オンプレミス環境を対象とした AIエージェント音声対話サービス基盤の開発

三浦　彰人*

Development of AI agent spoken dialogue service infrastructure for on-premises environment

Akito Miura*

要旨： XRビジネス創出事業や研究・教育環境の整備を目的とし，特定のサービスやプラットフォームに依存しない，AIエージェントとのリアルタイムな音声対話を実現するサービス基盤を開発した．また，サービス基盤の応用例として，山形県に詳しいAIエージェントと音声で対話できるサービスを開発した．
キーワード: AIエージェント，音声認識，検索拡張生成(RAG)，WebRTC，3Dキャラクター

１．緒　言

筆者は現在，山形県の「XR ビジネス創出事業」に参加している. XR技術は, 音声・画像認識など, AI技術によって成り立っているものも多い．したがって, XRで魅力的なコンテンツを創り上げたい場合, XRとAI双方の知見が不可欠である. また, 事業で得られた知見や成果は，学生の教育にも活かしたい．しかしながら，AI技術の多くは有償のクラウドサービスに強く依存しており, 教育・研究環境に取り入れるには難がある. 加えて, 既存サービスを利用するだけでは，AIサービス基盤の運用ノウハウが得られないといった問題もある.

そこで,「AIエージェントとの音声対話」をテーマとし,「音声対話を支えるAIサービス基盤」と，具体的な利用例としてのAIエージェントと音声対話ができるサービス」の開発を進めることにした.

２．課　題

AIエージェントとの対話を実現するサービスを構築する際は，以下の要素が必要である.

- 音声での対話を実現するもの
  - 音声抽出: 人間の音声の認識・抽出
  - 音声認識: 文字起こし
  - テキスト生成: 応答文の生成
  - 音声合成: 応答の音声化
- 対話環境をつくるもの
  - UI: キャラクターや対話内容の描画
  - 画像認識: キャラクターの仕草の生成

本研究はAI基盤構築のノウハウの蓄積も意図しているため，外部サービスは用いずに実現したい. しかしながら, 同様の大規模言語モデル(LLM)をゼロから作り上げるには大きなコストが掛かる.

３．目　標

そこで本研究では，モデルに関しては既存の公開されているものを利用し，ターゲットを「AIサービス基盤の開発」と，それを応用した「AIサービスの開発」に絞り開発を進める.

「AIサービス基盤の開発」では，音声認識，テキスト処理, 音声合成, 音声ブローカーのサービス基盤をオンプレミス環境向けに開発し，音声によるリアルタイムな対話を実現する.「AIサービスの開発」では，開発した基盤を応用し,「山形県に詳しいAIエージェントサービス」を実装する．これらをXRビジネス創出事業におけるXRサービスのプロトタイプ開発と，AIサービス開発をテーマとした教育・研究の足掛かりとすることを目指す.

４．AIサービス基盤の設計と実装

AIサービス基盤は，音声抽出・認識，テキスト処理, 音声合成と, それらを取りまとめる音声ブローカーで構成される(図１). 利用するモデルは，精度と速度のバランスを考慮し選定する．対話の応答速度については，おおよそ1秒程度を目指す[1]. 検証環境には，Intel Core i5-14500，NVIDIA RTX 4060Ti(16GB)，DDR5-5600 64GBのPCを用いる.

* 山形県立産業技術短期大学校庄内校
〒998-0102 山形県酒田市京田三丁目 57-4
* Shonai College of Industry & Technology
3-57-4 Kyoden, Sakata City, Yamagata, 998-0102, Japan

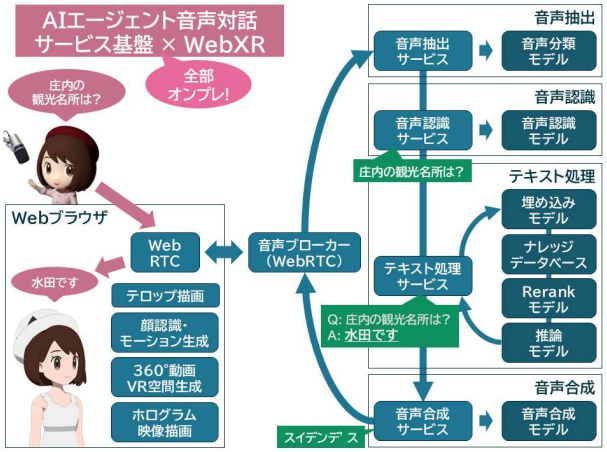

図１　AI エージェント音声対話サービス基盤の構成

4.1 音声抽出サービス

人間の声が含まれていない音声に対して音声認識処理を適用すると，無意味な認識結果が出力されてしまう．したがって，あらかじめ人間の声が含まれているか否かを確認し，人間の声のみを抽出する作業が必要となる．今回は，MediaPipe[2]の Audio Classifier タスクと YamNet モデルを用いて，フレーム単位で音声を分類し，抽出を行うサービスを実装する．発話が完了したかの判断は，無音時間を元に，音声抽出処理の時点で行う．

