上海大学

SHANGHAI UNIVERSITY

2022-23 学年夏季学期专业实习

个人研究（开发）报告

题目：基于开源 ASR、TTS 和大语言模型的中文智能语音助手开发

—— 以盾构机隧道施工场景为例

学生姓名：王欣洲 学号：20124444

指导老师：胡珉

日期：2023 年 8 月 25 日

目录

1. 研究背景

在现代社会中，智能助手作为人工智能技术的一种应用，正在逐渐渗透到各个领域，为人们提供便捷的服务和信息。随着语音识别技术和自然语言处理的不断进步，智能语音助手（AI Voice Assistant）在交流、信息检索和任务执行方面的作用越来越显著。

本文在为盾构机隧道施工工程现场，通过智能语音助手提供帮助信息的实用语境下，旨在通过描述、评价和测试开发此类中文语音助手各功能模块的实际过程，记录和评估实际开发的可行性，并完成相应 demo 的开发和部署。

1.1 已有和现实研究（开发）背景

在智能语音助手的商业化领域，德勤咨询《2021 年中国智能语音市场分析报告》[1]指出，随着智能语音驱动的智能驾驶，金融、运营商、教育、医疗等领域普及，专业级应用增强，其市场占有率和前景日益增长。在未来交互、技术输出、资讯同步、搜索革新领域，中文智能语音助手都有着较大的收入增长潜力。

在学术领域，也有许多文章从该类软件的使用价值和开发的角度系统阐述了语音助手的技术实现，相关的研究包括 Subhash 等 2020 年提出[2]，语音控制模块，即 ASR 和 TTS 部分，是语音助手的“重要增长特性”，这也是其改变人们生活方式的直接体现。它还从开发角度提出通过 GTTS 转为英文音频并用 python 的 playsound 播放的输出方案，也被本例采纳。详细描述此类助手开发的还有 Kim 等 2020 年发布的侧重于 Voice Control System 的研究[3]，其采用第三方语言模型的设计思路（其采用 Google 助手 API 作为智能算法部分），被本例采纳。从实用角度，Faruk[4]等也在 2022 提出，当下对于语音助手的可用性研究并不多见，并设计了采用 ISO 9241-11 框架作为测量工具的可用性度量系统，包括一系列多样的独立变量，用于衡量可用性。这套可用性度量方案的诸多变量被本例采纳。

1.2 研究（开发）意义和目标

隧道施工作为复杂而危险的工程领域，需要及时准确的信息交流和问题解决。而智能助手的开发为隧道施工工作提供了更高效、更智能的解决方案。中文语音识别技术能够使施工人员不再受限于键盘输入，而能够通过语音快速发起查询和指令。同时，基于大语言模型的智能回答能力，使得语音助手能够更自然、更贴近人类交流的方式与用户互动。

---

[1] 德勤 Deloitte《未来的语音世界：中国智能语音市场分析》，2021 年 12 月，未来的语音世界——中国智能语音市场分析 | 德勤中国 | 科技、传媒和电信行业 (deloitte.com)

[2] S Subhash, Prajwal N Srivatsa, S Siddesh, A Ullas, B Santhosh. "Artificial Intelligence-based Voice Assistant." IEEE. 29 Cites in Papers, 2912 Full Text Views, Artificial Intelligence-based Voice Assistant | IEEE Conference Publication | IEEE Xplore

[3] Tae-Kook Kim. "Short Research on Voice Control System Based on Artificial Intelligence Assistant." IEEE. Published in 2020 International Conference on Electronics, Information, and Communication (ICEIC). Cites: 12, Full Text Views: 765, Short Research on Voice Control System Based on Artificial Intelligence Assistant | IEEE Conference Publication | IEEE Xplore

[4] Faruk Lawal Ibrahim Dutsinma, Debajyoti Pal, Suree Funilkul, Jonathan H. Chan. "A Systematic Review of Voice Assistant Usability: An ISO 9241–11 Approach." SN Computer Science, volume 3, Article number: 267 (2022). Published on 03 May 2022, A Systematic Review of Voice Assistant Usability: An ISO 9241–11 Approach | SpringerLink

本项目的目标是完成一个针对隧道施工领域的中文语音助手的开发，它能够实现语音识别、自然语言理解和语音回答的功能。该助手能够在嘈杂环境下准确识别用户的语音指令，并根据问题提供合理、基本准确、比较人格化和自然的回答。

1.3 程序设计概述

智能语音助手包含五个功能模块[5]，涉及机器学习，网络响应，计算机听觉等领域。五个模块分别由 audio_recording.py, recognize.py, chat_response.py, tts_executor.py, playsound.py 实现封装，并由 launch.py 执行。由 env_output.py 记录环境和日志信息，asr_noise_accuracy.py 进行各 ASR 模型的抗噪能力测试评估。程序流程如下：

2. 语音助手 ASR 和 TTS 模块的测试和技术实现

语音识别与文本转语音，即自动语音识别（Automatic Sound Recognition）和自动文本转语音（Text To Speech），是智能助手开发的关键技术。其中，ASR 利用自动方法识别语音内容，将现场用户的声音转化为文本进行输入，TTS 则将语音助手软件经过算法得出的回答文本转换为声音，作为输出传递给用户。这些技术通常基于机器学习来实现，如后文中 MASR 所使用的门控卷积神经网络，抑或是 ASRT 使用的梯度决策树。

从软件开发的角度，本研究将通过分析当前成熟的开源中文 ASR 和 TTS 工程项目，通过系列测试，评估其性能、易用性、工程性，得出最适合所述软件的开源项目作为其实现的核心技术。该工程应兼具易用、适合在服务器上部署、严格开源等特点。

