南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目： 基于 ChatGPT 的语音聊天机器人

姓　　名： 李怀武

学　　号： 11911425

系　　别： 计算机科学与工程系

专　　业： 计算机科学与技术

指导教师： 刘 江

2023 年　5 月　8 日

诚信承诺书

1.本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2.除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3.本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4.在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：李怀武

2023 年 5 月 8 日

基于 ChatGPT 的语音聊天机器人

李怀武

（计算机科学与工程系　指导教师：刘江）

[摘要]：在传统的语音聊天机器人中，例如 Siri 或者是小爱同学通常只能作为简单的语音助手，做一些查找信息和拨打电话等简单的任务工作，不能回答用户广泛的问题或者与用户进行深度的聊天交流，为了解决传统语音聊天机器人存在的问题，基于 OpenAI 公司提供的 ChatGPT 和微软 Azure 的 API 来开发一款跨平台智能语音聊天机器人。在技术上，本项目采用了 Electron 作为应用骨架，Vue.js 框架和 ElementPlus 组件库开发前端界面，使用了 Flask 作为后端框架。此外，本文还扩展了 AI 的使用场景，使得用户不仅可以通过文本和 AI 交互，还可以在不方便输入文笔的情况下时通过语音交流和 AI 进行实时互动，同时要比主流的语音助手 AI 更加智能。

[关键词]：ChatGPT; Azure; Electron;Vue; Flask

目录

1.绪论

1.1 课题背景和意义

随着人们对生活质量和便捷生活要求不断提高，对于和更加智能的聊天机器人交流的需求也不断增长。本项目的目的是可以帮助人们更加方便地使用聊天AI技术，使得人们可以通过人类最自然的交流方式来和具有强大功能的ChatGPT聊天机器人交互，这样的设计不仅可以拓展聊天机器人的使用场景，提高用户的交互体验，同时和也让用户能更高效的和聊天机器人进行沟通，让用户可以在不方便进行文本输入时的使用场景下和AI进行实时互动。试想用户可以在开车，洗澡，吃饭时通过语音和ChatGPT聊天机器人沟通，是不是使得人们的生活更方便了呢。同时语音交流和文字交流不同，语音交流相比于冷冰冰的文字，能给用户更加富有感情的回复，满足特定群体用户对于情感和沟通的需求。例如孤寡老人或者抑郁症患者都可以使用该聊天机器人来满足沟通和情感需求。使得用户能够享受到科技带来的便利。

此外，本项目采用了跨平台的软件开发技术，使得用户可以在不同的操作系统上使用该软件（例如 Windows，MacOS和Linux操作系统），进一步增强了用户在使用本软件时的便捷性，同时也使得盲人和老年人等不方便进行文字输入的用户群体得以使用ChatGPT聊天机器人[11]，使得用户群体更加多元化，多样化。本项目的开发不仅仅对于促进AI聊天软件技术[9]x的发展有重要意义而且使得AI技术更加普及也具有重大意义。

1.1.1 已有工作的不足

市面上有许多的语音聊天机器人[17],例如苹果的siri，微软的小娜，小米的小爱同学。这些机器人的也非常的智能化[10]，能够在用户的命令下执行一些简单的功能。虽然这些机器人能够在特定的环境下执行一些任务[20]，给用户带来一定程度的帮助，但是他们还是存在一些不足之处。

首先，这些语音聊天机器人的识别和理解能力存在一定的缺陷。有时候用户的指令无法被聊天机器人识别，因为这些聊天机器人是基于预先设定的模板和规则来进行识别的，如果用户提供的输入不符合规则，那么用户的指令将不会得到识别，但是基于ChatGPT的聊天机器人由于是GPT模型对用户的输入进行解析和

理解，其分析能力要大大强于传统的聊天机器人，在很多时候能给出恰当和准确的回答。相较于传统模型而言，ChatGPT 在处理语音输入时存在错误识别和错误理解的情况将大大减少，用户从而有更好的使用体验。传统聊天机器人和用户交流的内容也相对简单，很难满足用户所有的功能需求。大多数聊天机器人只能在特定的几个功能上和用户进行沟通，如果用户想针对某个专业问题进行深度沟通时，传统的聊天机器人则不能做到，ChatGPT 则可以在多个专业领域上和用户进行对话，例如科学，技术，经济，健康，社会，法律，医学等内容，大大的丰富了用户的沟通使用体验。

1.1.2 面临的技术问题

从技术角度来看，本项目采用了一系列的技术来实现语音聊天机器人。其中，语音技术方面使用了微软 Azure 的语音合成[3]和识别 API[4]，首先是 API 对接上是一个难点，需要使用后端的 Flask[2]框架将 API 集成在后端之中，使得用户在前端操作时发送的 http 请求能够被后端接收到，后端根据用户的输入获取 API 相应的回答。这些都是后端所需要考虑的技术问题。

例如在集成语音识别 API 时，存在如下技术问题。首先开发者需要在后端环境中配置好相应的环境变量，以及在后端中设置好相应的参数，之后将相应的配置和麦克风音频输入传递给 Azure 语音识别 API，通过单次的语音识别，API 会将结果返回给后端，结果中会有相应的识别文字以及相关的细节信息，后端会进一步的将相应的信息提取出来进行下一步的处理。除此之外还有可能会有识别失败的情况，API 也会返回相应的错误原因。

