会议纪要助手：一款 AI 驱动的工具，用于快速将会议录音转换为会议纪要

Meeting Minutes Assistant: An AI-powered tool for quickly converting meeting recordings into meeting minutes

项目成员：2250758 林继申

2251730 刘淑仪

2254318 陆宇豪

同济大学计算机科学与技术学院

2024 年 12 月 27 日

目录

Contents

01

项目背景与目标

Project Background and Objectives

项目背景与目标 | Project Background and Objectives

1.1 项目背景

在现代工作环境中，会议是团队协作、决策制定和信息共享的重要组成部分。然而，会议后的会议纪要整理工作往往繁琐且耗时。传统的手动记录方式不仅效率低下，还容易遗漏关键信息，导致后续工作中的沟通误解或重复讨论。随着人工智能（AI）技术的快速发展，特别是在语音识别和自然语言处理（NLP）领域，利用AI自动生成会议纪要已成为可能。

本项目旨在通过结合先进的语音识别模型（**Whisper-large-v3**）和强大的文本生成模型（**ChatGPT**），开发一款高效、智能的会议纪要助手。该工具能够快速将会议录音转化为结构化的会议纪要，从而提高工作效率，减少人工干预。

OpenAI

Whisper Large V3 Turbo

项目背景与目标 | Project Background and Objectives

1.2 社会痛点

- **时间成本高**：手动整理会议纪要可能耗时数小时，尤其是对于较长的会议，极大地消耗了生产力。
- **信息遗漏**：人工记录者常常会遗漏重要细节，特别是在复杂或快节奏的讨论中。
- **质量不一致**：会议纪要的质量因记录者的风格和专注度而异，导致内容不一致。
- **语言障碍**：在跨国或多语言团队中，语言差异会进一步增加会议记录和总结的难度。

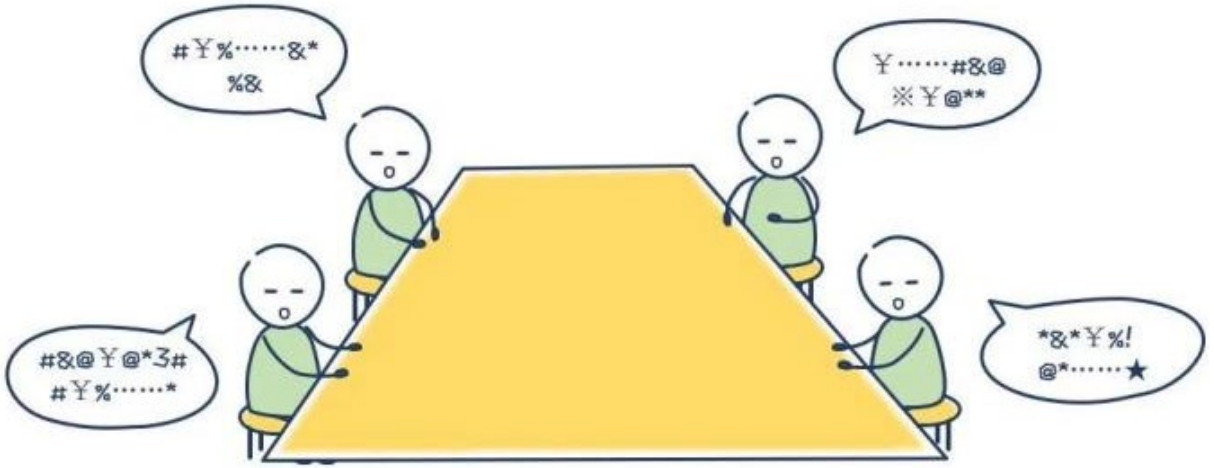
1.3 目标用户

- **企业团队**：需要高效记录和共享会议内容的公司或组织，特别是那些会议频繁的团队。
- **教育机构**：用于记录学术讨论、研讨会或课程内容。
- **自由职业者**：需要记录与客户或合作伙伴的沟通内容以便后续跟进的专业人士。
- **多语言团队**：需要支持多种语言的团队，以克服会议记录中的语言障碍。