2.1 中文 ASR 和 TTS 的开源工程部署、使用及其特点的初步分析

本研究汇总了六个开源中文语音识别和文本转语音项目[6]，部分支持双向技术，部分仅支持其中之一。项目主页和文档链接已收录于报告。研究认为环境配置成功后，各模型效果良好，但其易用性和二次加工的难易度（即工程性）仍需加以关注和分析。其中四个可用性较高，成为后期开发的备选，它们是 PaddleSpeech, MASR, ASRT, ESPnet。

根据其开源文档等信息，本研究在 Linux Ubuntu 或 Windows 系统上，分别实机部署并使用这些模型。在部署过程中，本研究将初步考察下述六类开源中文语音工程的可用性。

2.11 PaddleSpeech – 支持 ASR 和 TTS

PaddleSpeech 是基于飞桨 PaddlePaddle 的语音方向的开源模型库，用于语音和音频中的各种关键任务的开发，包含大量基于深度学习前沿和有影响力的模型。PaddleSpeech

---

[5] 各模块封装情况详见 readme.txt

[6] 它们分别是 PaddleSpeech, MASR, ASRT, ESPnet, WeNet 和 DeepSpeech

符合盾构智能助手开发项目所要求的开源特点，并兼备 ASR 和 TTS 两项功能。除此之外，PaddleSpeech 拥有完善且清晰的中文文档，得力于此，部署和使用过程较为顺利，这一过程也体现了其易用性。

★ **PaddleSpeech 部署过程、问题和相关文档：**

官方文档：PaddleSpeech/README_cn.md at develop · PaddlePaddle/PaddleSpeech · GitHub

官方硬件&环境要求：在 Linux 环境下，3.7 以上版本 python

测试环境：Linux Ubuntu 18.04 on VMware Virtual Machine - python3.10.6

```
/home/chriswang/ASR/PaddleSpeech_0619/venv/bin/python
/home/chriswang/ASR/PaddleSpeech_0619/env_output.py
architecture: ('64bit', 'ELF')
machine: x86_64
node: chriswang-virtual-machine
platform: Linux-5.15.0-56-generic-x86_64-with-glibc2.35
processor: x86_64
python_build: ('main', 'Nov 14 2022 16:10:14')
python_compiler: GCC 11.3.0
python_version: 3.10.6
release: 5.15.0-56-generic
system: Linux
version: #62-Ubuntu SMP Tue Nov 22 19:54:14 UTC 2022
```

环境配置过程：按照官方文档配置 paddlepaddle 环境，安装相关依赖。

配置过程中遇到的问题（仅含与 PaddleSpeech 工程本身相关的问题）：

- numpy 版本问题：在本机运行语音识别函数过程中，报“np.complex depreciation”错。可能的原因是，由于 PaddleSpeech 使用了 numpy 依赖，又由于本机 Ubuntu 系统以前配置过全局的 numpy 环境，而该版本的 numpy 与 PaddleSpeech 工程中使用的 numpy 版本不兼容，而配置过程中默认没有在虚拟环境中安装 numpy 的兼容版本。
- 解决：在虚拟环境下安装兼容版本的 numpy。

```
(venv) chriswang@chriswang-virtual-machine:~/ASR/PaddleSpeech_0619$ pip install numpy==1.21.6
```

★ **PaddleSpeech 部署&使用的特点分析：**

不论是在 ASR 还是在 TTS 过程中，注意到首次运行均会进入 Setuptools Replacement 过程（该过程是在运行__init__.py 进行工程初始化）。这个过程在本机测试时耗费约 20~30 秒，这个过程用来预训练模型生成缓存。

就准确度而言，PaddleSpeech 表现比较优秀。

a. ASR - 调用依赖函数即可，除首次外响应速度均为即时。

```
from paddlespeech.cli.asr.infer import ASRExecutor
asr = ASRExecutor()
result = asr(audio_file="zh.wav")
```

b. TTS - 调用依赖函数即可，支持高级声音混合（男女声音均支持），自适应标点符号停顿，准确度很高。

```
from paddlespeech.cli.tts.infer import TTSExecutor
tts = TTSExecutor()
tts(text="content", output="output.wav")
```

PaddleSpeech 经过测试后，得出的最大缺陷依然是首次运行的配置过程问题。尚且没有找到如何越过首次使用的__init___.py 的预加载的方法。对比后文提到的其他语音处理工程，PaddleSpeech 是唯一需要预载的。另外，虽然 PaddleSpeech 对开发者使用很友好，但不支持语言模型的替换[7]，即无法轻易调整参数。

2.12 MASR – 仅支持 ASR

MASR 是一款基于 Pytorch 实现的自动语音识别框架，MASR 全称是神奇的自动语音识别框架（Magical Automatic Speech Recognition），MASR 致力于简单，实用的语音识别项目。可部署在服务器，Nvidia Jetson 设备，未来还计划支持 Android 等移动设备。MASR 使用的是门控卷积神经网络（Gated Convolutional Network），网络结构类似于 Facebook 在 2016 年提出的 Wav2letter，只使用卷积神经网络（CNN）实现的语音识别。

MASR 的特点是拥有非常健全且活跃的开源社区。MASR 的 pytorch 实现是该社区众多开发者共同开发的结果，使用的技术单一且简单，并支持自定义语言模型。

★ MASR 部署过程、问题和相关文档：

官方文档：masr: 中文语音识别，提供预训练模型，高识别率 Chinese Speech Recognition; Mandarin Automatic Speech Recognition; (gitee.com)

