


パーソナライズされた話題提供モデルの アプリケーション化に向けて

 武蔵野大学工学部数理工学科佐々木研究室 学部4年
CATechLounge ML/DS
白川桃子

目 次

- 
- 01 - 話者分離の手法調査
 - 02 - 今回活用する話者分離の手法
 - 03 - Google CloudのApp Engine / Cloud Runとは
 - 04 - アーキテクチャ図
 - 05 - シーケンス図
-
-

01

話者分離の手法調査

pyannote.audio

Pythonのオープンソースフレームワーク
機械学習のフレームワークPyTorchに基づいている

Google Cloud 話者ダイアライゼーション

Google CloudのSpeech-to-Textに付随する機能

Deepgram

2015年に設立したスタートアップ企業のサービス
AIを用いた企業向けの音声認識に特化している

[参考] GitHub“pyannote-audio” <https://github.com/pyannote/pyannote-audio>
Google Cloud“音声録音内の異なる話者を分離する” <https://cloud.google.com/speech-to-text/v2/docs/multiple-voices?hl=ja>
deepgram“Deepgram Voiice AI:Text to speech + Speech to Text APIs” <https://deepgram.com/>

01

話者分離の手法調査

pyannote.audio

Pythonのオープンソースフレームワーク
機械学習のフレームワークPyTorchに基づいている

Google Cloud
話者ダイアライゼーション

GoogleのSpeech-to-Textに付随する機能

Deepgram

2015年に設立したスタートアップ企業のサービス
AIを用いた企業向けの音声認識に特化している

[参考] GitHub“pyannote-audio” <https://github.com/pyannote/pyannote-audio>
Google Cloud“音声録音内の異なる話者を分離する” <https://cloud.google.com/speech-to-text/v2/docs/multiple-voices?hl=ja>
deepgram“Deepgram Voiice AI:Text to speech + Speech to Text APIs” <https://deepgram.com/>

今回活用する話者分離の手法

Google Cloud 話者ダイアライゼーション

GoogleのSpeech-to-Textに付随する機能

文字起こしする際に、話者ダイアライゼーションを有効にすると
音声内に含まれる異なる音声を識別することができる！

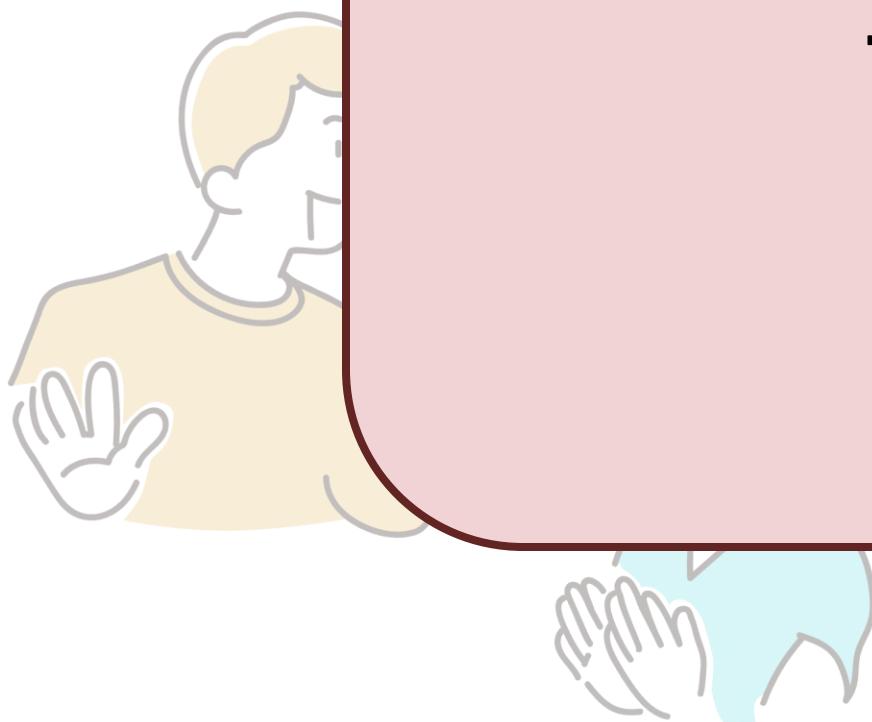


- 1 最近なんかハマってることある？
- 2 夜に散歩するのが楽しくてさ、 ...
- 1 わかる！夜の空気って気持ちいいし、 ...

Google
話者ダ

-問題点-

**音声認識結果と各ユーザーを
どのように結びつけるか**



ることある?
しくてさ、...
って気持ちいいし、...