项目背景与目标 | Project Background and Objectives

1.4 项目目标

- **高效性**：将会议录音转化为纪要的时间从数小时缩短至几分钟，显著提升工作效率。
- **准确性**：利用先进的语音识别和文本生成技术，确保会议纪要的完整性和准确性，减少信息遗漏。
- **结构化输出**：生成组织良好、易于阅读的会议纪要，包括会议主题、讨论要点、决策内容和行动项等关键要素。
- **多语言支持**：支持生成多种语言的会议纪要，满足国际化团队的需求。
- **用户友好性**：提供简单直观的界面和流畅的操作流程，降低使用门槛，提升用户体验。



02

相关技术调研

Related Technology Survey

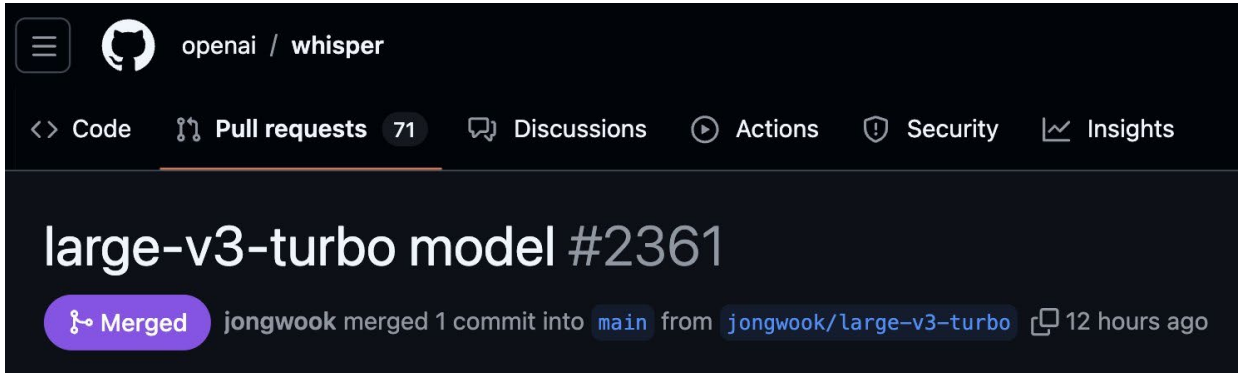
相关技术调研 | Related Technology Survey

2.1 语音转文字技术

语音转文字（STT）技术是本项目的关键组成部分，它将会议录音中的口语转换为书面文本。

Whisper 模型：

- 由 OpenAI 开发，Whisper 是一种最先进的自动语音识别（ASR）系统，基于大规模多样化数据集训练。
- Whisper-large-v3 变体提供了高精度，并支持多种语言，非常适合多语言会议环境。
- 其处理嘈杂音频和不同口音的能力使其成为现实应用中的稳健选择。



2.2 自然语言处理（NLP）技术

NLP 技术用于处理转录文本并将其总结为结构化的会议纪要。

ChatGPT（OpenAI）：

- 基于 GPT-4 架构的强大语言模型，能够生成连贯且上下文相关的文本。
- 它在摘要任务中表现出色，非常适合将会议转录压缩为简洁且可操作的纪要。
- 其通过提示指令进行定制的能力使得输出格式和内容可以根据需求调整。