MASR 语音识别算法原理解析（非官方）：(73 条消息) MASR 语音识别算法简介_HELLOWORLD2424 的博客-CSDN 博客

MASR 对比其他开源 ASR 项目[8]：docs/compare.md · wangnanbo/masr - Gitee.com

官方硬件&环境要求：Linux&Windows （官方文档中未提及 python 版本要求）

测试环境[9]：Linux Ubuntu 18.04 on VMware Virtual Machine - python3.6

---

[7] 因为 PaddleSpeech 已经是建立在 PaddlePaddle 团队基于 Python 机器学习库 PaddlePaddle 开发的二次工程，因此仅能适配 PaddlePaddle 自带的语音语料库模型

[8] MASR 中文社区的创作者自行总结的分析比对，主要是与 ASRT 进行比对，也提到 DeepSpeech

[9] MASR 的测试环境除 Python 版本外与 PaddleSpeech 一致

环境配置过程：按照官方文档以及工程文件内置 readme.md 说明文件进行环境配置。由于工程使用了 pytorch 依赖，使得最新版本的 python 与之不兼容；同时，该模型支持且仅支持自定义的模型训练，即无预训练的语言模型，需要开发人员自行选择语言模型训练，这都导致了更高的环境配置的复杂程度。对此，作者也提供了 docker 版本（封装环境版）。然而出于熟悉工程的目的，还是尝试为其重新配置了环境，并基本能够成功运行。

配置过程中遇到的问题（仅含与 MASR 工程本身相关的问题）：

- python 版本问题：MASR 与常用 python 版本不兼容。由于官方文档中并未提及明确的 python 版本要求，在使用 python3.10 时，出现无法安装 torch 依赖的错误。这是由于本文测试的 MASR 工程版本是基于 MASR 原理进行重新设计的 PyTorch 版，而原版中没有使用 torch 依赖（原作者已停止更新），因此文档中并未指出 python 版本过高会出现问题（pytorch 目前不支持更新于 python3.7.9 的发行版本，关于 pytorch 兼容版本的信息：https://stackoverflow.com/a/58902298/5090928）。
- 解决：通过虚拟环境运行 python3.6，并安装相应的兼容版本的 pytorch

★ **MASR 部署&使用的特点分析：**

使用方法：调用卷积函数训练和预测即可。

```
import _init_path
from models.conv import GatedConv
model = GatedConv.load("pretrained/gated-conv.pth")
text = model.predict("test.wav")
```

控制台在虚拟环境下用 python 执行 demo。

```
(venv) chriswang@chriswang-virtual-machine:~/ASR/MASR_0619$ python3
examples/demo-recognize.py
```

MASR 的部署文档完善，部署阻碍较小。此外，在可二次开发性方面，官方提供了 6 个预设的中文语言模型，可以替换使用，亦可以替换自定的训练集。该工程不需要首次运行预载。MASR 也同时支持即时录音响应。并提供完善的服务器部署文档。

2.13 ASRT – 仅支持 ASR

ASRT 项目是一个基于深度学习的中文语音识别系统，采用基于深度卷积神经网络和 CTC 的 TensorFlow.Keras 来构建语音模型。值得一提的是，此系统还自带一个基于 HTTP 的服务器软件，可以较为便利地搭建 API 服务器，以便其他客户端发送 API 请求。

★ **ASRT 部署过程、问题和相关文档：**

GitHub 项目地址：https://github.com/nl8590687/ASRT_SpeechRecognition

基于 ASRT 的语音识别客户端应用：

- Windows 桌面版：https://github.com/nl8590687/ASRT_SpeechClient_WPF
- Windows 10 UWP 版：https://github.com/nl8590687/ASRT_SpeechClient_UWP

ASRT 工程官网（作者个人维护）：

ASRT: Auto Speech Recognition Tool | AI 柠檬 (ailemon.net)

作者个人知乎专栏对 ASRT 的介绍：

ASRT：一个中文语音识别系统 - 知乎 (zhihu.com)

官方硬件&环境要求：Linux: Ubuntu 18.04 + / CentOS 7 + 或 Windows 10/11

Python: 3.7 - 3.10 及后续版本 TensorFlow: 2.5 - 2.11 及后续版本

测试环境：Linux Ubuntu 18.04 on VMware Virtual Machine - python3.10

环境配置过程：按照 Github 主页文档进行配置。

配置过程中遇到的问题：

- Flask 架构&网络&文档问题：该工程又采用客户端+服务端的 Flask 架构，在配置环境过程中出现了许多问题。另外，在客户端使用的过程中，需要联机调用 API，并且许多依赖（如 tensorflow 和训练数据集）文件很大，经历多次断线。工程文档方面，诸多作者提供的链接所指向的网站无人维护（Forbidden 403），造成障碍。
- 解决：未能彻底解决，但通过使用代理、使用作者提供的网盘镜像，以及参考 csdn 开发者文章获取需要的依赖信息，成功了运行了其中的 demo。
- 报 tensorflow deprecated 错误：据悉是 tensorflow 版本问题，未能正确安装 tensorflow 版本。目前还没有可行的解决方案。

decay is deprecated in the new Keras optimizer, please check the docstring for valid arguments, or use the legacy optimizer, e.g., tf.keras.optimizers.legacy.Adam.