而在语音识别得到结果之后，开发者还需要将获取到的结果信息转化为可以使用的字符串信息。并且将相应的信息传递给 ChatGPT API，之后会获取到 ChatGPT 返回的回答。此时如果用户使用到的是语音模式，则还需要进一步的调用 Azure 的语音合成 API，此时则需要再后端设定好相应的参数，例如使用的语言和使用的语音包。之后则会通过语音将 ChatGPT 相应的回答通过语音播放出来，而如果使用的是文字模式，则并不会调用语音合成 API。如何设计好后端的接口以及相应 ChatGPT 和语音合成 API[8]则是另一大技术问题和难点。

而在前端界面，需要使用 Vue.js[5]，Electron.js[6]和 elementPlus 等技术框架，需要将相应的技术整合在一起并实现一个完成的聊天机器人[18]界面，

如何解决不同技术之间的兼容性问题，以及保证前后端之间通过 HTTP 请求数据交互的问题成为本项目的另一大技术问题。

1.2 主要贡献

本项目采用了先进的语音合成技术和聊天 AI 模型，结合了 OpenAI 的 ChatGPT[1]和微软 Azure 的语音合成和识别 API，使得软件可以给用户提供高质量的语音识别和合成技术，还保证了模型能给出高质量的回答，因此用户在使用软件时才会有高质量的体验。同时因为采用了 Electron.js[7]作为应用的骨架，Electron 集成了 Chromium 内核和 Node.js 来使得用户可以使用前端技术开发桌面应用[12]。其一大优点是跨平台性，Electron 开发的应用能在 macOS，Windows 和 Linux 三大主流系统上运行。同时 Electron 框架[14]使得开发者开发时可以使用熟悉的 Web 技术开发桌面应用，本次开发技术使用了 Vue.js 和 ElementPlus[6]作为前端开发技术。Vue.js 是一种 MVVM 架构的渐进式框架[13]，使得开发者既能简单上手开发项目还保障了项目运行的高性能，而 ElementPlus 是基于 Vue3 的组件库，能给应用前端界面提供丰富的组件来构建美观的界面。而 Flask 框架是一种基于 python 语言的后端框架，通过 Flask 框架开发后端能够集成 ChatGPT 和 Azure 语音技术 API。

本项目之所以采取前后端分离架构来开发程序，是因为前后端分离架构开发有如下几个优点。第一是前后端分离开发架构可以提高开发效率和灵活性，开发时前端后端的代码不会互相影响。此外，前后端分离开发还可以降低程序复杂度和维护成本，因为前后端各自独立，开发时前后端可以采用不同的技术框架，增加开发的便捷性，同时使得项目开发时的代码耦合度降低。因此既保证了程序开发的高质量和高效率，还为未来的维护和扩展提供了更多的可能性。

1.3 论文结构

本论文包括绪论、设计方法和开发流程、系统详细实现细节、系统测试和展示、总结和展望等五个主要部分。

第一章绪论部分包括课题背景和意义、主要贡献以及论文结构的介绍。

第二章设计方法和开发流程部分包括系统架构、系统功能以及开发流程的介绍。

第三章系统详细实现细节部分包括前端实现细节和后端实现细节的介绍。

第四章系统测试和展示部分介绍了系统的测试和展示情况。

第五章总结和展望部分对本论文的工作进行总结，并对未来的工作展望。

最后，本论文包括参考文献、附录以及致谢等部分。

2. 设计方法和开发流程

本项目的系统设计基于瀑布模型将开发流程分为五个阶段，分别是需求定义，系统设计，实现与单元测试，集成与系统测试[19]和运行与维护。首先需要通过需求定义找到系统开发的目标和描述，接下来将结合硬件软件对系统进行设计，之后对系统中每个模块单元进行实现和测试，之后再将所有模块集成在一起，并对系统进行测试，最后还需要对系统进行运行和维护，发现问题就返回之前的步骤重新执行各个阶段的工作直到开发完成。

