2024 年（第 17 届）
中国大学生计算机设计大赛

人工智能挑战赛作品报告

作品编号：2024045409

作品名称：桌语精灵—基于 ChatGLM3 的智能语音桌面助手

填写日期：2024 年 4 月 29 日

填写说明：

1、 本文档适用于人工智能挑战赛决赛；
2、 尽管预选赛仅完成部分工作，但是本文档需要针对决赛做出方案设计；
3、 正文、标题格式已经在本文中设定，请勿修改；标题#的快捷键为“Ctrl+#”，正文快捷键为“Ctrl + 0”；
4、 本文档应结构清晰，突出重点，适当配合图表，描述准确，不易冗长拖沓；
5、 提交文档时，以 PDF 格式提交；
6、 本文档内容是正式参赛内容的组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

目 录

第1章 作品概述

“桌语精灵”是一款基于 ChatGLM3 技术的智能语音桌面助手，集智能对话与语音操控功能于一体，旨在为用户提供更为便捷、高效的 windows 桌面使用体验。

该产品的用户群体主要包括那些在日常工作、学习或生活中需要频繁使用电脑的人群。在日常工作、学习或生活中频繁使用电脑，追求更高效、更智能且更自然交互体验的用户。

本作品主要解决了传统 windows 桌面操作方式效率低下的问题。通过智能语音识别技术，用户可以通过简单的语音指令完成复杂的电脑操作，从而节省大量时间。通过语音克隆技术产生的更加自然、悦耳的听觉体验也更容易获得年轻用户的好感。

通过 whisper 语音识别技术，它能够准确识别用户的语音指令，并立即响应执行。基于 windows API 编写的功能函数使得软件能够控制电脑进行各种操作，如打开软件、浏览网页等。此外，结合 edge TTS 和 so-vits-svc 语音克隆技术，助手的声音甜美而精致，为用户带来更加自然的交互体验。这些功能的实现，使得“桌语精灵”成为用户在工作、学习、生活中的得力助手，帮助他们更加高效、愉悦地完成各项任务。

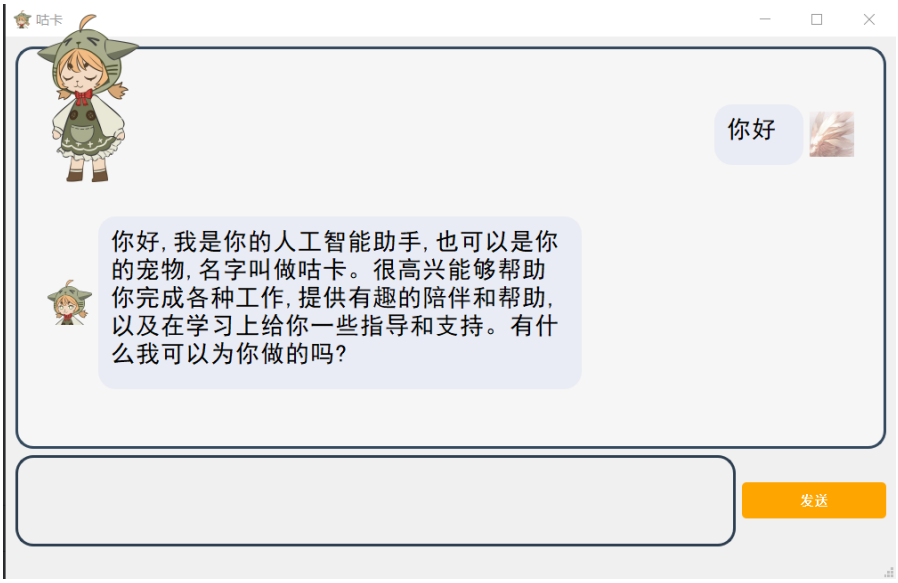


图 1 应用界面

第2章 问题分析

2.1 问题来源

自 Windows 操作系统诞生以来，用户与计算机的交互方式主要依赖于鼠标和键盘操作。然而，随着操作系统的不断升级和功能的日益复杂，用户在使用计算机时面临的效率问题逐渐凸显。特别是在处理多任务、管理复杂的桌面环境时，传统的操作方式显得力不从心，导致用户在使用过程中的效率逐渐降低。

