AI-based VR Conversation With Yifan

A Project
Presented to
XR:MTL, Galilei, Ubisoft, ConcordAI, Concordia University
Montreal, QC, Canada

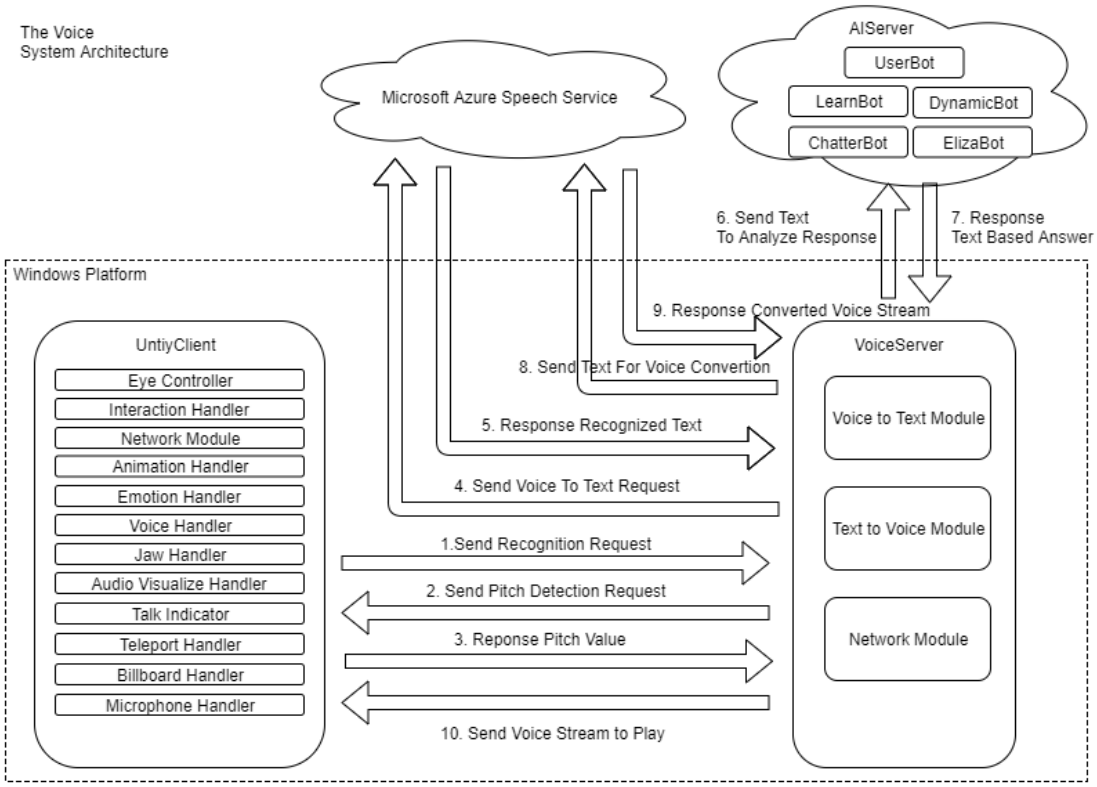
ConcordAI 12-Week Program Project

Team: The Voice
Shuo Pang / Korhan Akcura

Introduction

This is an immersive dialogue experience based on technologies of machine learning, virtual reality, natural interaction, speech recognition, and voice synthesis, etc.
In a beautiful park, you can chat with your friend at zero distance. Your friend can express their feelings for your speech. Your friend knows everything, no matter what question you ask, she will answer it correctly. This is a dialogue experience based on dialogue topic data training, real-time web search, and advanced techniques such as natural language synthesis.

System Architecture



Problems We Solved

1. For the solutions of speech recognition, we did some research on this, finally select Microsoft Azure Cognitive Service because of better performance, but problem is it's not easy to use with Unity3d, so we find a solution that creates a VoiceServer which is individual process running on windows platform to connect Azure Service, that makes extra work but worth to do it. What is more, the quality of voice to text conversation was dependent on noise at the environment and the speaker's accent.
2. For the facial animation to perform different emotions, normally model should have blendshapes for emotion animating and blending, but the resources we got is not very professional, only have bones on face, so we can only control bones to perform different emotion, that causes a little bit unnatural and low amount of emotions, so we only implement a happy and sad face.
3. We use microphone sampling to the player's voice pitch to detect how much it should perform on certain emotion, but it's not work like we expected, because we considered the pitch level and keyword together to determine results, for example, "I am happy" with 100 pitch will perform a 100% happy face, but "I am not happy" with 100 pitch still perform 100% happy face with our current synonym based implementation. In the future iterations the negations can be detected.
4. We had a problem with finding good quality conversation data. For example, using Cornell Movie-Dialogs Corpus and Ubuntu Dialog Corpus did not give relevant results so we excluded them from our training. For this reason we decided to keep our initial data set small. Because the data was limited, during the conversation, player always got the same responses, for this reason, we develop a module called "Dynamic bot", to implement a runtime search function when player send any texts, if player said "Who is Michael Jackson", "Dynamic bot" will search on Internet immediately and get an abstract result. This way our bot is only trained with relevant information.
5. Because of we connect lots of service during conversation, caused an obvious latency, to eliminate it, on backend, we use MongoDB instead of SQLite, which has faster response time, and on client side, we add some redefined sentences, when the player says something, character will send a random thinking sentence before backend responses correct answer, this will make the player feels like the character responses quicker to archive better player experience.