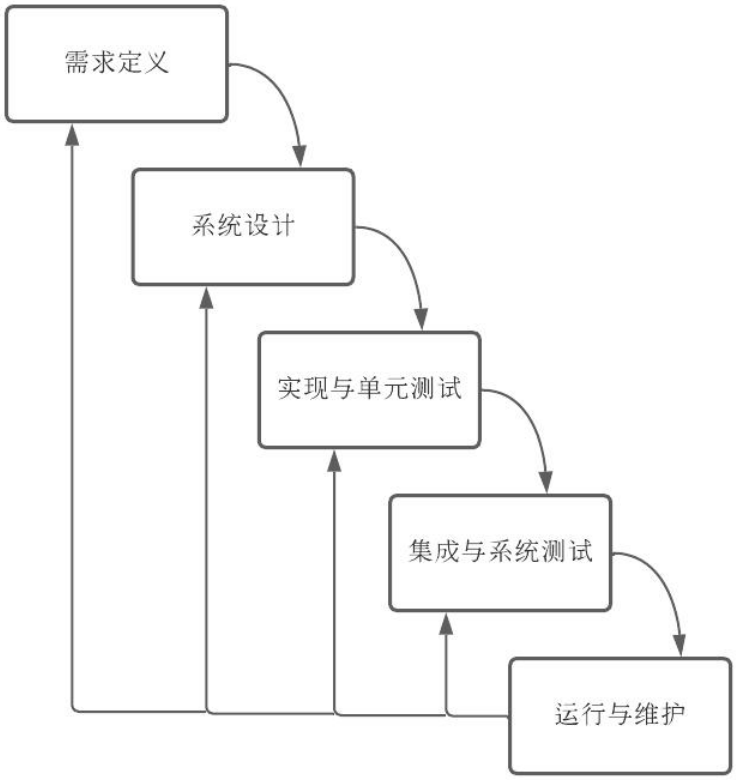

图 1　瀑布模型图

2.1 需求分析

聊天机器人软件[15]开发需要考虑到许多方面的用户需求。而在软件开发时，我们需要考虑到软件的功能性需求和非功能需求。本软件的功能需求考虑到了用户需要使用到的功能，也可以理解为软件需要提供到的服务。本软件的功能包括用户可以使用多种语言包括中文，英文来使用软件，用户的语音输入能够被语音识别功能转化为文字，机器人的文字输出能够被语音合成功能转化为语音，而用

户可以依照需要沟通的内容而选择不同的模型进行交流，聊天功能中用户可以使用 Davinci003 模型来进行日常对话的沟通，而在问答功能中用户可以使用 GPT3.5-Turbo 模型来解决问题的提问与回答的需求。

除了软件开发时我们需要关注功能性需求，我们还需要关注非功能需求。非功能需求是关于本系统实现过程中对软件的性能，安全性以及可用性的需求。本软件的非功能需求第一是需要系统的稳定性，用户需要能够在不同的网络状态下都能正常使用本软件，前后端的网络沟通以及 API 访问皆需正常能使用。同时还要求系统的性能性，合理的系统设计能保证实现系统的主要功能占用资源不会过多，用户能够在不同条件的终端下正常使用本软件。

图 2　用户需求图

2.2 系统设计

2.2.1 系统架构设计

在系统设计时，不仅仅需要考虑功能性需求还需要考虑到非功能性需求。例如软件系统的性能，安全性，可维护性以及可用性都需要在设计中考虑。本项目采取了客户端—服务端体系架构不仅完成了用户的功能性需求还最大程度的考虑了非功能性需求。本项目的前端应用采用 Electron+Vue 架构，而后端采用的

是 Flask 集成 OpenAI ChatGPT 和 Azure 语音 API，前端后端的交流通过 HTTP Post 请求沟通。

图 3　系统设计图

使用客户端—服务端架构的好处是放在服务端的功能和客户端的功能是分离开的，这样如果有多个客户端时，只需要一个服务端就可以给多个客户端提供服务，而如果客户端功能和服务端是整体的，如果有多个客户端时就需要有多个服务端提供服务，这样便造成了资源浪费的情况。但是客户端—服务端也会有缺点便是如果服务端不能正常工作则相应的所有的客户端也无法正常工作，因此为了解决这个问题本项目也可以在本地启动服务器来代替云端服务器的作用，只需要将云端服务器的 ip 地址和端口号替换成本地服务器的 ip 地址和端口号即可。

2.2.2 系统功能设计

在满足软件的非功能需求之后设计之后，本软件需要按照功能需求来设计单元和整体系统。在之前的需求分析中我们知道本软件有如下几个需求，首先需要满足中英文双语的沟通功能，这需要我们设计一个语言单元来控制，而用户的语音之所以可以被转换为文字，这需要我们设计一个语音识别单元，之后我们需要设计一个语音合成单元使得机器人的文字输出能转化为语音播放，最后还需要设计模型单元，模型单元的作用在于可以按照用户的需求切换用户使用的模型，而且我们需要按照不同的模型设计相应的子单元，这样我们得到了总体的功能单元设计图。

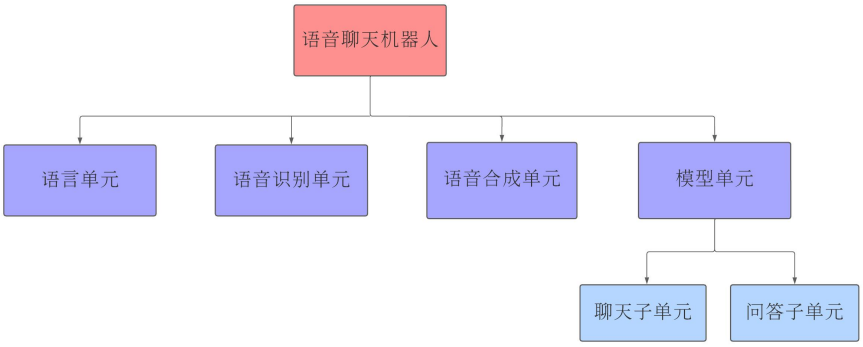

图 4 功能单元设计图

语言模块的作用是用户可以通过语言单元来控制是通过中文进行交流还是使用英文进行交流，在后续的对话中 ChatGPT 会以英文或者中文进行回答。用户也可以在不选择初始语言的情况进行沟通，这样的话 ChatGPT 处于默认设定模式会按照用户的输入来判断使用什么语言进行回答。

语言识别单元的作用是在用户使用语音聊天的时候将语音识别并转化为对应语言的字符串，之后语音识别单元会将识别好的字符串传递给后续的模型单元进行下一步处理，根据用户使用的模式不同，语音识别单元会将相应字符串传给不同的模型模块。而当用户使用文字沟通功能的时候，语音识别单元则不工作，相应文字内容会跳过语音识别单元传给后续的模型单元。