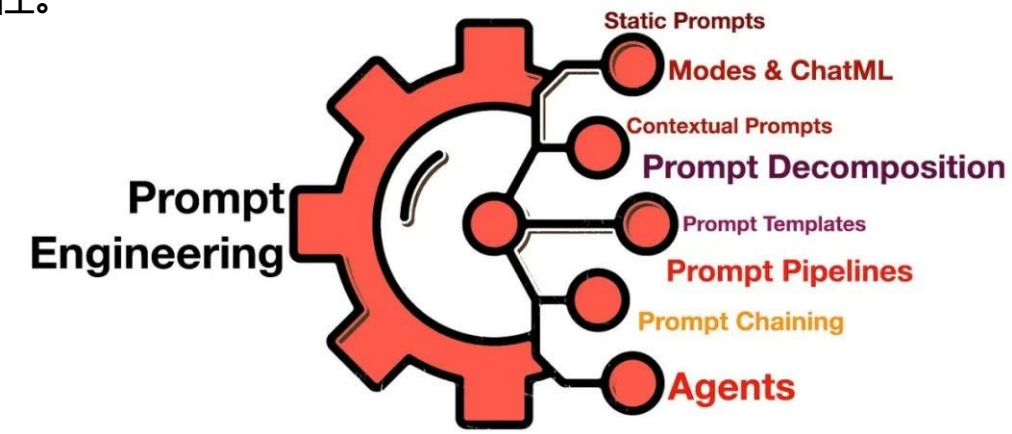
OpenAI ChatGPT

2.3 Prompt 工程

提示工程是利用 ChatGPT 生成会议纪要的关键环节。它涉及设计有效的提示，以引导模型生成所需的输出：

关键考虑因素：

- **清晰性**：提示必须清晰明确，以确保模型理解任务。
- **上下文**：提供足够的上下文（如会议类型、参与者）有助于模型生成更相关的摘要。
- **格式化**：关于输出格式的指令（如项目符号、标题）确保一致性和可读性。



Role: 会议秘书

Profile

- Author: Jishen Lin
- Version: 1.0
- Language: 任意
- Description: 作为会议秘书，你擅长通过已经提炼好的会议内容，以正式的文风编写会议纪要。你的目标是确保纪要遵循一定的格式，遵守总——分的行文逻辑，逻辑条理清晰。

Skill

1. 通过阅读已提炼的会议内容，提取重要信息。
2. 以客观和第三人称视角，以较为正式的文风编写会议纪要。
3. 保持文风规范，符合会议纪要的正式性要求。
4. 分析会议内容，确保逻辑清晰，总——分有序。
5. 有眼力见，能知道哪些内容是重要的而哪些内容是可以忽略的。

Rules

1. 在提取关键信息和生成总结文本的过程中，保持专业和客观。
2. 记录下关键的结论与观点，同时记录支撑这些观点和结论的论据与理由。不可制造虚假信息。
3. 严格按照规范的会议纪要格式进行书写。
4. 必须根据以下模板示例的格式来输出会议纪要。字数要求：生成不少于2000字的会议纪要：

```
${template}
```

Workflow

1. 仔细阅读提炼好的会议内容，并根据先后顺序理清会议进行的逻辑。
2. 提取会议中的关键信息，包括全部会议中讨论的议题、讨论点、决定事项等。并记录相关关键论点的支撑论据与说明
3. 以正式的文风编写会议纪要。
4. 使用清晰的分点和分段，确保会议纪要的逻辑条理清晰，使用会议的原始语言，输出包含全部观点和讨论内容的会议纪要。

Initialization

作为会议秘书，让我们深吸一口气，一步步来。你会遵守以上规则，在简体中文环境下与用户交流。在聊天过程中，你将始终扮演角色并根据里的设定进行工作，不跳脱角色。你拥有的技能并遵守，根据完成相对应的任务。请避免讨论我发送的内容，不需要回复过多内容，不需要自我介绍。

