國立臺灣科技大學電機工程系

112學年度第一學期實務專題

專題成果報告

AI Agent

組 員： 姓名：周柏宇 學號：B11032006

姓名：康閎茗 學號：B11032055

指導老師：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(簽名)

中華民國 113年 02月 27日

### 貢獻

周柏宇: 50% 康閎茗: 50%

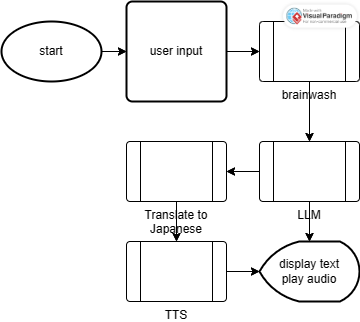
## 簡介

[AI Agent](https://github.com/MeowOhMeow/ai-agent/tree/new) 是一個交互類似 chatGPT 的 side project，這個 project 能夠根據用戶輸入的中文產生中文文字回應，以及日文配音。這個 project 使用了 openAI api、VITS 取得回應並轉成日文語音。

## 流程

1. 文字、語音(TODO)輸入：接受使用者文字、語音的輸入。
2. 角色催眠：利用 prompt engineering 建構 Agent 的性格。
3. LLM：使用本地端(TODO)或線上api服務生成回應。
4. 後續處理：
   * 把LLM回應的中文翻譯成日文，以提供TTS(text-to-speech)生成語音。有兩種方式，使用本地端或線上翻譯服務。
   * 使用VITS生成語音。
5. 輸出：中文文字搭配日文語音。

* GUI：能實現基本功能的 user interface。
* CLI：功能等同於 GUI，但是運行在 cmd。



## 

## 關鍵技術

* LLM：[OpenAI API](https://openai.com/blog/openai-api)
* 去背景噪音：[facebookresearch/demucs](https://github.com/facebookresearch/demucs)
* 分群：nemo/vad\_multilingual\_marblenet, nemo/titanet\_large, nemo/diar\_msdd\_telephonic [NVIDIA/NeMo: NeMo: a framework for generative AI (github.com)](https://github.com/NVIDIA/NeMo/tree/main)
* 語音輸入：whisper(TODO)
* 翻譯：
  + [translators · PyPI](https://pypi.org/project/translators/)
  + [facebook/nllb-200-distilled-600M · Hugging Face](https://huggingface.co/facebook/nllb-200-distilled-600M)
* 語音合成：[Plachtaa/VITS-fast-fine-tuning](https://github.com/Plachtaa/VITS-fast-fine-tuning/tree/main)

## 使用方法

1. 在 VITS 下 build [Plachtaa/VITS-fast-fine-tuning](https://github.com/Plachtaa/VITS-fast-fine-tuning/tree/main)，具體請參考 [LOCAL.md](https://github.com/Plachtaa/VITS-fast-fine-tuning/blob/main/LOCAL.md)
2. 在 cmd 輸入 conda env update environment.yml
3. 在 VITS\output\_models 放入 finetune\_speaker.json, G\_latest.pth [載點 - Google drive](https://drive.google.com/drive/folders/1-40Dd3CRDzVGoL5gdRj39oN_uWKdOvp2?usp=sharing)
4. 在 GPT\openai\_key 放入 openai api key
5. run GUI.py 或 CLI.py

如果單純想使用語音可以用 [Plachtaa/VITS-fast-fine-tuning](https://github.com/Plachtaa/VITS-fast-fine-tuning/tree/main) 的release，搭配第三點的模型

## 限制

歷史紀錄長度上限被 hard coded 成 5

## 其他參考

[zixiiu/Digital\_Life\_Server](https://github.com/zixiiu/Digital_Life_Server)

## 實作

1. 資料前處理：
   1. 從各種途徑找到聲音角色的影片、音檔。
   2. 將所有影片、聲音轉換成 mp3 格式。
   3. 將 mp3 轉換成 mono(單聲道) 的 wav 格式。
   4. 使用 demucs 降躁，僅保留人聲。
   5. 使用 nemo 套件做 VAD 動作，這步會從音檔中判斷出哪些speakers在甚麼時候說話。其中TitaNet，是用歐洲語系訓練出來的，在日語分類上的結果可能稍差，這個反應在輸出成果。
   6. 使用上一步產生的 rttm file，將每一個角色的聲音剪出來。
   7. 手動取得其中一集目標角色的音檔，再使用 speakerverification\_en\_titanet\_large 判斷兩者是否為同一個 speaker，每集僅留下相同speaker 的檔案。
2. 訓練 VITS (使用 Plachtaa 的實作)
   1. 音檔再使用一次 demucs ，一樣僅保留人聲。
   2. 透過 whisper 產生日文文字，並切割音檔成較小的片段為後續資料標註做準備。
   3. 使用 [pyopenjtalk · PyPI](https://pypi.org/project/pyopenjtalk/) 產生 phoeneme 做為資料標註。
   4. 將處理過的音檔以及 phoeneme 丟入模型做 finetune。

使用 3060 laptop 6GB，訓練2+小時可以達到目前的效果。

1. 實作 interface
   1. OpenAI\_API
      1. \_\_call\_\_：使用者輸入 + 催眠 + 歷史紀錄。
      2. regenerate\_response：將上一次的 prompt 重新丟給 openai api 生成新的回應。
      3. get\_last\_response：給 TTS 重新生成語音用。
   2. Translator
      1. V1：使用 [translators · PyPI](https://pypi.org/project/translators/)。
      2. V2：透過 [transformers · PyPI](https://pypi.org/project/transformers/) 使用 [facebook/nllb-200-distilled-600M · Hugging Face](https://huggingface.co/facebook/nllb-200-distilled-600M) 進行翻譯。
   3. API
      1. \_\_call\_\_：輸入文字 prompt，回傳中文回應、日文語音。
      2. regenerate\_audio：重新生成語音。
      3. regenerate\_response：重新生成回應、語音。
   4. TransTTS：僅使用Translator、TTS。
   5. GUI：使用 tkinter 實作。
   6. CLI：輸入 ‘help’ 可以查詢指令細節。

### 改進方向

* 資料前處理產生的音檔質量不佳，可以考慮人工剪輯或使用其他 VAD 技術。
* [facebookresearch/demucs](https://github.com/facebookresearch/demucs) 的 600M distilled 翻譯質量不佳，所以我們後來改用線上服務的選項，可以尋找其他開源模型，或者自行訓練。
* [VITS](https://arxiv.org/abs/2106.06103) 是2021/06的論文，距離現在已經2年半，應該有其他較新的模型會有更好的效果，但從頭訓練語音模型需要大量的算力，這比較難實現。

### 其他

PPTs 裡面是組員們互相報告各個技術原理的powerpoints。