语音合成单元的作用是用户在使用语音聊天的时候将文字转化为相应语言的语音，并播放给用户。语音合成单元承接在模型单元之后，如果用户使用的模式是语音沟通模式，则模型单元会将从 API 获取到的文字信息传递给语音合成单元，如果用户使用的模式是文字沟通模式，则相应的文字信息会直接传递给软件前端显示出来，则并不会传递给语音合成单元。

在模型单元中，用户可以选择以聊天为主的模型和以问答为主的模型，以满足用户对不同模型的需求。同时由于不同的模型调用 API 的方式以及输入输出的方式不同，根据不同的模型则需要设计不同的子模块，前序单元的输入在进入模型单元之后需要按照模型和功能的分流给不同的子模块来进行处理，而子模块的输出则会传递给模型单元进行统一处理输出传递给后续单元。

同时在完成各单元的逻辑设计之后，还需要对软件的前端界面进行设计。在软件前端中主要分为三个部分，第一个部分是左侧的功能栏，相应的可以交互的功能，例如语言选择，模型选择，聊天模式以及问答模式等功能会放在左侧的功

能栏中。第二个部分是聊天显示栏，相应的聊天记录会以文本的形式显示在聊天显示栏中，其中 ChatGPT 的回答和用户的回答被分开展示出来。第三个部分是书输入栏，输入栏的位置位于聊天显示栏下方，在输入栏中用户可以文字输入或者语音输入，相应的文本框以及按钮都会在输入栏中。

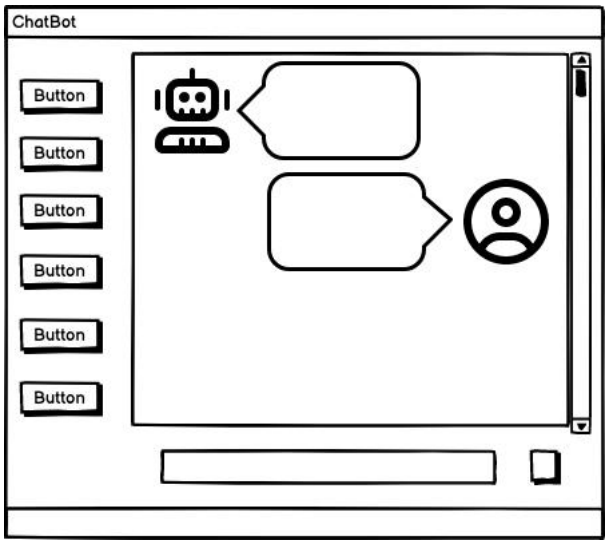

图 5　前端设计示意图

3. 系统详细实现

在系统详细实现细节章节中，将会系统实现细节详细描述，分别是系统开发流程，单元实现两个部分。

3.1 系统开发流程

首先我将阐述系统开发流程，在完成了需求定义和系统设计之后，将进行单元的实现和测试部分，在确保所有单元都通过测试后再进行系统的集成，系统的集成会将不同的单元集成为整体，并对系统整体的功能性做测试。如果系统的功能性测试通过，则后续会进行相应代码的运行和维护,确保软件在运行过程中不会出现问题。在实现与单元测试部分中需要对语言单元，语音识别单元，语音合成单元以及模型单元进行实现和测试，而在集成与系统测试部分需要将上述单元集成在前端和后端中，再对系统进行整体测试。

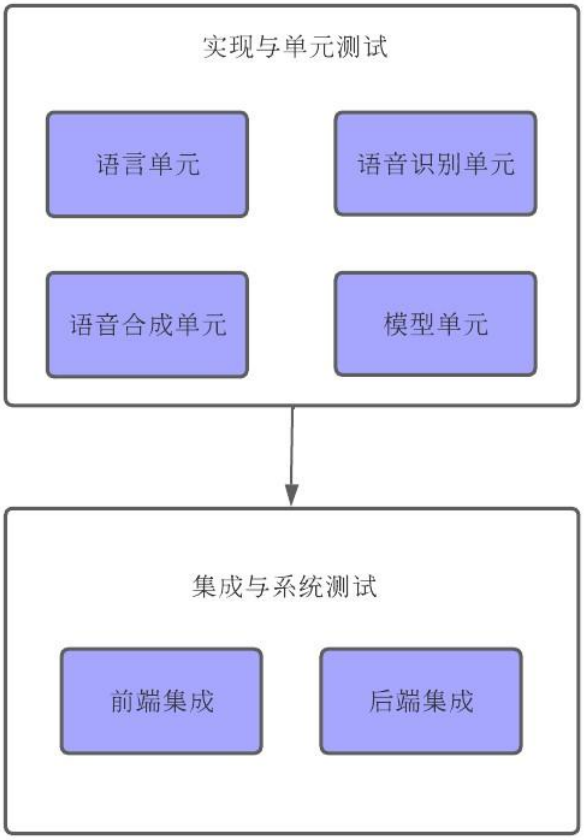

图 6　系统开发测试流程图

3.2 单元实现

本小节中会介绍各个单元的介绍和实现过程，在每个单元实现模块会讲述其作用和实现思路并且会展示相应的样例代码。同时还会讲述相应单元在系统中的作用和工作原理，以便更好的理解各个单元在系统中的功能。