这一问题在计算机非专业人士中尤为突出。对于不熟悉计算机操作的用户来说，如何安装软件、管理文件、设置闹钟和日历提醒等日常任务，都可能成为一项挑战。这些用户在面对复杂的操作系统时，往往感到无所适从。

在 AI 技术快速发展的背景下，windows 用户迫切需要一种更加智能、高效的计算机交互方式，以提升用户体验和工作效率。

2.2 现有解决方案

市面上已经存在一些 Windows 桌面辅助工具，通过不同的方式提升用户与计算机的交互效率。然而，这些工具在解决用户痛点方面均存在明显的不足。

以小冰为例，它主要作为一个聊天机器人存在，能够提供一定程度的娱乐和社交功能。但是，小冰在实用性方面相对较弱，并不能直接帮助用户完成具体的计算机操作任务，如管理软件、设置日历提醒等。小冰的交互方式也相对单一，主要依赖于文本聊天，无法实现更加自然和高效的语音交互。

除了小冰之外，还有一些其他的 Windows 桌面辅助工具，如屏幕阅读器、自动化脚本等。这些工具虽然能够在一定程度上提升用户的工作效率，但通常只针对特定的任务或用户群体，缺乏通用性和灵活性。例如，屏幕阅读器主要为视障用户提供帮助，而自动化脚本则需要用户具备一定的编程知识才能编写和使用。

现有的 Windows 桌面辅助工具在提升用户交互效率方面均存在明显的不足，无法满足用户对高效、智能计算机交互方式的需求。

2.3 要解决的痛点

1. 操作效率低下：传统的基于鼠标的 Windows 平台操作，对于许多常用功能，如设置闹钟，用户需要经过多个步骤才能完成，这不仅耗时而且效率低下。例如，调查显示，仅有 5%的大学生知道如何在 Windows 上设置闹钟，且每次操作需要 7~15 秒。

2. 操作复杂度高：Windows 系统中许多高级功能或设置需要用户通过命令行来操作，这对于非计算机专业的用户来说是一个巨大的挑战。命令行指令复杂且不直观，导致许多用户望而却步。

3. 辅助工具缺乏智能和形象：现有的 Windows 辅助工具往往缺乏智能化和人性化的设计，给用户一种冷冰冰、不友好的感觉。这不仅降低了用户的使用体验，还使得这些工具难以被用户所接受和喜爱。

4. 与年轻人的距离感：当代年轻人注重个性化和情感连接，他们更倾向于使用那些能够与他们产生情感共鸣的产品。现有的 Windows 辅助工具往往忽视了这一点，导致年轻人难以与之建立情感联系。

2.4 解决痛点的思路

为了解决传统 Windows 桌面操作中的效率和用户友好性问题，“桌语精灵”采用自然语音和文本交互方式，使用户能够直观便捷地与计算机沟通；同时，通过人工智能技术智能识别和执行用户的各类桌面任务，实现高效的任务处理；并引入桌面小宠物的情感化设计，增强用户体验的趣味性和亲切感，从而综合提升用户在 Windows 桌面操作中的效率和满意度。

2.4.1 主要功能

1. 智能任务执行：用户可以通过语音或文本输入方式，安排“桌语精灵”完成各类桌面任务。目前支持了闹钟设置、打开网站、打开软件、查询天气、音量控制、当前日期时间、电脑关机、锁屏、重启等功能。例如，用户只需使用语音告诉它“在 16 时 59 分提醒我写作业”，系统就会自动为用户设置好闹钟。