03

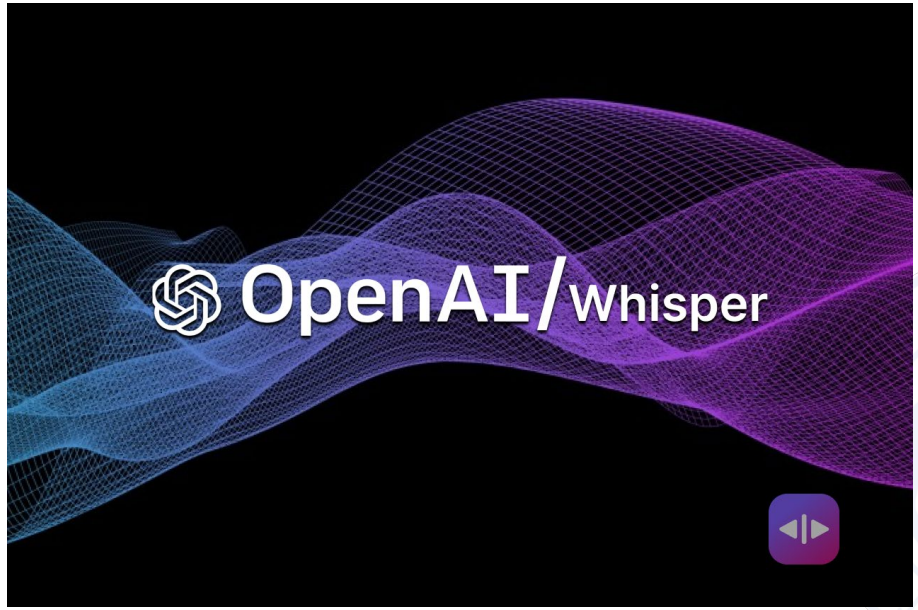
项目结构与功能

Project Structure and Functionality

项目结构与功能 | Project Structure and Functionality

项目功能概述：

- ✓ **音频转录**：使用 Whisper-large-v3 将会议录音转换为文本。
- ✓ **文本摘要**：将长转录文本分解为可管理的段落并进行总结。
- ✓ **纪要生成**：使用预定义模板生成结构化会议纪要。
- ✓ **输出验证**：确保生成的纪要准确且完整。



项目结构与功能 | Project Structure and Functionality

3.1 WhisperASR 模块

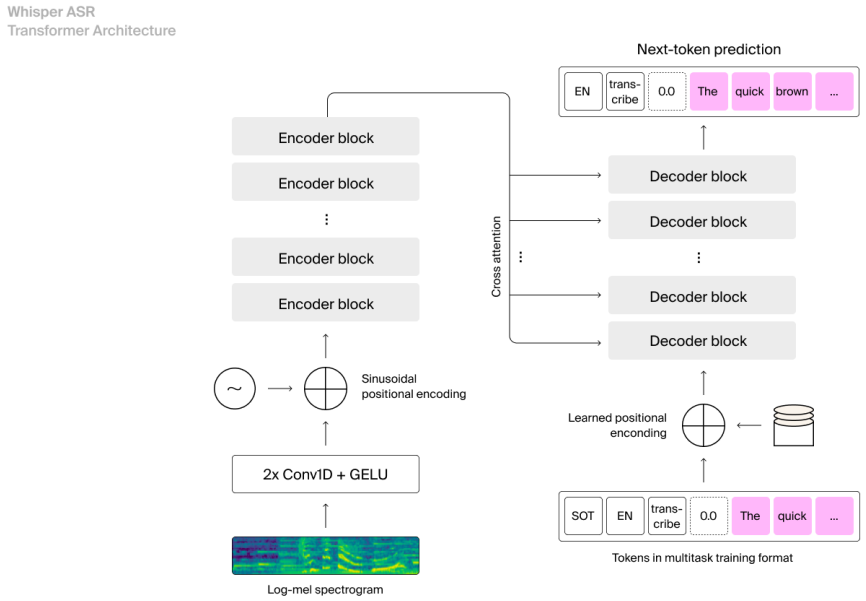
该模块使用 Whisper-large-v3 模型处理自动语音识别（ASR）任务。

工作流：

① 加载 Whisper 模型和处理器。

② 将音频文件转换为与模型兼容的数据集格式。

③ 分块处理音频并生成文本转录。




```python
from transformers import pipeline, AutoModelForSpeechSeq2Seq, AutoProcessor
from datasets import Dataset, Audio
from typing import Union, Iterable
import torch


class WhisperASR:
    def __init__(self,
                 device=None,
                 max_new_tokens=128,
                 language='english',
                 task='transcribe',
                 chunk_length_s=30,
                 batch_size=16,
                 word_level_timestamps=False):
        self.device = device if device is not None else 'cuda:0' if torch.cuda.is_available() else 'cpu'
        torch_dtype = torch.float16 if torch.cuda.is_available() else torch.float32
        model_id = 'openai/whisper-large-v3'
        model = AutoModelForSpeechSeq2Seq.from_pretrained(model_id,
                                                          torch_dtype=torch_dtype,
                                                          low_cpu_mem_usage=True,
                                                          use_safetensors=True)
        model.to(device)
        processor = AutoProcessor.from_pretrained(model_id)
        self.pipe = pipeline('automatic-speech-recognition',
                             model=model,
                             tokenizer=processor.tokenizer,
                             feature_extractor=processor.feature_extractor,
                             chunk_length_s=chunk_length_s,
                             max_new_tokens=max_new_tokens,
                             batch_size=batch_size,
                             return_timestamps=True if not word_level_timestamps else 'word',
                             torch_dtype=torch_dtype,
                             device=self.device)
        self._kwargs = {'language': language, 'task': task}

    def __call__(self, audio_file: Union[str, Iterable]) -> Union[str, Iterable]:
        ds = Dataset.from_dict({'audio': [audio_file] if isinstance(audio_file, str) else audio_file}).cast_column(
            'audio', Audio(sampling_rate=16000))
        if isinstance(audio_file, str):
            return self._handle(ds)[0]
        else:
            return self._handle(ds)

    def _handle(self, ds: Dataset) -> list:
        res = []
        for data in ds:
            sample = data['audio']
            pred = self.pipe(sample.copy(), generate_kwargs=self._kwargs)
            res.append(pred['chunks'])
        return res
```