2.14 ESPnet – 支持 ASR 和 TTS

ESPnet 是一个端到端语音处理工具包，涵盖端到端语音识别、文本转语音、语音翻译、语音增强、说话人分割、口语语言理解等领域。ESPnet 使用 PyTorch 作为深度学习引擎，同时遵循 Kaldi 风格的数据处理、特征提取的格式。

★ **ESPnet 部署过程、问题和相关文档：**

官方文档：Usage — ESPnet 202304 documentation

Github 项目地址：GitHub - espnet/espnet: End-to-End Speech Processing Toolkit

官方硬件&环境要求：Linux: Ubuntu / CentOS 7 / debian 11 & Python: 3.7 - 3.10 及后续版本 & 无 Windows 支持

测试环境：Linux Ubuntu 18.04 on VMware Virtual Machine - python3.10

环境配置过程：按照 Github 主页文档及官方文档进行配置。需要注意的是直接调用函数会自动下载训练集，使用 python download 命令下载很慢。需要自行寻找可用的中文语音训练集，并将其放在该路径下：

…\site-packages\espnet_model_zoo\a1dd2b872b48358daa6e136d4a5ab08b

★ **ESPnet 部署&使用的特点分析：**

经过初步判断，ESPnet 在全部所述工程中，其自带的功能接口是最全面的。在部署过程中，亦未遭遇较大阻碍。尽管有待后续进一步测试，直观上，ESPnet 的 ASR 准确度很高，且速度很快，可以达到即时转换。其另一个显著优点是，其提供的语言模型以及工程文件是所有参与比对的中文语音识别项目中最小的，这有利于在服务器上进行部署。在配置环境的过程中亦未出现大的障碍，易用性较好。

a. ASR - 通过调用 soundfile 库和预设函数即可实现

```
audio, rate = soundfile.read("zh.wav")
nbests = speech2text(audio)
text, *_ = nbests[0]
```

b. TTS - 官方文档位置：

CMU 11492/11692 Spring 2023: Text to Speech — ESPnet 202304 documentation

虽然 ESPnet 支持 Text to Sound，但是其中文标准女声库调用的标贝（data-baker）的【中文标准女声音库】，虽然该语料库号称开源，但却不支持商用，见：

标贝开源语料库网站：https://www.data-baker.com/open_source.html

基于这一原因，本例不再考虑其 TTS 功能作为后续开发的备选。

2.15 WeNet & DeepSpeech – ASR

环境配置问题：均遇到系统不支持报错，猜测 WeNet（在 Windows 上和 Linux Ubuntu 上均进行了测试）和 DeepSpeech（在 Linux Ubuntu 上进行了测试）的 python 发行库不支持 Ubuntu 20.04 系统，也不支持 Windows11 系统，也可能有其他原因，目前尚未得到解决。

2.16 其他[10]ASR 和 TTS 服务

阿里云 ASR 和腾讯云 ASR：腾讯云提供”免费+付费“的中文自动语音识别服务：功能体验 - 语音识别 - 控制台 (tencent.com) 腾讯云经过测试，有些卡顿，基本准确；特点是可以选择音频类别，即电话和非电话。另一个优势是支持包括粤语、上海话在内的方言识别。

Microsoft Azure (TTS)：提供免费的 API 接口，缺点是现阶段需要注册账号，申请资格，支付验证费用：微软 Microsoft Azure——文本转语音(TTS) REST API 使用教程 - 知乎 (zhihu.com) & 文本转语音 – 真实 AI 语音生成器 | Microsoft Azure

2.2 可用开源 ASR 工程的速度、准确度（抗噪性能）测试

对 2.1 中可用的工程进行速度、准确度、抗噪能力的综合测试。该测试的设计是，基于开源中文音频语料数据库（即包含音频文件和其对应的正确内容文本的数据库），进行结果比对测试。准确度方面，测试设计的指标主要是 CER（Character Error Rate）、WER

[10] 非开源或无法进行本地部署的

（Word Error Rate）、MER（Missed Error Rate）和 MIL（Message-Intent-List Error），详见 2.23 测试设计 – 原理&指标。速度方面，使用平均处理时间，字均处理时间。本研究选取了涵盖噪声的数据集进行测试，一定程度上能够检验其抗噪性能。

2.21 开源语料库

几个最新免费开源的中文语音数据集：https://blog.ailemon.net/2018/11/21/free-open-source-chinese-speech-datasets/

Aishell 语料库 OpenSLR：http://www.openslr.org/33/

Aishell 是由北京普智声科技有限公司发布的开源汉语普通话语音语料库。该语料库邀请了来自中国不同口音地区的 400 名人参与录制，通过专业的语音标注和严格的质量检查，手工转录的准确率达到 95%以上。这些数据可以免费用于学术研究。

该中文语料库的特点是涵盖了多个场景，例如商业办公、工业生产等。其优势是，它给出的文字对应文件的准确率较高。另外，其语料范围涵盖含有杂音的音频，能够一定程度上模拟盾构机隧道作业的噪声环境。缺点是它对于噪声的等级等参数缺少量化，在某种程度上降低了本研究对抗噪性能测试方面结论的可信程度。

2.22 音频对应文字文本处理

对音频对应的文本文件进行 NLP，包括进行正则化，即使用 re 删除脚本文件中的停顿空格等操作，具体流程详见附录：

…[11]

```
match = re.search(r'(BAC\S+)\s+([\u4e00-\u9fa5])', line)
```

这一过程的目的是将该音频数据库对应的文本文件做预处理。包括简单的格式调整和数据降噪，以便后续比对测试。

2.23 测试设计 - 原理&指标

ASR 效果评测原理与实践：AI 科普文章 | 语音识别准不准

ASR（Automatic Speech Recognition）语音识别测试测试流程：ASR（Automatic Speech Recognition）语音识别测试测试流程 - 北向。 - 博客园 (cnblogs.com)