3.2.1 语音识别单元

本项目的语音识别单元是基于微软的 Azure Cognitive 服务提供的语音转文字 API。使用 Azure 的语音识别 API 的一大优点是可以实时的进行语音转文字，而且可以直接调用设备的麦克风来获取用户的音频信息，这样的话用户的使用体验上会比先录制再调用 API 进行音频处理更好。因为实时语音转文字要比录制转文字速度更快，通常在用户说完话时，相应的文本已经生成出来了传递给模型单元进行下一步处理了，用户减少了等待时间。另一个好处是 Azure 语音识别 API 支持不同的语言进行识别，在切换中英文识别时只需要更改 speech_config 中 speech_recognition_language 的语言配置即可，这对多语言支持的语音聊天机器人开发而言是一大优点[16]，减少了冗余代码以及代码耦合度，增加了代码的灵活度以及后续维护的便利性。

本单元的代码逻辑是先从系统中获取到相应配置好的密钥以及使用区域，之后在配置相应的对应的语言和音频配置，后续配置用户使用的麦克风配置。之后通过音频配置和麦克风配置来配置语音识别器对象，并通过语音识别器对象获取麦克风的音频输出以及产生识别出来的结果。该结果包括了识别出来的字符串输出以及识别成功与否的信息。如果成功识别了语音内容，相应语音内容会被传递到模型单元进行下一步处理。

```
speech_config = speechsdk.SpeechConfig(subscription=os.environ.get('SPEECH_KEY'), region=os.environ.get('SPEECH_REGION'))

speech_config.speech_recognition_language="zh-CN"

audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)
speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config, audio_config=audio_config)

print("Speak into your microphone.")
speech_recognition_result = speech_recognizer.recognize_once_async().get()

if speech_recognition_result.reason == speechsdk.ResultReason.RecognizedSpeech:
    print("Recognized: {}".format(speech_recognition_result.text))
elif speech_recognition_result.reason == speechsdk.ResultReason.NoMatch:
    print("No speech could be recognized: {}".format(speech_recognition_result.no_match_details))
elif speech_recognition_result.reason == speechsdk.ResultReason.Canceled:
    cancellation_details = speech_recognition_result.cancellation_details
    print("Speech Recognition canceled: {}".format(cancellation_details.reason))
    if cancellation_details.reason == speechsdk.CancellationReason.Error:
        print("Error details: {}".format(cancellation_details.error_details))
        print("Did you set the speech resource key and region values?")
```

图 7　语音识别样例代码

3.2.2 语音合成单元

本项目的语音合成单元也是基于微软的 Azure Cognitive 服务，微软 Azure 提供了语音合成 API。微软 Azure 语音合成 API 优势在于提供了 110 种语言和 270 种语音包，这样可以既能支持中英双语语音聊天机器人的多语言需求，也能在多种语音包中选择最合适的语音包。Azure 语音合成的另一个优势是在于可以直接通过 API 生成储存在内存中的音频流，这样不需要先生成音频文件之后再对文件进行播放，因此用户能够更快速地听到聊天机器人的回复，减少了等待时间。

本单元的代码逻辑也需要先从系统中获取到相应配置好的密钥以及使用区域，需要注意的是这里获取到的配置和之前语音识别的配置是一样的。之后在语音配置和音频配置中配置相应的语音合成的语音地区以及声音包名字，并且设置好使用的扬声器，后续使用前面配置好的语音配置来生成语音合成器对象，并通过语音合成器对象将模型单元输出的结果直接通过扬声器播放出来。

```
speech_config = speechsdk.SpeechConfig(subscription=os.environ.get('SPEECH_KEY'), region=os.environ.get('SPEECH_REGION'))
audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)
speech_config.speech_synthesis_voice_name='en-US-JennyNeural'
input=""
speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, audio_config=audio_config)
speech_synthesis_result = speech_synthesizer.speak_text_async(input).get()

if speech_synthesis_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:
    print("Speech synthesized for text [{}]".format(input))
elif speech_synthesis_result.reason == speechsdk.ResultReason.Canceled:
    cancellation_details = speech_synthesis_result.cancellation_details
    print("Speech synthesis canceled: {}".format(cancellation_details.reason))
    if cancellation_details.reason == speechsdk.CancellationReason.Error:
        if cancellation_details.error_details:
            print("Error details: {}".format(cancellation_details.error_details))
            print("Did you set the speech resource key and region values?")
```

图 8　语音合成样例代码

3.2.3 模型单元

本项目的语音合成使用的是 OpenAI 提供生成文本尖端语言模型，选择 OpenAI 的语言模型的作为本项目的模型单元的核心是因为 OpenAI 公司的 ChatGPT 具有世界领先的 NLP 技术，并且其提供的 ChatGPT API 也能很好的和微软的 Azure 语音合成和语音识别 API 兼容，因为两者的 API 都可以使用 python 语言进行调用，因此采用后端也是 python 语言构建的 Flask 框架就能很好地将两个公司提供的服务集成起来，使得开发起来十分便捷。同时使用 OpenAI 提供的语言模型 API 的另一大优势是选择的多样性，OpenAI 提供了不同的模型以满足语言模型在不同场景的应用，例如 davinci 模型和 gpt-3.5-turbo 等模型。本项目