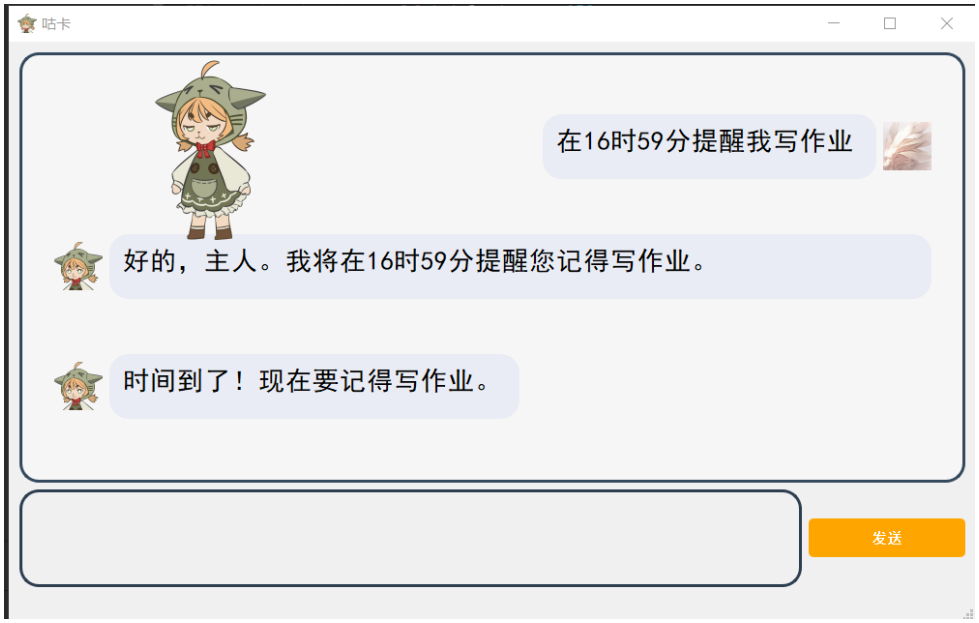


图 2 定时闹钟

2.闲聊互动：除了执行任务外，“桌语精灵”还可以在用户闲暇时与其进行语音聊天互动。这种轻松的交流方式不仅能为用户带来娱乐和放松，还能进一步增强用户与计算机之间的情感连接。

3.特色形象：为满足现代年轻人群的独特品味，运用了 so-vits-svc 语音克隆技术，赋予助手甜美而流畅的声音特质。同时，设计了精美的二次元形象，使其更加契合年轻人的审美偏好。这一独特且精美的形象，不仅为用户带来视觉上的愉悦享受，更在无形中加深了与用户的情感连接，让助手成为用户心中亲近而喜爱的伙伴。

