The inner workings of ZiroAI - Documentation

Dear reader,

I hope this documentation find you well and not because something is broken.

I have organised the documentation in handy little parts so you can easily find what you’re looking for.

Happy reading,

Ruby

# 1 Chatbot

The chatbot of the ZiroPay app consist of 2 parts. An intent recognition model, to see what the user could want. And if that fails a Large Language Model, TinyLlama by Meta.

## 1.1 deployment

The chatbot and it’s code can be found in the repository. This code along with the dockerfile in that folder should be deployed together. Upon running the dockerfile the server starts and the chatbot is live. By doing so 2 endpoints are exposed: /chat and /voice. The first taking a string and the latter taking an audio file, more on that later.

These endpoints then respond with the chatbot’s generation.

## 1.2 adding intents

To add more predefined responses to the chatbot you can add them to the intents.json file, following the structure found there. The tag is an internal tag used as a label for the prompt, this is also what our first model predicts. The patterns are examples of what the user could say, the more variety in these patterns, the better the model becomes at predicting intent. Lastly the responses are what the chatbot could reply, one of these is picked at random. With a minimum of one response defined.

After you’ve added your desired conversation to the chatbot. You need to run [train.py](http://train.py), which takes the intents file, trains the neural network and saves the parameters of it in data.pth. Which is automatically loaded.

To run [train.py](http://train.py) you will need to run from an environment with PyTorch installed.

## 1.3 connecting to systems within the app

To connect the chatbot with other systems in the app for checking account balance or other tasks. You need to add some logic to the handle\_intent found in intent\_recognition.py.

Here you can add if statements or a switch to check the incoming “intent” parameter against whichever intent you need. For example checking against “balance” and then adding maybe an API call to another part of the app. You can then just return the acquired data with your response and the system will handle the rest.

## 1.4 changing the language model.

As the language model for the chatbot is not fine tuned, it can be easily swapped out for a different model found on HuggingFace. However since the TinyLlama model uses a pipeline whilst most models use a separate tokeniser/model, some slight changes are necessary in the logic of generating a response.

## 1.5 Recording audio for voice commands

There are some constraints on what type of audio the chatbot can handle. Mainly due to the constraints that the whisper model lays out. It should be in single channel, mono. Have a sampling rate of 16kHZ and a .wav file.

## 1.6 Every file and briefly what it does.

* [App.py](http://App.py) starts the Flask app and exposes endpoints
* Data\_management.py is a helper class used for saving chats and interactions for further use or analysis
* Intent\_recognition.py runs the model on new prompts for a prediction and handles the intents
* Main\_chat.py combines all the files and handles the interactions.
* [Model.py](http://Model.py) is a basic neural network class is high forms our intent recognition
* Nltk\_utils.py provides some functions from the natural language toolkit. Used in processing our texts for intent recognition.
* Response\_generation.py holds the functions that interact with the model to generate responses.
* Speech\_recognition.py transcribes audio files into text.

# 2 Split-the-bill

The split-the-bill features aims to take a receipt and output the items on that receipt.

Unfortunately I did not manage an implementation of this. However I suggest you link the OCR flutter implementation with n8n to ask chatgpt to identify the entities in a transcriptions. Just pass along the transcription and your desired json format to process it into the app afterwards for a proof of concept for the investors.

However in the future I suggest you make use of Graph Convolutional Neural Networks. These add spatial awareness to images which means you could use it to rearrange the OCR transcripts in a logical way instead of trying to identify all the parts. Then processing it becomes a lot easier.