测试工具库：Python - 语音识别文本相似性度量库 jiwer，可计算文字错误率 WER、匹配错误率 MER 等相似性度量指标 - StubbornHuang Blog

WER（Word Error Rate）：WER 是词错误率的缩写，是用于评价 ASR 效果的重要指标，它用来衡量预测文本与标注文本之间的错误率，精确的说，是用于衡量识别结果中词级别的错误率。它计算被错误识别的单词数量与总单词数量之间的比例。较低的 WER 值表示系统在词级别上的识别准确性更高。

[11] 代码流程详见附录 nlp_transcript.py

$$WER = \frac{\#Deletions + \#Insertions + \#Substitutions}{\#ReferenceWords}$$

（Deletions：删除错误字符数，Insertions：插入错误字符数，Substitutions：替换错误字符数，ReferenceWords：总字符数）

需要特别说明的是，WER（Word Error Rate），即词错误率，往往用于测试英文文本[12]。这是因为字母体系语言语句中最小单位是词（Word），而中文最小单位是汉字（Character），因此在中文语音识别任务中，使用字错率（Character Error Rate，CER）来衡量 ASR 识别效果。两者的计算方式相同，变量单位不同。在中文领域，有时也会使用 WER 表示该指标，但实质上是 CER。

MER（Missed Error Rate）：MER 用于测量系统未能正确识别的语音片段所占的比例，即漏报错误。较低的 MER 值表示系统更准确地识别语音片段。

WIL（Word Information Loss）：用于评估系统在转换语音输入为文本输出时所丢失的信息量。它衡量了识别过程中对原始语音中所包含信息的损失程度。WIL 越低，表示识别结果与原始语音输入更接近，信息丢失较少，识别效果较好；而 WIL 较高则表示信息丢失较多，识别效果不佳。

MIL（Message-Intent-List Error）：MIL 是消息-意图-列表错误的缩写，用于评估系统在语音助手等应用中正确地识别用户的意图和生成正确的回应。它涉及对消息（用户输入）、意图（用户意图）和列表（可能回应列表）之间的匹配准确性。尽管在本报告范围内，无法对该指标做出测试，但 MIL 很可能为后续研究提供一个有效的指标。

2.24 测试中遇到的问题

- PaddleSpeech：内存溢出错误。这个错误的问题在于 PaddleSpeech 在训练时如果内存不够，会直接做 kill 进程处理，即这个错误无直接报错也无错误日志可查的。
- 解决：在硬件层面添加内存。由于 PaddleSpeech 工程本身的代码设计，它的训练对于硬件要求很高。在为虚拟机分配 4G 内存时，PaddleSpeech 只成功识别了语料库的前 5 条；8G 内存时识别了 21 条；16G 内存时识别了约前 100 条。目前还没有完全从软件层面解决这个问题的方法。

2.25 测试程序设计简述[13]

使用 Python 第三方语音识别测试库 jiwer 进行对各 ASR 模型的测试。该库内置了计算上述个指标的函数。

以下是部分代码和注释[14]：

---

[12] 包括英文文本在内的最小表意单位为单词（word）的语言体系。这一体系的语言被称为“形态素语言”（Agglutinative Language），通常通过将词缀添加到词干上来表示不同的语法和语义关系。适合使用 WER 进行对应的准确度评价；相应的，以字（character）为最小单位的语言被称为“音素语言”（Alphabetic Language）。这类语言使用字母（音素）作为构建单词的基本单位，每个字母代表一个特定的语音音素，比如中文。适合使用 CER 进行准确度评价

[13] 部分对照测试结果样例见附录 espnet, masr, paddlespeech_noise_test.csv

[14] 测试代码见附录 asr_noise_accuracy_test.py

```
…
# 遍历每个语音样本进行测试
for audio_path, reference in zip(test_audio, reference_text):
    counter += 1
    start_time = time.time()  # 记录开始时间
    transcription = recognize_speech(audio_path)  # 对语音进行识别
    processing_time = time.time() - start_time  # 计算处理时间

    try:
        error_rate = cer(reference, transcription)  # 计算词错误率（Word Error Rate）
    except ValueError:
        error_rate = 0

    tpc = processing_time / len(transcription)  # 计算平均每字符处理时间
    …
```

通过 jiwer 库，可以较为方便地进行准确度测试。将经由模型得出的识别文本（Transcription），和给定的默认准确[15]的对应文本（Reference）进行比对。

2.26 初步的测试结果 - 366 条语音数据

限于研究体量和硬件，本项目为 MASR, ESPnet, PaddleSpeech 三个工程的 ASR 进行了测试，测试集为 366 条。并得出了如下结果：

模型	CER	MER	WIL	处理时间	字均处理时间
MASR	0.1071131136	0.611	0.61	0.311923899	0.026327601
ESP_Net	≈ 0.004	0.003	0	5.517344459	0.440989218
PaddleSpeech	0.023133293	0.222	0.22	6.347445911	0.618048923

（时间单位：秒）

2.3 综合分析和结论

从 2.1 的初步部署结果中可以得出，在六个中文开源 ASR 和 TTS 工程中，WeNet 和 DeepSpeech 在部署方面，未能成功在 Linux 系统上运行，也就意味着服务器部署可行性较差，即便采用，也可能对后期维护造成负担。ASRT 工程文件较为臃肿，不适合部署在服务器上，且有同样的易用性问题。MASR 虽然处理速度极快，但处理准确度上相较 ESPNet 和 PaddleSpeech 有明显的劣势，CER 达到了约 0.1。ESPNet 速度适中，准确度极高，CER