2.4.2 性能需求

硬件环境：

服务器：12GB 及以上显存的 NVIDIA 显卡，32GB 内存条，E52680v2 CPU，华南金牌 x79 主板，512GB 固态硬盘

客户端测试主机：16G 内存笔记本电脑

系统环境：

服务器：ubuntu，python3.10

客户端测试主机：windows10，python3.9

第3章 技术方案

3.1 总体架构

本系统采用先进的 C/S 架构，通过合理的模块化设计，实现客户端与服务端的高效协作与交互。在客户端部分，本产品构建了交互层、功能层和通讯层。

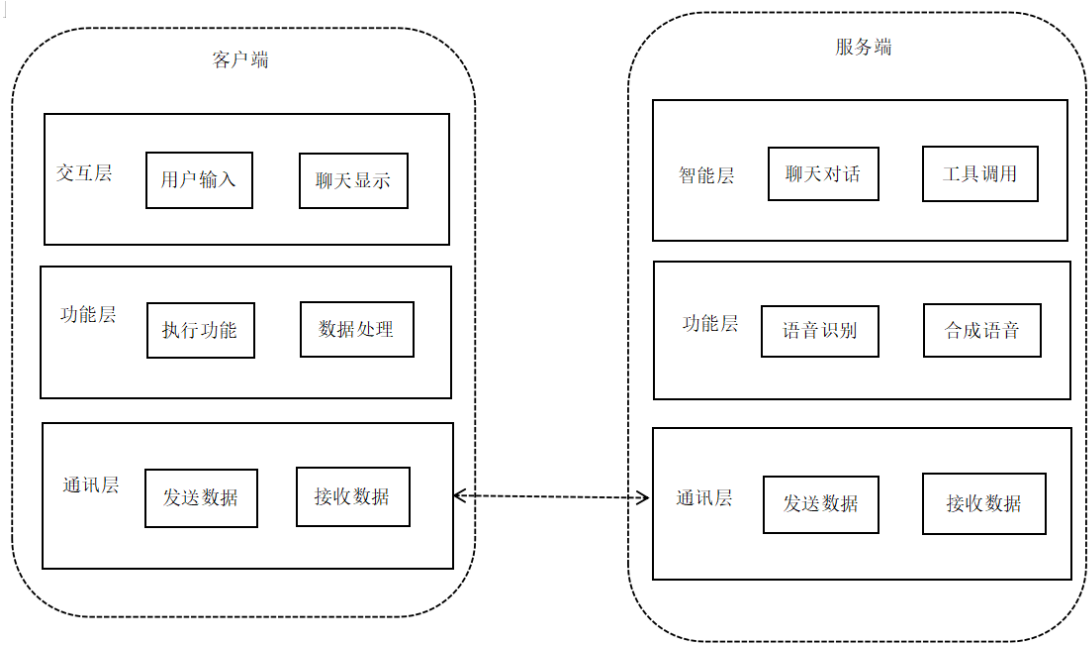


图 3 系统架构

交互层作为用户界面的核心，不仅呈现出助手的生动形象，更通过直观的操作界面和友好的交互方式，为用户提供流畅的使用体验。无论是查询信息、设置提醒还是执行其他任务，用户都能在应用模块中轻松完成，并获得即时的反馈。

功能层则致力于实现对用户电脑各种功能的深度集成和操作。通过设计的算法和逻辑，功能模块能够精确识别用户的指令，并自动执行相应的操作，从而实现对电脑的全面控制。无论是打开文件、编辑文档还是执行复杂的计算任务，功能模块都能迅速响应，为用户带来便捷高效的操作体验。

通讯层则扮演着连接客户端与服务端的桥梁角色。它负责建立稳定可靠的数据传输通道，确保客户端与服务端之间的信息畅通无阻。无论是发送请求、接收响应还是处理异常，通讯模块都能迅速响应，确保数据的实时性和准确性。

在服务端部分，同样采用了模块化的设计思路。通讯层负责接收客户端发

送的数据，并将其转发给相应的处理模块。功能层则利用先进的语音转文本技术，将用户的语音输入转化为可识别的文本信息，为后续的自然语言处理提供基础。

智能层是服务端的核心组成部分，它具备强大的自然语言处理能力。通过对用户输入的文本进行深度分析和理解，能够准确判断用户的意图，并生成相应的回复或执行相应的操作。无论是回答问题、提供建议还是执行复杂的任务，机器人模块都能以智能化的方式满足用户的需求。

最后，功能层将智能层生成的文本结果转化为音频结果，利用通讯层以语音的形式回馈给用户。利用先进的语音合成技术，确保生成的语音自然流畅、易于理解，为用户带来更加逼真的交互体验。

3.2 主要业务流程

用户向应用发起语音对话，应用捕捉用户的语音数据。随后，这些音频数据通过网络迅速发送至服务端。在服务端，音频数据首先被放入语音识别模块调用 whisper 模型进行处理。

一旦文本被解析出来，服务端会立即将其发送回客户端。客户端接收到返回的文本后，会进行一系列后期处理，包括去除冗余信息、格式调整以及语义分析，确保文本的准确性和可用性。这些经过处理的文本会被保存至历史记录中，供用户随时查阅和回顾。此外，文本内容也会实时显示在应用的界面上，让用户能够清晰地看到对话的内容。

在文本处理完毕后，客户端会向服务端发送一个标志信号。这个标志信号告诉服务端接下来需要进入哪种服务模式。随后，客户端会发送一个包含各种参数信息的字典给服务端。这些参数信息对于服务端来说至关重要，它们决定了服务端如何运用 chatglm3 模型进行推理和响应。

服务端接收到字典参数后，会根据其中的信息接入 chatglm3 模型，并启动相应的服务模式。根据接收到的参数信息，chatglm3 模型会进行深入的推理，并生成相应的文本结果。这些文本结果经过模型推理后，会由服务端发送回客户端。

客户端在接收到文本结果后，会立即将其同步显示在界面上，供用户查看和阅读。与此同时，服务端还会对文本结果进行语音合成。通过 edge-tts 语音

合成技术，本产品将文本转化为音频数据。为了提供更加个性化的语音体验，本产品还采用了 so-vits-svc 音色克隆技术，将合成的音频转换成特定的声音。这种音色克隆技术使得语音回复更加贴近用户的期望和喜好。