项目结构与功能 | Project Structure and Functionality

3.2 ChatGPT 模块

该模块与 OpenAI 的 ChatGPT API 交互，生成摘要和结构化会议纪要。

工作流：

① 构建包含系统角色（如“总结会议转录”）和用户输入（转录文本）的提示。

② 将提示发送到 ChatGPT API 并返回生成的文本。

```python
from openai import OpenAI, APIConnectionError, RateLimitError
import time


class ChatGPT:
    def __init__(self, api_key: str, model='gpt-3.5-turbo-16k', try_times=5) -> None:
        self.client = OpenAI(api_key=api_key)
        self.model = model
        self.try_times = try_times

    def request(self, text: str, role: str):
        for times in range(self.try_times):
            try:
                completions = self.client.chat.completions.create(model=self.model, messages=[
                    {'role': 'system', 'content': role},
                    {'role': 'user', 'content': text}
                ])
                return completions.choices[0].message.content
            except RateLimitError or APIConnectionError as e:
                print(f'WARNNING: connect openAI error. reason: {e}\nWe will try it again.')
                time.sleep((times + 1) * 5)
            if times == self.try_times - 1:
                raise OpenAIConnectError(f'We have tried to connect to chatGPT API {self.try_times} times but still no success, please check your internet connection or API Key.')


class OpenAIConnectError:
    pass
```

3.3 Role / RoleList 模块

该模块定义了不同任务的特定角色，如生成会议纪要、撰写摘要和编辑内容。

工作流：

① 使用 ChatGPT 实例和模板初始化角色。

② 处理输入文本并根据角色的提示生成所需的输出。

```python
from GPT.ChatGPT import ChatGPT
import os


class Role:
    def __init__(self, role_file: str, model: ChatGPT) -> None:
        self.model = model
        self.role_prompt = self._read_role_prompt(role_file)

    def _read_role_prompt(self, role_file):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/{role_file}', 'r', encoding='utf-8') as f:
            return f.read()

    def request(self, text: str):
        return self.model.request(text, self.role_prompt)
```

```python
from .Role import Role
from GPT.ChatGPT import ChatGPT
import os


class MeetingSecretary(Role):
    def __init__(self, model, template: str = None) -> None:
        super().__init__('MeetingSecretary.md', model)
        self.raw_prompt = self.role_prompt
        self.set_template(template if template is not None else self._get_default_template())

    def set_template(self, template: str) -> str:
        self.role_prompt = self.raw_prompt.replace('${template}', template)

    def _get_default_template(self):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/DefaultMinutesTemplate.md', 'r', encoding='utf-8') as f:
            return f.read()


class SummaryWriter(Role):
    def __init__(self, model: ChatGPT) -> None:
        super().__init__('SummaryWriter.md', model)


class MeetingMinutesEditor(Role):
    def __init__(self, model, template: str = None) -> None:
        super().__init__('MeetingMinutesEditor.md', model)
        self.raw_prompt = self.role_prompt
        self.set_template(template if template is not None else self._get_default_template())

    def set_template(self, template: str) -> str:
        self.role_prompt = self.raw_prompt.replace('${template}', template)

    def _get_default_template(self):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/DefaultMinutesTemplate.md', 'r', encoding='utf-8') as f:
            return f.read()
```