将集成 davinci 模型和 gpt-3.5-turbo 两种模型分别作为聊天机器人日常对话使用的模型和聊天机器人回答模式的模型。

首先介绍一下 gpt-3.5-turbo 模型，gpt-3.5-turbo 模型是 gpt-3.5 系列最强大的模型也是最经济的模型。gpt-3.5-turbo 也是支持多轮聊天对话模型，因此在调用其时需要将之前的聊天内容一起作为输入，之后 gpt-3.5-turbo 会联系上下文给出最贴切和全面的回答。聊天记录会储存初始的系统信息，用户输入信息以及模型返回的信息三种信息，用户发送第一次消息前，聊天记录中的系统角色会在聊天系统中告诉 gpt-3.5-turbo 模型它的作用，以用来指示模型后续回答的方向和目标，例如告诉它是一个乐于帮助人的机器人。在初始化系统消息后，当用户发送第一次消息后，用户发送的信息会和初始的系统信息一起存入聊天记录中，之后聊天记录会作为 API 的输入，gpt-3.5-turbo 模型会产生基于聊天记录中系统信息和用户输入一起产生输出结果，之后相关输出结果会作为单元模型的输出，同时输出内容也会存入聊天记录中，当用户输入新的消息后会继续储存在聊天记录中，重复执行上述步骤直到聊天记录长度达到模型接受输入的最大值。

```
def response_gpt_text():
    data = request.get_data()
    data = json.loads(data)
    input_message= {"role": "user", "content": data["input"]}
    message_list.append(input_message)
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=message_list
    )
    message_list.append(response.choices[0].message)
    msg=response.choices[0].message.content
```

图 9 GPT3.5 模型样例代码

在 davinci 模型中的储存聊天记录的方式和 gpt-3.5-turbo 大致相同，主要区别在于 gpt-3.5-turbo 将聊天记录储存在数组里，而 davinci 模型将聊天记录以字符串的形式储存下来。除此之外，开发者可以对模型进行配置，例如可以改变 temperature 来控制 davinci 模型的采样温度使得输出更集中确定，还可以通过 max_token 控制聊天记录的长度，通过 presence_penalty 来引导模型向新方向进行交流，通过 frequency_penalty 避免模型产生重复的对话。

```
def response_davinci_text():
    data = request.get_data()
    data = json.loads(data)
    input_message= data["input"]
    global prompt_str
    prompt_str=prompt_str+"你:"+input_message+"\nChatGPT:"
    response = openai.Completion.create(
    model="text-davinci-003",
    prompt=prompt_str,
    temperature=0.5,
    max_tokens=300,
    top_p=1.0,
    frequency_penalty=0.5,
    presence_penalty=0.0,
    stop=["你:"]
    )
    prompt_str=prompt_str+response.choices[0].text+"\n"
    msg=response.choices[0].text
```

图 10　Davinci 模型样例代码

模型单元本设计的核心模块，其能够从前端收到并处理数据。当前端通过 HTTP Post 请求发送消息时，模型单元会立即响应并接收这些请求。模型单元会将请求中的 JSON 数据解析为输入，通过 OpenAI ChatGPT API 进行处理并生成输出。模型单元将输出填充到 HTTP 请求回答中，以便前端能够接收到处理后的数据，并显示出来。

3.2.4 语言单元

语言单元在系统中的配置和其他几个单元关系紧密。其主要作用是允许用户更改语音合成单元、语音识别单元以及模型单元的语言和语音包设置。用户可以使用语言单元轻松切换语音合成单元的语言，以满足和聊天机器人不同语言沟通的需求，确保本项目能合理地满足多语言需求。如下是更改语言设置的样例代码，展示了中文语音包和英文语音包的配置。

```python
@app.route("/getChinese", methods=["POST"])
def getChinese():
    global message_list
    message_list=[
        {"role": "system", "content": "请只说中文"},
    ]
    global prompt_str
    prompt_str=" "
    msg="success"
    return make_response(msg, 200)

@app.route("/getEnglish", methods=["POST"])
def getEnglish():
    global message_list
    message_list=[
        {"role": "system", "content": "Please speak english only"},
    ]
    global prompt_str
    prompt_str=" "
    msg="success"
    return make_response(msg, 200)
```

图 11　语言设置样例代码

3. 2. 5 前端模块

前端模块是整个系统中用户可以交互的部分，采用了 Vue.js、Electron.js 和 ElementPlus 技术来实现聊天界面。在聊天界面中，用户可以与 ChatGPT 进行实时的交互，发送消息并接收 ChatGPT 的回复。通过 Vue.js 的数据绑定机制，聊天界面能够动态地显示用户和 ChatGPT 的聊天记录，每当有新的消息都会及时显示出来，并且会在每个回复的消息旁边标注相应的回复时间。使用 Electron.js 技术能够将聊天界面打包成桌面应用程序，用户可以直接在自己的电脑上使用该应用程序进行聊天，而不需要在浏览器中打开网页。除此之外，前端模块还使用了 ElementPlus 来实现聊天界面的样式和交互效果。Elementplus 是一个基于 Vue.js 的 UI 框架，在聊天界面中，Elementplus 提供了多种 UI 组件，如按钮、对话框、输入框，进度条等，使得用户能够更加方便地进行聊天操作。

```
<template>
    <div>
        <el-row >
            <el-col :span="24">
                <el-scrollbar height="500px">
                    <section class="chatlist">
                        <ul>
                            <div v-for="item in records" :key="item.id">
                                <li class="user" v-if="item.type==1" style="list-style-type:none;">
                                    <el-avatar class="avatar" :src="user"/>
                                    <div class="time"><cite><i>{{item.time}}</i>User</cite></div>
                                    <div class="text" v-html="item.content"></div>
                                </li>
                                <li class="server" v-if="item.type==2" style="list-style-type:none;">
                                        <el-avatar class="avatar"  :src="chatgpt"/>
                                    <div class="time"><cite>ChatGPT<i>{{item.time}}</i></cite></div>
                                    <div class="text" v-html="item.content"></div>
                                </li>
                            </div>
                        </ul>
                    </section>
                </el-scrollbar>
            </el-col>
        </el-row>
        <el-row height="50px">
            <el-col :span="23"  id="input">
                <el-input
                v-model="textarea"
                autosize
                type="textarea"
                placeholder="Please input"
                >
                </el-input>
            </el-col>
            <el-col :span="1">
                <el-button @click="send" :icon="Promotion" type="text" plain circle />
            </el-col>
        </el-row>
    </div>
</template>
```