03

Google CloudのApp Engine / Cloud Runとは

	App Engine	Cloud Run
共通	サーバーの管理が不要 従量課金制 オートスケーリング(=負荷に応じてサーバーの台数やスペックを自動的に調整する)	
デプロイ	GCP(=Google Cloud Platform)が用意する ランタイム環境のみ	任意のランタイム環境で コンテナを作れる
アクセス状態(オススメ)	常時アクセスがある	常時はアクセスがない
開発	アプリ開発に集中できる (インフラ管理やスケーリングをGCが行う為)	コンテナ技術に精通した開発者が 使い慣れた環境で開発できる

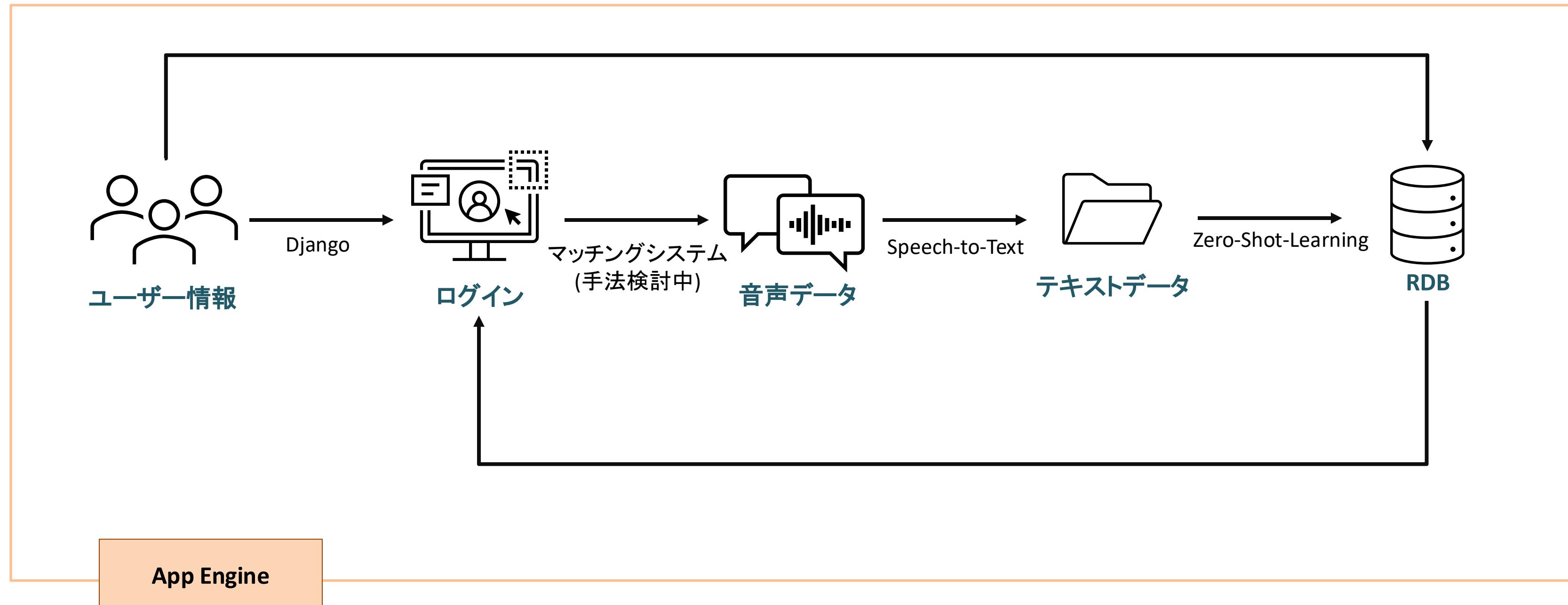
[参考] ユタラプトル”【GCP】App EngineとCloud Runの違い” <https://qiita.com/goziku/items/17a9d51cb2962c9b32e5>
ジーアイクラウド株式会社”Cloud Run/App Engine/Cloud Functionsの使い分け” <https://dryaki.gicloud.co.jp/articles/choice-serverless>