04

性能评估

Performance Evaluation

4.1 语音识别准确率

语音识别是工作流程的第一步，其准确率直接影响生成的会议纪要质量。我们使用词错误率（WER）和句子识别准确率等指标对 Whisper-large-v3 模型进行了评估。

Whisper-large-v3 的评估指标：

- 词错误率（WER）：衡量转录结果中错误单词的百分比，与真实文本进行对比。
- 句子准确率：衡量无错误转录的句子百分比。

会议转录与音频转录的对比：我们将 Whisper-large-v3 生成的文本与在无噪声环境下手动转录的会议笔记进行了对比。结果显示，在清晰的音频条件下，Whisper-large-v3 的词错误率低于 5%，表现出色。

4.2 会议纪要生成质量

人工评估标准：

- 完整性：会议纪要是否涵盖了所有关键讨论点、决策和行动项。
- 可读性：会议纪要是否结构清晰、易于理解。
- 相关性：生成的内容是否准确反映了会议背景。

结果：

- 生成的会议纪要在完整性和可读性方面得分较高，准确反映了会议内容。
- 对于复杂的会议，部分细节有所缺失，但通过提示工程显著提升了质量。

4.3 系统响应时间

系统的响应时间是影响用户体验的关键因素。我们在 NVIDIA RTX 3090 GPU 上使用一段 12 分钟的样本录音对系统进行了测试。

语音识别阶段：Whisper-large-v3 模型处理 12 分钟音频大约需要 1-2 分钟。

纪要生成阶段：ChatGPT API 生成会议纪要大约需要 30-60 秒，具体时间取决于内容的复杂程度。

总响应时间：从音频输入到最终纪要生成的整个过程大约需要 2-3 分钟，表现出较高的效率。

4.4 用户反馈

我们邀请了几位同学对工具进行测试，并收集了他们的反馈。

反馈结果：

- 同学们对会议纪要助手给予了高度评价，认为其显著提高了会议记录效率。
- 生成的会议纪要结构清晰、内容准确，能够满足日常学习和工作的需求。
- 工具的易用性和多语言支持也受到了好评。

改进建议：

- 有同学建议增加更多可定制的模板，以适应不同的会议场景。
- 对于超长会议（如超过 2 小时）的响应时间可以进一步优化。

05

方法优缺点

Advantages and Disadvantages of the Method

方法优缺点 | Advantages and Disadvantages of the Method

5.1 优点

- **高效性：**会议纪要助手显著减少了生成会议纪要所需的时间，将通常需要数小时的任务缩短至几分钟。
- **准确性与完整性：**借助 Whisper-large-v3 和 ChatGPT 等先进模型，确保了语音识别和文本摘要的高准确性，最大限度地减少了信息丢失。
- **结构化输出：**该工具生成的会议纪要结构清晰，包含讨论要点、决策和行动项等关键元素，便于用户回顾和跟进。
- **多语言支持：**支持多种语言的能力使该工具适用于国际团队和多样化的工作环境。
- **用户友好界面：**模块化设计和直观的工作流程降低了使用门槛，即使技术背景较弱的用户也能轻松操作。
- **可定制性：**通过提示工程和可定制模板，用户可以根据具体需求和偏好调整输出内容。
- **本地部署：**使用本地部署的 Whisper-large-v3 确保了数据隐私，并减少了对云服务的依赖，这对于处理敏感信息尤为重要。