图 12　聊天界面布局样例代码

4. 系统测试和展示

本章将介绍本项目的系统测试，分别展示中文日常聊天测试，英文日常聊天测试，中文问答测试，英文问答测试四个测试的结果。其中日常聊天测试后的模型是 davinci 模型，而问答测试后的模型是 gpt3.5-turbo。

在中文日常聊天测试环节，我们采用了 davinci 模型进行测试。针对中文日常聊天场景设计了一系列测试用例，包括闲聊、问候、咨询等多种情景，以测试该模型在实际应用中的表现，下面展示了日常打招呼的对话场景。

图 13　中文日常聊天测试

在英文日常聊天测试环节，同样采用了 davinci 模型进行测试。针对英文日常聊天场景也设计了一系列测试用例，包括日常对话的多种情景，以测试该模型在英语语境下的表现，下面展示了英文语境打招呼的场景。

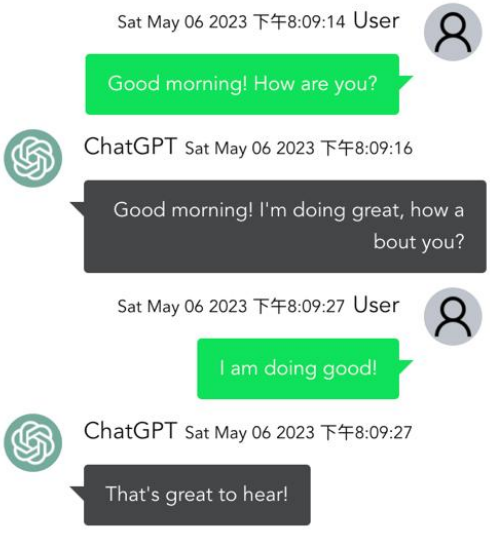

图 14　英文日常聊天测试

在中文问答测试环节，采用了 gpt3.5-turbo 模型进行测试。该模型相较于 davinci 模型具有更强的语言理解和推理能力，更加适合问答模式。下面展示了机器人对我们提出的关于语音聊天机器人的应用前景的回答。

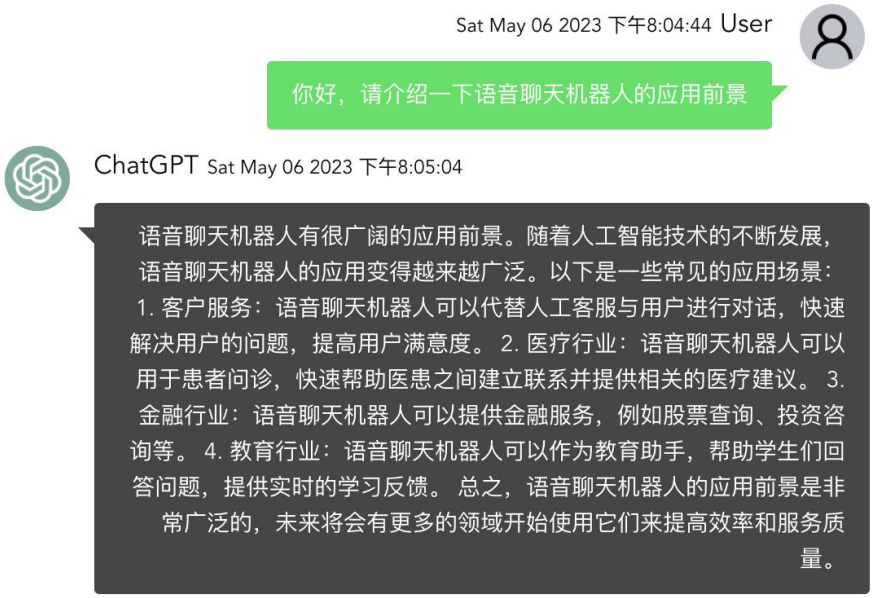

图 15　中文问答测试

在英文问答测试环节，同样是采用了 gpt3.5-turbo 模型进行测试。下面同样展示了机器人对我们提出的关于语音聊天机器人的优点的回答。机器人指出了有提高客户体验，提升了效率，减少了成本，24 小时工作和集成简洁性五个优点。