4.2 音声認識サービス

抽出した音声に対して音声認識処理を適用し，認識結果のテキストを出力する．今回は，音声認識モデルである Nue ASR[3]を用いて音声認識サービスを実装する．また，音声認識処理は，発話中であってもその時点の認識結果が返されるようにする．これにより，後続のサービスでの事前処理や，発話中のテロップの描画などが行えるようになる．

4.3 テキスト処理サービス

認識されたテキストを元に，対話文を生成する処理を行うテキスト処理サービスを実装する．単に推論モデルを呼び出し，応答を生成するだけでも最低限の対話は可能である．しかし，「山形県に詳しいエージェント」といったものを実装したい場合，ファインチューニングなどが必要となり，コストが高くなる．そこで今回は，検索拡張生成(Retrieval Augmented Generation，RAG)の手法を取り入れる．RAG では，質問が来た際にナレッジデータベースから関連する情報を取得し，それを組み合わせて生成を行う．これにより，モデルに手を入れることなく，モデルに含まれていない情報を含めたテキストを生成できる．

RAG 用ナレッジデータベースとしては，検索クエリとなる情報との類似性で検索できるベクトルデータベースを用いる．加えて，テキストからベクトルを得る埋め込みモデル，検索結果を評価する Rerank モデル，生成を行う推論モデルが必要となる．本基盤では，ベクトルデータベースとして Weaviate，埋め込みモデルとして ruri-large[4]，Rerank モデルとして ruri-reranker-large，推論モデルとして Llama-3-ELYZA-JP-8B-GGUF[5]を用いる．また，対話処理や内容の調整，ナレッジの管理を Web インタフェースから一括して行えるよう，LLM アプリ開発プラットフォームの Dify をセルフホスティングした上で導入する．以上を組み合わせたテキスト生成処理の流れは，図２の通りである．

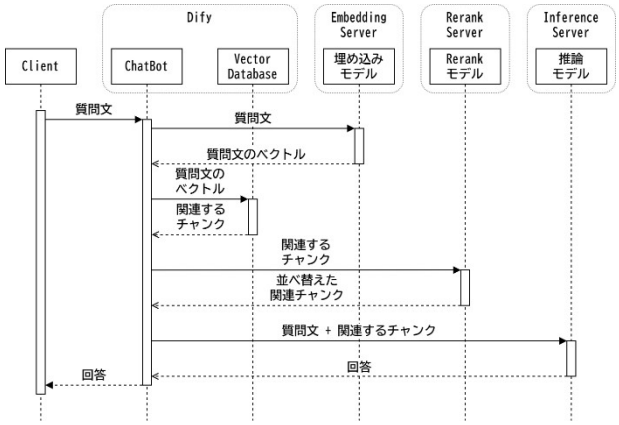

図２　検索拡張生成(RAG)のシーケンス

4.4 音声合成サービス

テキスト生成サービスによって生成された応答文の読み上げ音声の合成を行うサービスを実装する．今回は，音声合成エンジン VOICEVOX を用いてこれを実現する．VOICEVOX では，音声合成クエリに母音と子音の長さの情報が含まれる．この情報はキャラクターのリップシンクに有用であるため，音声と合わせてレスポンスとして返す．また，合成結果は Redis を用いてキャッシュしておく．

4.5 音声ブローカーサービス

WebRTC によるリアルタイム音声通信と，ユーザーと各サービスワーカー間の音声の仲介を行う音声ブローカーサービスを実装する．音声ブローカーは，音声の仲介の他，生成したテキストやリップシンク用発話データを，DataChannel 経由でクラ