Client Structure Design

Interactive Handler
This module contains a function that handle all interact behaviors, such as move forward, turning, move backward, range detect, microphone distance detect, etc.

Emotion Handler
Controls the face bones on a model, convert text-based command to facial animation.

Animation Handler
This is the controller of model’s animator, have a different kind of animation state, and a simple state machine to implement full body and upper body animations. In animator, there are different animation layers to implement blending.

Voice Handler
This module receive voice stream and other commands from VoiceServer by the local network, parse them and play voice sound in unity as 3d sound.

Audio Visualization
This module able to detect player’s voice highest pitch during talks then sends to the backend to implement emotions with different weights.

AIServer Design

AI Server consists of 4 individual response models, 1 emotion model and an utils package:

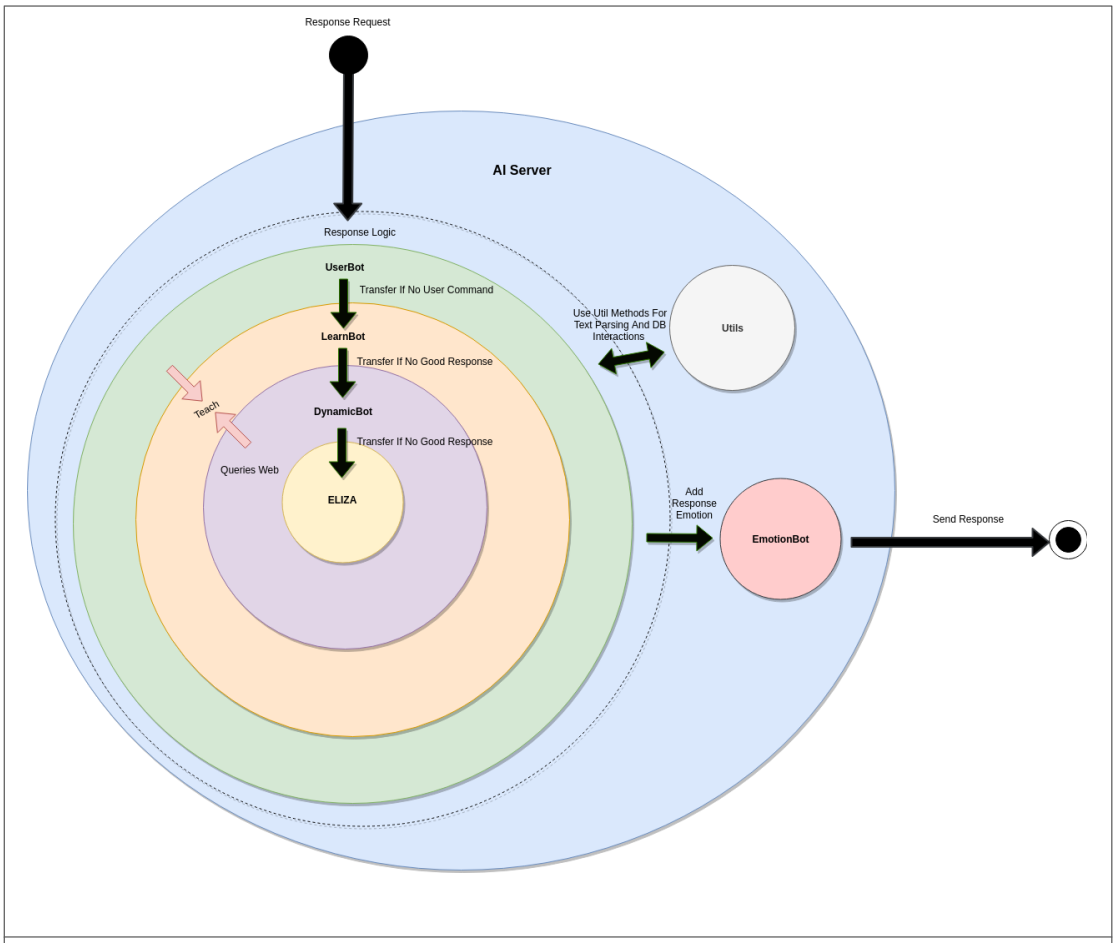


Fig II. AI Server Architecture

You can imagine this design to be functioning like a decorator pattern. Even though response bots (models) do not contain each other, the lower bots are covered with the response capacities of the higher ones. But if the higher level bot is able to provide an answer that satisfies its good answer criteria, the answer is directly used without going to the lower bot. We used an hybrid approach on machine learning methods.