[15] 实际上是约 95%的准确度

达到了 0.004，但缺陷是它并不兼具 ASR 和 TTS 两端的功能[16]。这意味着，如果单独采用其 ASR 功能，会增加将一个 TTS 工程整合进最终的成品软件的工作，这往往意味着环境冲突和更臃肿的工程大小。PaddleSpeech 在兼具准确度、速度和工程完整性的同时，保证了开源属性。

因此根据目前的信息，得出采用 PaddleSpeech 作为实现智能语音助手的 ASR 和 TTS 模块的核心技术的结论[17]。

3. 语音助手智能算法（AI）模块的技术实现

语音助手的智能算法将用户的输入转换为输出。这个过程的实现取决于具体场景。就本报告所设计的盾构机隧道施工场景来说，本课题设想两种实现途径。

其一是通过人工设计的文本决策数据进行回答的生成。其数据结构可以是决策树或者二维表。就决策树而言，具体来说，算法提取用户输入文本的关键词，并生成预备好的相应的回答的集合。其特点是准确度高，可能适用于回答需要极为严谨的智能软件。这一种实现的主要前提是对现场的问题和相应回答的文本的健全数据库，在这个前提下，技术上的障碍较小，本课题从技术角度出发，并不深入探究。

另一种是基于目前流行的大语言模型，即将现场的问题和回答的数据交由语言模型进行训练。在这个过程中，大语言模型充当分类器，同时，它也具备至少两点优势：

a. 对回答文本语言在一定程度上的“人格化”，“自然化”的功能：大语言模型在“润色”措辞这一方面的能力非常优秀。仅仅基于预备的数据库中的回答进行简单集合，所产生的文本输出相较而言比较“机械化”，在表达上，可读性和亲和力都不强。

b. 对输入文本的纠错：以 OpenAI 旗下的 ChatGPT 为例，即便输入的中文语法不通畅、有错字、漏字，在很大程度上不影响 ChatGPT 的正确返回。这恰好能弥补在 ASR 环节可能出现的因噪声或多义词、多音字等因素导致的文本识别错误。

3.1 ChatGPT API 的调用、主要问题和解决

在语音助手中使用大语言模型可以通过调用 ChatGPT API 来实现。API（Application Programming Interface）作为允许不同应用程序之间相互通信的接口，借由本地服务器向 OpenAI 服务器发送请求并获得响应的方式，实现对话。

…[18]

```
response = openai.Completion.create(
    engine="text-davinci-003",  # 选择模型引擎
    prompt=user_input,
    max_tokens=50  # 控制生成回答的长度
)
```

---

[16] ESPNet 的 TTS 功能并非严格意义上的开源，因为它调用了一个非开源中文语音语料库进行训练

[17] 需要特别说明的是，选择 PaddleSpeech 着重考虑了本项目的体量因素，因此强调了易用性在决策中的权重。然而其他同类程序的开发，应当综合考虑速度、准确度、抗噪性能、开源属性等多种因素。

[18] 代码详见附录 chat_response.py

```
# 提取生成的回答
generated_answer = response.choices[0].text.strip()
```

显而易见的，这一实现有两个主要问题：

a. 网络问题：OpenAI 的 API 并不支持来自中国大陆地区的访问。其解决途径有三种：

- 将 API 模块部署在位于境外的服务器：通过将这部分代码部署在亚洲地区的境外服务器，可以兼备低延迟和绕过来自境内 IP 访问被拒绝的问题。经测试，通过基于腾讯云的部署于东京的服务器进行访问，响应速度非常优秀。而通过使用 nc 或 ssh 技术连接位于境内的服务器和位于东京的服务器，传输速度也是可预料的在接受范围内。也就是说，经过测试，这是一个可行的解决。
- 使用代理技术：主流的服务器开源代理技术有 squid 等。本报告不深入探究。在最终的 demo 中，采取的是这一种途径。
- 将类似的大语言模型部署在本地：在 Github 等开源平台上，可以找到将 ChatGPT 等大语言模型部署在本地的工程。但这样做有两个显著的问题，一是尚不能明确这是否符合大语言模型开发和供应商的规则，二是这类工程非常庞大，对服务器算力要求很高，在部署和运行层面都会遇到困难。

b. 响应速度和精确度的不可控性：

由于使用了第三方非开源技术，是否能够成功获得返回还体现在 OpenAI 服务器是否正常提供服务等不可控因素。另外，该模型的返回可能出现不合规、不合适、不准确的内容。在某种程度上，可以通过告知用户内容由第三方人工智能算法参与生成来规避问题，也能使得用户不过分依赖软件给出的回答。

3.2 ChatGPT 基于多轮对话实现自定义训练集

在实际应用中，用户与语音助手之间往往是多轮的对话。为了让语音助手在这种情况下表现更好，本例通过构建自定义训练集来进行模型微调。这意味着，前述中的一系列与隧道施工相关的对话，包括用户的提问和系统的回答仍然需要——但它不必是严格的决策树结构，即包含严格对应的问题文本和回答文本。它只需要是陈述性的知识内容即可。这是因为，通过多轮对话的训练，模型可以学会处理上下文信息，应对复杂的用户查询。

```
…[19]
pretrained_data = "回答围绕盾构机相关信息，不超过 30 个汉字，不多于 2 句话"
messages = [{"role": "user", "content": ""}]
str_in = "【用户输入的语音，如：如何处理黄色信号灯闪烁的情况？】"

for text in [pretrained_data, str_in]:
    messages.append({"role": "user", "content": text})
    generated_answer = askChatGPT(messages)
```