イアントに送信する(図３)．また，音声ブローカーとワーカー間は，WebSocket を用いて通信する．

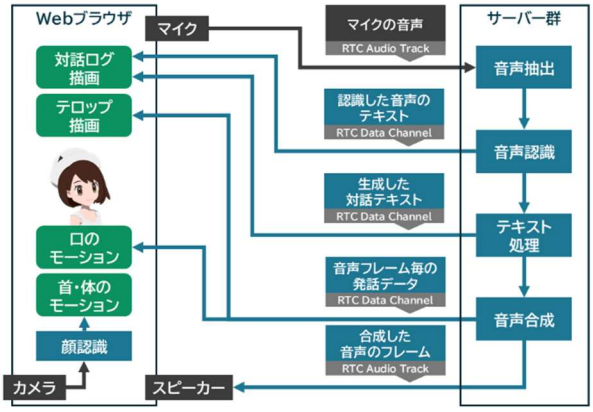

図３　音声ブローカーサービスを経由するデータの流れ

4.6 コンテナ化とサービスディスカバリー

ユーザー数の増加やGPU性能の限界などを考慮すると，複数のサーバーによる分散処理が必要になる可能性が高い．そこで，各サービスワーカーをDocker コンテナ化し，Docker Compose によりワーカー群の一括デプロイを実現する．これらにより，サービス全体のスケーラビリティを確保する．

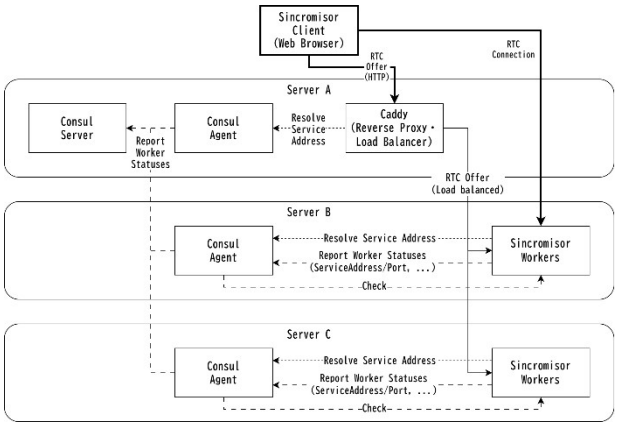

図４　Consul によるサービスディスカバリー

また，サービスワーカーの増加に伴い，サービスワーカーの稼働状況の把握が煩雑になる．そこで本サービス基盤では，サービスディスカバリーの仕組みとして Consul，ロードバランサーとして Caddy を導入する．各ワーカーのアドレスと稼働状況を Consul に集約し，その情報を基に Caddy とワーカー間の通信の振り分ける(図４)．

５．AI サービスの設計と実装

AI サービス基盤の利用例として，「山形県に詳しい AI エージェント対話サービス」を実装する．

5.1 ユーザーインタフェース

ユーザーインタフェースでは，AI エージェントの 3D キャラクターに加え，対話履歴，発話中の音声のテロップを提供する(図５)．キャラクターの描画には Three.js を用い，キャラクターモデルのフォーマットには VRM-1.0 形式を採用する．VRM により，差し替えが容易に行えるようにする．また，スマートフォンや OBS Studio の Browser Source など，さまざまなプラットフォームや用途で利用できるよう，レスポンシブデザインを取り入れる．

図５　ユーザーインタフェース

5.2 キャラクターの仕草の実装

キャラクターの仕草は，主に口と頭，首の動きを実装する．口の動きについては，DataChannel 経由で得られる発話データを基に，Morph Target を制御し表現する．頭と首の動きについては，ブラウザ上で顔認識を行い，それに合わせてボーンを制御し表現する(図 3)．顔認識には，MediaPipe の Face Detector タスクと BlazeFace モデルを用いる．

5.3 山形データベースの構築

山形県に詳しいエージェントに必要な，RAG 用データベースを構築する．今回は，Wikipedia 日本語版のアーカイブから山形県に関連する記事約 7000 件を抽出し，フィルタリングとチャンク化を行った．加えて，記事を基にした Q&A 形式のデータを約 43 万件生成した．生成の際は，ユーザーの質問に近いチャンクをこれらの中から検索し，組み合わせて処理を行う．

6．評　価

6.1 対話の円滑さの評価

本サービス基盤における対話の円滑さは，音声抽出と認識・テキスト処理・音声合成に掛かる時間が重要となる．特に音声認識はレスポンス速度に大きく影響するため，音声認識のベンチマークを重点的に行った．図 6 は，音声認識クエリのターンアラウンドタイムを，認識対象音声の長さごとに計測したものである．音声 1 秒あたりおおよそ 0.1 秒掛かり，音声の長さに応じて線形に増加することが読み取れる．

