テキスト）＋ MeCab/SudachiPy

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

geminiモデルのインストールコマンド：

**# pip install google-generativeai**

APIキーの設定（安全管理）

APIキーを環境変数として設定します

export GOOGLE\_API\_KEY="YOUR\_GEMINI\_API\_KEY"

Pythonでは：

import os

import google.generativeai as genai

genai.configure(api\_key=os.getenv("GOOGLE\_API\_KEY"))

モデル呼び出し

Gemini-1.5-Pro（または最新モデル）を使用します。

model = genai.GenerativeModel("gemini-1.5-pro")

response = model.generate\_content("こんにちは、あなたは法律AIです。")

print(response.text)

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

データフロー：

入出力仕様（I/O設計）

🎤 入力（Input）

種別 内容 型 例

音声 ユーザー発話（.wav / .mp3） bytes / str “user\_input.wav”

テキスト 音声変換後 or 直接入力 str 「夫が家事を全くしません」

💬 出力（Output）

種別 内容 型 例

テキスト解析結果 発話の構造と意味分析 dict { "intent": "不満訴え", "subject": "夫", "topic": "家事負担" }

Gemini解析結果 Gemini応答 str 「家事分担の偏りに関する不満が検出されました。」

**# 全体フロー**

[音声入力]

↓

Whisper（音声→テキスト）

↓

MeCab/SudachiPy（形態素解析）

↓

Gemini（文脈理解・意図抽出）

↓

[出力：構文＋意図解析結果]

**# 将来の統合を見据えた統合設計**

riri\_core/

├── dialogue\_understanding/

│ ├── **\_\_init\_\_**.py

│ ├── whisper\_interface.py # 音声→テキスト

│ ├── morpho\_parser.py # MeCab or SudachiPy

│ ├── gemini\_core.py # Gemini連携

│ └── dialogue\_understander.py # 統合制御

├── emotion\_analysis/

├── legal\_reasoning/

├── main.py

└── config.py

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

将来統合のための設計上の注意点

項目 説明 理由

入出力をJSON化 他モジュール間連携を容易にする Emotion / Legalモジュールに渡しやすくする

Whisper・Gemini・形態素解析をクラス分離 交換性を確保 モデル切替やAPI変更に対応

Geminiの呼び出し結果を構造化JSON出力 法令推論や折衷案生成に転用可 下流層での処理が容易

環境変数管理（.env使用） APIキーのセキュリティ確保 外部公開を防止

モジュール内でロギング処理追加

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_