需要指出的是，上述 pretrained_data 中的内容仅仅是一个简单的例子。在实践中，这

[19] 代码详见附录 chat_response.py

里将被写入的内容是大量的预备数据，即隧道施工相关的对话，包括用户的提问和系统的回答。

4. ASR&TTS 和智能算法的整合和 demo 上线

4.1 demo 环境和各模块整合

将各个模块统一环境，兼顾封装性、运行效率、可部署等特性完成 demo 的开发。这一 demo 使用 PaddleSpeech 作为 ASR 和 TTS 模块的核心，调用 ChatGPT API 从 OpenAI 服务器获取结果响应。这个 demo 可以在设置了代理的本地 Linux 系统上运行。由于完全使用了跨平台技术，也应当可以在其他操作系统上运行。

这一版本的 demo 的问题在于，直观上说，在响应速度和用户交互方面都不够流畅。另外，它仅能通过本地通过境外服务器转发代理实现运行。为了解决这个问题，本项目还设计了另一个基于 Http 的，将 ChatGPT 功能模块部署在境外服务器上的联网版本（见 4.2 上线部分），它可以在任何能获取 Http 响应的设备上运行。

4.11 录制用户语音生成.wav 文件

使用 pyaudio 和 wave 库即可实现：

…[20]

```
stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)  # 创建录音文件
```

可调参数包括录音时间，格式，声音频率和大小等。

```
audio_recording.start_audio(5, "zh.wav")
```

4.12 使用选定的 ASR 模型进行中文语音识别

在本例中，使用 PaddleSpeech 模型进行 ASR 识别：

…[21]

```
asr = ASRExecutor()
 return asr(audio_file=audio)
```

在 launch.py 中调用 recognize.py 中的 recognize 函数。

```
str_in = recognize.recognize("zh.wav")
```

4.13 将识别结果发送至 ChatGPT 大语言模型获取响应

---

[20] 代码详见附录 audio_recording.py

[21] 代码详见 recognize.py

详见 3.2 多轮对话实现预训练数据集。

response = chat_response.multi_response(str_in)

4.14 使用选定的 TTS 模型进行文本转语音

本例使用 PaddleSpeech 默认女声：详见 2.11.b

tts_executor.tts_execute(response)

并基于 playsound 库中的 playsound 函数进行播放。

playsound("output.wav")

4.2 基于 PaddleSpeech 和 ChatGPT 的中文语音助手 demo 上线

在本例的软件上线操作中，主要使用了 Django 和 Nginx 架构[22]。本例并没有将整个工程上线至服务器，而是设计了将涉及 ChatGPT 的智能算法逻辑模块上线至服务器，将 PaddleSpeech 的 ASR 和 TTS 模块部署在本地。其原因一方面受限于项目体量，另一方面，基于机器学习的 ASR 技术超过了可用服务器的计算负荷，分离部署也更体现工程性。由于选题限制，该部分不深入讨论，下面是部分代码[23]：

```
def send_request_to_django():
    # Django 服务器的 URL
    django_url = "http://124.222.120.214/ai_voice_assistant_handle/hi"


    try:
        # 发送 GET 请求到 Django 服务器
        response = requests.get(django_url)

        # 检查响应状态码
        print(response.status_code)
        if response.status_code == 200:
            # 获取返回的文本数据
            response_text = response.text
            return response_text
        else:
            print(f"Request failed with status code: {response.status_code}")
    except requests.exceptions.RequestException as e:
        print(f"Request error: {e}")
```

[22] 由于选题限制，本文不深入阐述。有关 Django 和 Nginx 架构的内容见附录

[23] 详细代码见附录 request.py

4.3 访问智能算法模块的演示 demo

使用浏览器访问即可测试 demo 的当前版本。在服务器持续维护期间，访问 http://124.222.120.214/ai_voice_assistant_handle/此处添加问题[24]。等待响应，往往需等待 30~60 秒。由于在服务器端使用了代理技术，不需要任何特殊网络设置即可访问。该模型已经经过一定的和盾构机相关的训练，它已经了解包括：

- 用户的问题与盾构机和隧道相关
- 考虑到对话的连贯性，返回的内容不宜过长
- 对于相同问题尽可能返回相似答案（随机参数设置为了较为固定的值）

下述内容是一些提问案例和回答：

■ 124.222.120.214/ai_voice_assistant_handle/你可以帮到我什么

Response：我可以提供关于盾构机的基本知识、工作原理以及应用领域等方面的帮助。如有具体问题，我也会尽力为您解答。

■ 124.222.120.214/ai_voice_assistant_handle/盾构机隧道作业的安全注意事项

Response：排除地质风险，严格遵守操作规程，确保通风和救援措施，保证作业人员和设备的安全。

■ 124.222.120.214/ai_voice_assistant_handle/盾构机是谁发明的

Response：盾构机由日本工程师中村清次郎于 1963 年发明。

5. 结论和展望

本项目通过整合语音识别、文本生成和文本转语音技术，实现了一个基本的语音交互系统。作为结论，研究初步证明了中文语音助手开发的可行性，并提供了相关的可供相似项目或本例后续开发的资料和其来源。本文着重测试了多个主流开源 ASR 和 TTS 项目，并分析了他们在各个指标上的优劣。本文建议了开发此类助手的设计思路、技术选择，并实现了基本功能的演示，为后续此类项目的开发和研究提供了一定的依据。