03

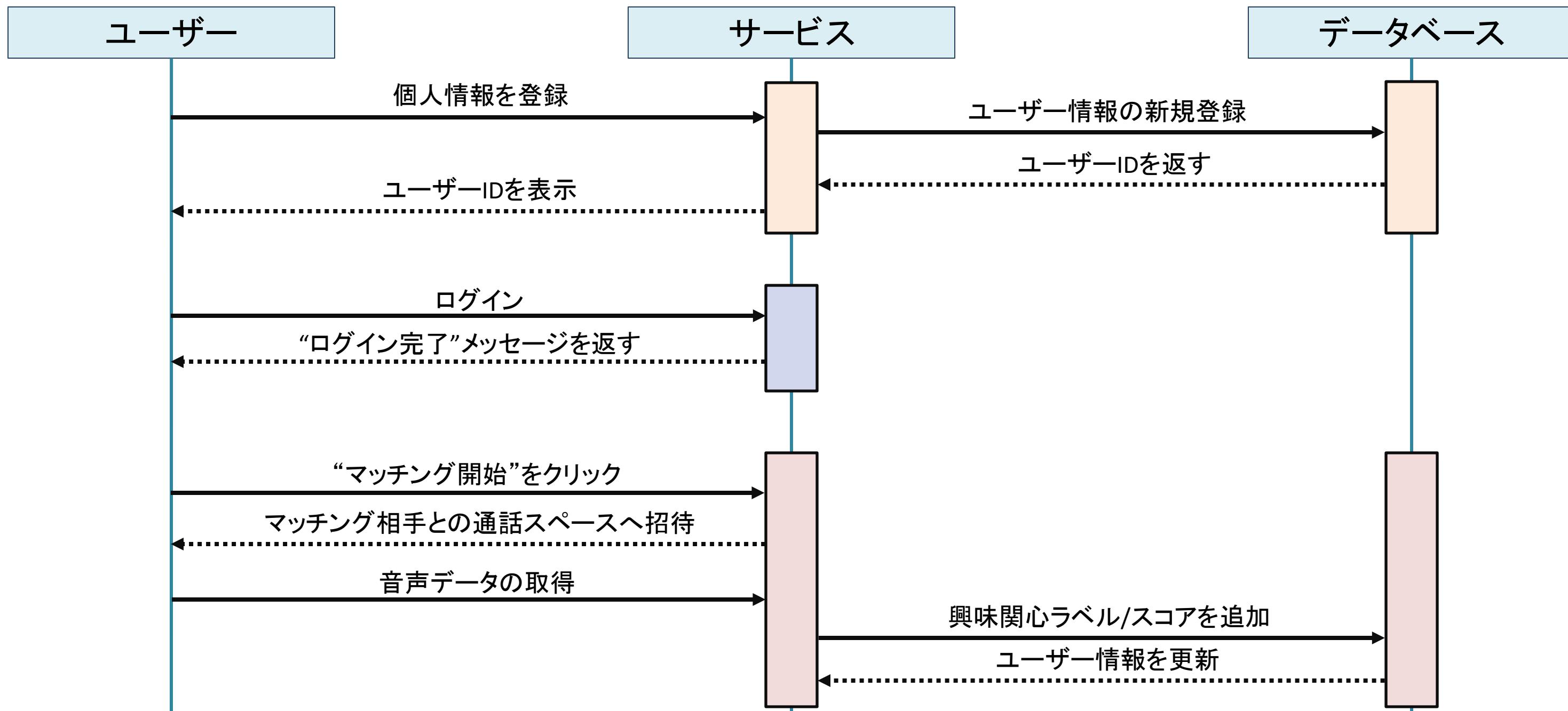
Google CloudのApp Engine / Cloud Runとは

今回使うサービス		
	App Engine	Cloud Run
共通	サーバーの管理が不要 従量課金制 オートスケーリング(=負荷に応じてサーバーの台数やスペックを自動的に調整する)	
デプロイ	GCP(=Google Cloud Platform)が用意する ランタイム環境のみ	任意のランタイム環境で コンテナを作れる
アクセス状態(オススメ)	常時アクセスがある	常時はアクセスがない
開発	アプリ開発に集中できる (インフラ管理やスケーリングをGCが行うから)	コンテナ技術に精通した開発者が 使い慣れた環境で開発できる

[参考] ユタラプトル”【GCP】App EngineとCloud Runの違い” <https://qiita.com/goziku/items/17a9d51cb2962c9b32e5>
ジーアイクラウド株式会社”Cloud Run/App Engine/Cloud Functionsの使い分け” <https://dryaki.gicloud.co.jp/articles/choice-serverless>



[参考] Lucidchart"アーキテクチャ図作成方法とツール、書き方の紹介" <https://www.lucidchart.com/blog/ja/how-to-draw-architectural-diagrams>
Django documentation"Django ドキュメント" <https://docs.djangoproject.com/ja/5.1/>



[参考] 株式会社システムインテグレータ“シーケンス図とは？必要性や構成要素、作成時のポイントまで” <https://products.sint.co.jp/ober/blog/sequence>