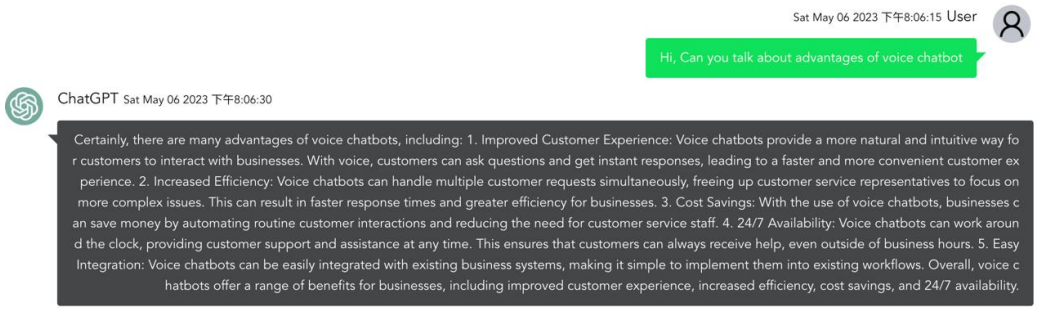

图 14　英文问答测试

总的来说，机器人在四个测试场景测试良好，同时机器人也能在语音聊天环境下顺利使用英文或者中文和用户进行沟通，至此本项目的系统测试通过。

5. 总结和展望

5.1 项目总结

本文介绍了基于 ChatGPT 的语音聊天机器人的实现。本项目采用了后端 Flask，前端 Electron+Vue.js 技术栈，开发了一款跨平台的桌面聊天机器人应用。同时，本项目还集成了 OpenAI ChatGPT API、微软 Azure 语音识别以及语音合成服务，项目支持中英文沟通以及文字和语音和机器人的沟通。

5.2 未来展望

本项目还有一些可以优化的地方。第一点是，本项目还可以增加模型的数量，接入更多的模型，例如 GPT4 模型，GPT4 模型更加强大，能提供更加丰富、准确的回答，满足用户的不同需求。第二点是，可以增加聊天记录离线储存功能，使得用户可以随时查看以前的对话记录，并且在无网络环境下也可以查看聊天记录。第三点是可以考虑拓展除了中英文之外的语言加入系统，以支持更多语言的交流，这样有助于拓展本项目的用户群体。

参考文献

[1] OpenAI. OpenAI API [EB/ OL]. https://platform.openai.com/docs. [May. 07, 2023].

[2] Pallets Project. Flask Documentation. [EB/ OL]. https://flask.palletsprojects.com/en/2.2.x/. [May. 07, 2023]

[3] Microsoft. Text-to-Speech. [EB/ OL]. https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/text-to-speech. [May. 07, 2023].

[4] Microsoft. Speech-to-Text. [EB/ OL]. https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/index-speech-to-text. [May. 07, 2023].

[5] Evan You. Vue.js Guide. [EB/ OL]. https://vuejs.org/guide/introduction.html. [May. 07, 2023].

[6] Element Team. Element Plus. [EB/ OL]. https://element-plus.org/en-US. [May. 07, 2023].

[7] OpenJS Foundation. Electron. [EB/ OL]. https://www.electronjs.org. [May. 07, 2023].

[8] Tan X, Qin T, Soong F, et al. A survey on neural speech synthesis[J]. arXiv preprint arXiv:2106.15561, 2021.

[9] Malodia Suresh,Ferraris Alberto,Sakashita Mototaka,Dhir Amandeep,Gavurova Beata. Can Alexa serve customers better? AI-driven voice assistant service interactions[J]. Journal of Services Marketing,2023,37(1).

[10] Yin Bin,Wu Shu Qi. Enhancing organizational communication via intelligent voice assistant for knowledge workers: The role of perceived supervisor support, psychological capital, and employee wellbeing [J]. Frontiers in Communication,2023.

[11] Vieira Alessandro Diogo,Leite Higor,Volochtchuk Ana Vitória Lachowski. The impact of voice assistant home devices on people with disabilities: A longitudinal study[J]. Technological Forecasting & Social Change,2022,184.

[12] Li Nian,Zhang Bo. The Research on Single Page Application Front-end development Based on Vue[J]. Journal of Physics: Conference Series,2021,1883(1).

[13] Matthew Tyson. Angular, React, Vue: JavaScript frameworks compared[J]. InfoWorld.com,2021.

[14] Paul Krill,Paul Krill. GitHub releases Electron 1.0 for desktop app developers[J]. InfoWorld.com,2016.

[15] Adamopoulou E, Moussiades L. An overview of chatbot technology[C]//Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16. Springer International Publishing, 2020: 373-383.

[16] Dahiya M. A tool of conversation: Chatbot[J]. International Journal of Computer Sciences and Engineering, 2017, 5(5): 158-161.

[17] Shafeeg A, Shazhaev I, Mihaylov D, et al. Voice Assistant Integrated with Chat GPT[J]. Indonesian Journal of Computer Science, 2023, 12(1).

[18] Cahn J. CHATBOT: Architecture, design, & development[J]. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science, 2017.

[19] Shawar B A, Atwell E. Different measurement metrics to evaluate a chatbot system[C]//Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies. 2007: 89-96.

[20] Nagarhalli T P, Vaze V, Rana N K. A review of current trends in the development of chatbot systems[C]//2020 6th International conference on advanced computing and communication systems (ICACCS). IEEE, 2020: 706-710.

致谢

大学四年如梦似幻，但在这段时间里，我汲取了许多知识和感悟，结交了许多优秀的师长和朋友。因此，我由衷地感谢母校为我四年的培养和指导，也感谢那些一路相伴并在成长道路上给予我帮助和启示的人们。首先，在论文即将完成之际，回顾紧张却又充实的学习和开发过程，我要特别感谢刘江老师和章晓庆师兄，他们提供了宝贵的建议和指导，让我更加深入地了解了研究领域的前沿和趋势，顺利完成毕业设计。