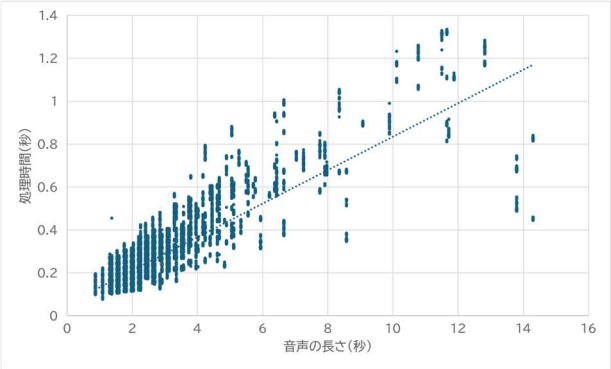

図 6 音声認識クエリのターンアラウンドタイム

加えて，120 件の質問を用意し，RAG なし・ありのテキスト生成クエリのレスポンスタイムをそれぞれ計測した．RAG なしの場合は平均 0.1 秒，RAG ありの場合は平均 0.6 秒であった．また，音声合成クエリのターンアラウンドタイムは，CPU を用いてランダムな名詞を 5 つ読み上げるクエリを実行した場合，平均 1.0 秒であった．これらを合計すると，音声抽出で 0.6 秒，音声認識で 0.5 秒，テキスト生成で 0.1～0.6 秒，音声合成で 1 秒と，おおよそ 2.2～2.8 秒となる．このままでは目標の 1 秒に収まらないため，対話を先読みし生成しておく，発話中に相槌を打つといった機能を追加するなどといった手法の組み合わせが必要である．

また，イベント会場などのノイズが多い環境では，意図しない音声を認識しがちであった．ノイズキャンセリングや指向性マイクなどが必要である．

6.2 応答内容の評価

RAG によりどの程度応答の精度が向上するかを評価するため，正答数をまとめた (表 1)．正答か否かは，人間が実際の情報を確認する形を取った．

表 1　正答数の評価

	正解	一部正解	不正解
RAG あり	83	15	22
RAG なし	31	26	63

RAG なしの場合，無関係の情報のほか，武蔵村山の情報が混ざるといった不具合が多発した．RAG ありの場合は，ナレッジにある情報の正答率は大きく向上した．しかしながら，ナレッジにない情報，特に店舗情報などの誤りが目立った．用途に合ったナレッジの整備が今後重要となると言える．

6.3 実行コスト(GPU)の評価

システム全体で要する VRAM は 16GB 程度となった．VRAM の節約方法としては，モデルの数値精度を下げる方法などがあるが，処理時間が長くなる，精度が低下するといった副作用が生じることが多く，トレードオフとなる．

4．結　言

本研究では，オンプレミス環境を対象とした AI エージェント音声対話サービスとその基盤の実装と評価を行った．研究成果は先述の XR プロジェクトの他，情報通信システム科学生の卒業研究などでも活用されている．

環境音などの意図しない情報が混じりやすい音声認識は，オンプレミス環境での需要が一定程度ある．そのような環境においても手軽に音声認識サービスが利用できるようになったことで，今後の応用が期待できる．

ただし，ローカル LLM で実現可能なものには一定の限界がある．サービス企画・開発は，その限界を理解した上で行うことが重要である．

文　献

1) 長岡千賀: 音声対話における交替潜時が対人認知に及ぼす影響, ヒューマンインタフェースシンポジウム 2002 論文集, pp.171-174(2002)
2) Google LLC.: MediaPipe Solutions guide, https://ai.google.dev/edge/mediapipe/solutions/guide (2024)
3) Hono, et al.: Integrating Pre-Trained Speech and Language Models for End-to-End Speech Recognition, *Findings of the Association for Computational Linguistics ACL 2024*, pp.13289-13305 (2024)
4) Hayato Tsukagoshi, Ryohei Sasano: Ruri: Japanese General Text Embeddings, https://arxiv.org/abs/2409.07737 (2024)
5) Masato Hirakawa, et al.: elyza/Llama-3-ELYZA-JP-8B, https://huggingface.co/elyza/Llama-3-ELYZA-JP-8B (2024)