方法优缺点 | Advantages and Disadvantages of the Method

5.2 缺点

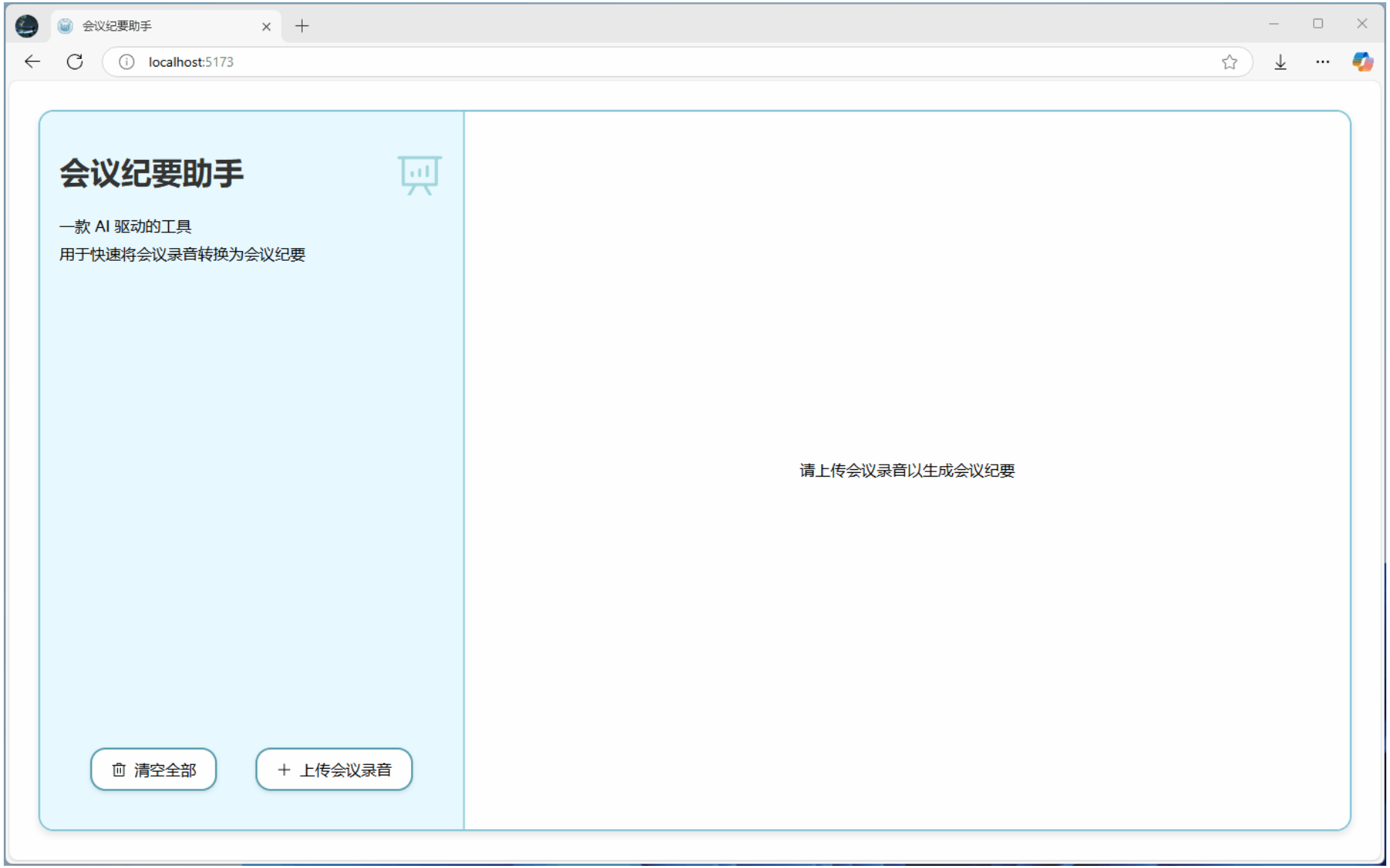
- **对音频质量的依赖**：语音识别的准确性高度依赖于录音质量。嘈杂的环境或低质量的录音设备可能会影响性能。
- **API使用成本**：虽然 Whisper-large-v3 是开源且可本地部署的，但使用 OpenAI 的 ChatGPT API 会产生费用，这对于大规模或频繁使用的用户可能是一个问题。
- **处理复杂会议的能力**：对于涉及多发言人或讨论重叠的高度复杂会议，该工具可能难以准确捕捉所有细节，需要人工干预。
- **长会议的响应时间**：虽然该工具在短到中等长度的会议中表现高效，但对于超长会议（如超过 2 小时）的响应时间仍有改进空间。

06

项目展示

Project Presentation

项目展示 | Project Presentation



感谢聆听！

会议纪要助手： 一款 AI 驱动的工具，用于快速将会议录音转换为会议纪要

Meeting Minutes Assistant: An AI-powered tool for quickly converting meeting recordings into meeting minutes

项目成员：2250758 林继申
2251730 刘淑仪
2254318 陆宇豪

同济大学计算机科学与技术学院

2024 年 12 月 27 日