作为延申和展望，这一系统在未来还具备如下潜能。

尽管在本项目中使用了 PaddleSpeech 作为语音识别和文本转语音模型，但随着深度学习和人工智能领域的不断发展，未来的语音技术仍有巨大的研究和应用潜力。在针对特定场景进行抗噪的领域，本例也涉及尚浅。探索更高准确率的 ASR 模型、更自然流畅的 TTS 模型，以及更好地处理多语言和口音的技术，都是进一步提升语音交互系统性能的方向。其中，对方言和行业术语的识别尤其具有创新价值。

本研究的重心在于语音助手的后端技术实现，但健全的应用软件往往需要优秀的用户端设计以提高交互性。这包括用户自定义的 UI 界面，以及针对场景的用户交互设计，这

[24] 如下述案例所示，使用“给定网址+输入”，并使用任意主流浏览器访问即可

也有利于技术的商业化。

6. 附录

各位老师好，这是 20 级王欣洲的暑期专业实习附录。

这份文本用于指出我所上交的文件的内容结构和我对它们的个人理解。

★ 这份文件夹中包含的文件并不是可执行的，它们没有执行的环境。该文件中包含的所有文件都是独立的，他们是从 6 个语音工程项目，测试项目，上线 demo（基于 nginx 的 django 项目）等多个工程中挑选出来的有代表性的程序。他们互相不依赖。仅作代码展示之用。

★ 在答辩过程中，我会在后台预备好报告中提及的所有流程的环境准备，以备演示。这也包括该文件中提及的所有代码的执行。

专业实习报告-20124444-王欣洲.docx - 是本人在专业实习期间研究课题的报告。由于该课题有明确的开发目标，所以也类似于开发报告。

asr_noise_accuracy_test.py - 进行对中文语音识别工程进行准确度（各项指标）测试时，所使用的程序。

audio_recording.py - 用户录音模块的主要程序。

chat_response.py - 智能回答算法模块的主要程序。向 OpenAI 发送请求并接收返回。这一部分事实上部署在了服务器上。

env_output.py - 由于软件开发在至少 3 个平台（Linux Ubuntu 虚拟机，Windows 主机，腾讯云代理的 Linux ubuntu 远程服务器）上进行，该脚本可以用于输出运行环境，作为记录，用于方便排除环境相关的错误。

launch.py - 即 main 函数所在的文件。

recognize.py - 展示了 ASR 模块函数的调用。此处选取了 PaddleSpeech 的 API 进行展示。但在测试中，由于有 6 个待测 ASR 工程，这个模块有 6 个不同版本。

tts_executor.py - 展示 TTS 模块函数的调用。基于同样原因，各个工程的该文件各不相同。

request.py - 本地向服务器发送获取 OpenAI 请求的程序。

espnet, masr, paddlespeech_noise_test.csv - 3 个 ASR 工程的部分测试结果。

nlp_transcript.py - 预处理 aishell_transcript 数据集的程序

nlp_processed_transcript.txt - 经过预处理的 aishell 数据集

aishell_transcript_v0.8.txt - aishell 原始数据集

★ 关于 demo:

我制作了 3 个版本的成品可供展示和下载。

1. （推荐：不包含 ASR 和 TTS 功能）仅访问智能助手的 AI 的方式（该模型已经经过了一定的和盾构机相关的训练）：

使用浏览器，以"http://124.222.120.214/ai_voice_assistant_handle/"+"您的输入"的格式直接进行 url 访问。

比如：”124.222.120.214/ai_voice_assistant_handle/你可以帮到我什么“。

等待约 30-60 秒即可获得回答响应（浏览器页面上会渲染回答），时间取决于服务器端的网络情况。该 IP 是长期维护的。

2. （包含完整的功能）Linux Ubuntu 本地版：

它包含完整的软件功能，包括使用麦克风作为输入，访问 AI，语音输出。虽然没有经过在其他平台上运行的测试，但理论上说，它自带了封装的虚拟环境，在 linux 系统上，使用”source venv/bin/activitate“激活虚拟环境，使用”python3 launch.py“即可在控制台运行程序。在 Windows 系统上，使用 activate.sh 激活虚拟程序，使用”python launch.py“运行。较为不便的是，这个版本不经过代理服务器，而是直接经由本地向 OpenAI 发送请求。这需要您的设备运行了全局代理，或在 OpenAI 包中进行相应的代理相关的代码的修改。

由于该版本较大，您可以通过下面的链接下载，或直接在微信上向我索取。

http://124.222.120.214/download/PaddleSpeech_0619

3. （包含完整的功能）Windows 本地版：

这个版本包含完整功能，与 Linux 版本不同的是，它借由我的云服务器向 OpenAI 发送请求。而云服务器本身运行代理。值得一提是，我本来租用的位于境外的服务器在最近恰好到期，才导致这一”代理+代理“的复杂方案。由于不熟悉 Windows 开发环境，这个版本还存在一些未知的问题，如果您感兴趣，可以在微信上向我索取。

★ 有关受限于题材未展示的部分实现，如 nginx 和 django 等，请参考我的个人博客：

Django: http://124.222.120.214/media/notion_files/080823/0808/Django_Basics.html

nginx 相关：
http://124.222.120.214/media/notion_files/070923/Nginx%20Access%20Log%20Generator%20ac0ae7cc599a40f09d36f4031e6e9063.html