Server.py

This is the main server code that runs the the application. It is a Flask application that runs in port 5000 by default.

utils/*
The utils package to handle text parsing and database interactions.

models/user/*
This model contains the response algorithm for specific user commands. It contains the logic to response to name of the user and to train LearnBot by user's input. The UserBot uses an instance of this model. It is the first response mechanism of the AI server. The command to teach something is "I want you to learn this.".

models/chatterbot/*
This model contains an algorithm to that can be dynamically trained by data input. It is a lazy learning algorithm which determines the best response by a distance calculation on learned data. For this we are using a ChatterBot version that we customized based on our need. The LearnBot uses an instance of this model. It is the second response mechanism of the AI server. It is initially trained by existing datasets. It continuously learns from UserBot and DynamicBot. It will use MongoDB if it finds an active instance. If not, it will generate a SQLite3 file.

models/webquery/*
This model contains some algorithms to query data from the web base on the user's request. Thanks to this model, we have an infinite amount of training data which can be obtained when needed. The DynamicBot uses an instance of this model. It is the third response mechanism of the AI server. It continuously trains LearnBot. Any improper response this bot provides can be later corrected through the commands of UserBot.

models/eliza/*
This model contains an algorithm to mimick classic 1966 implementation of ELIZA natural language conversation program. We use this algorithm to always have something to say as worse case. This was the first response model we integrated into our system. This provided us with worst case responses which we could take as base and improve on top. ElizaBot uses an instance of this model. It is the fourth response mechanism of the AI server. It provides the worst case responses in case other mechanisms are not able to generate a response that satisfies their criteria. It doesn't train LeanBot since this model's responses are too generic.

models/emotion/*
This model contains an algorithm to detect emotions based on synonyms of happy and sad. It detects the intensity of the emotion based on the pitch value that is provided by the client. If the emotion is not detected as

happy or sad, it is returned as natural. EmotionBot uses an instance of this model. All of the generated responses passes through this bot to get the emotion parameters added.

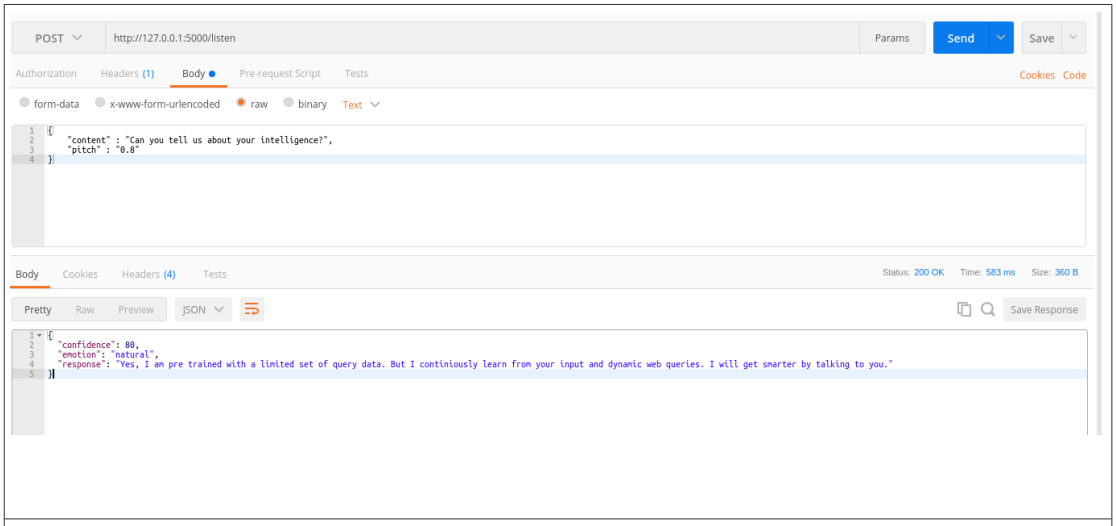


Fig III. Example Request and Response

```
{
   "content": "I would be happy to learn about your intelligence algorithm?",
   "pitch": "0.8"
}
```

Example I. Sample Request

```
{
   "confidence": 80,
   "emotion": "happy",
   "response": "I am pre trained with a limited set of query data. But I continuously learn from your input and dynamic web queries. I will get smarter by talking to you."
}
```

Example II. Sample Response

Voice Server Design

Voice Server have 4 modules:

Speech-To-Text
Text-To-Speech
There are all connected to MS Azure service.

Backend Connection
This module will send converted text content to our backend to analyzing, then get responses with correct answer and other parameters.

Network Module
This module will send messages between unity project and this process as a local network service.

Appendix

Source Code:
https://github.com/yifnzhao/The-Voice

Live Demo:
Between following timestamp: 00:54:12-01:19:00
https://www.facebook.com/ConcordAI/videos/271039787151563/