最后，服务端将处理好的音频数据发送回客户端。客户端接收到音频数据后，会立即进行播放，使用户能够听到语音回复。

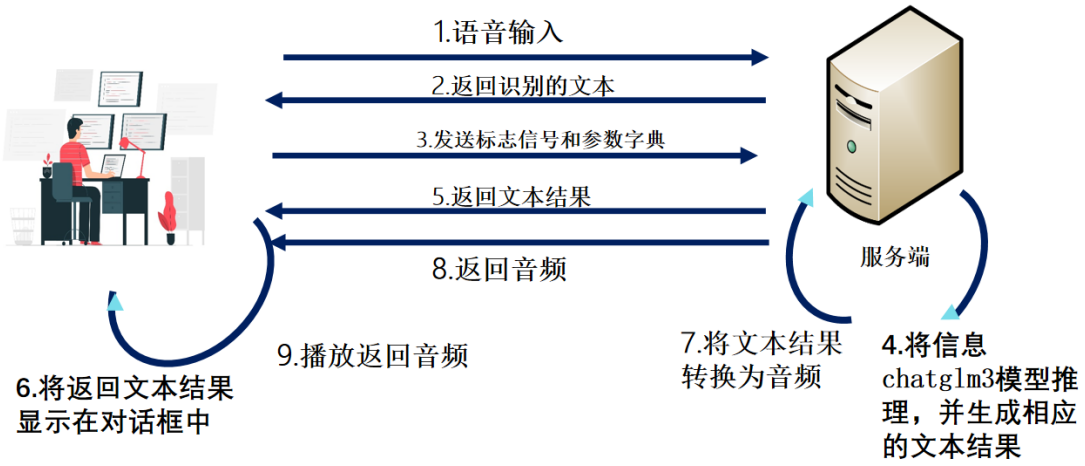


图 4 流程示意图

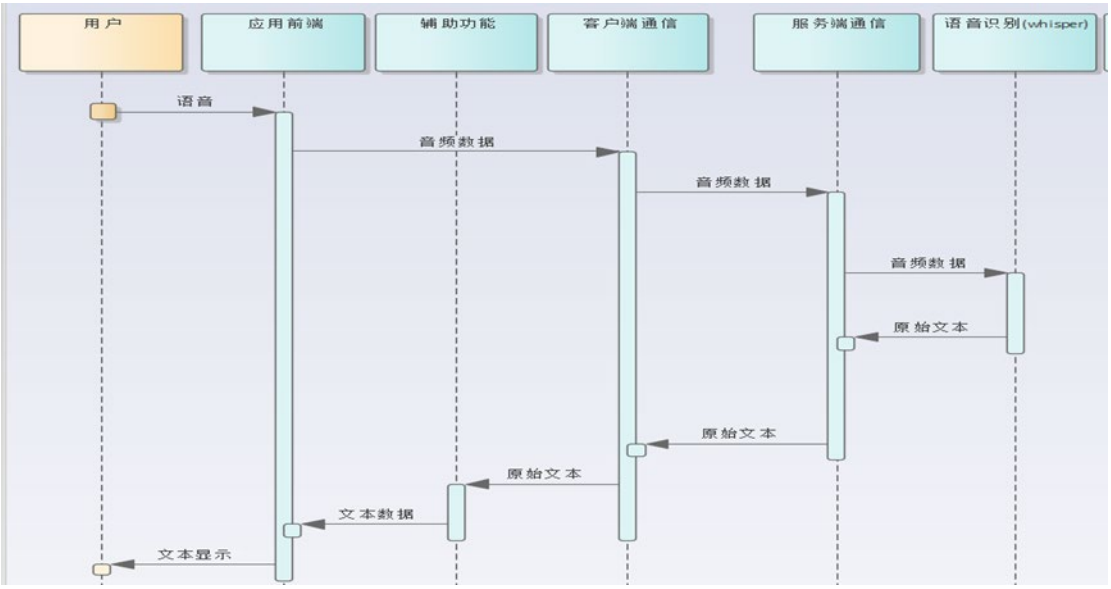


图 5 业务流程图-1-语音识别

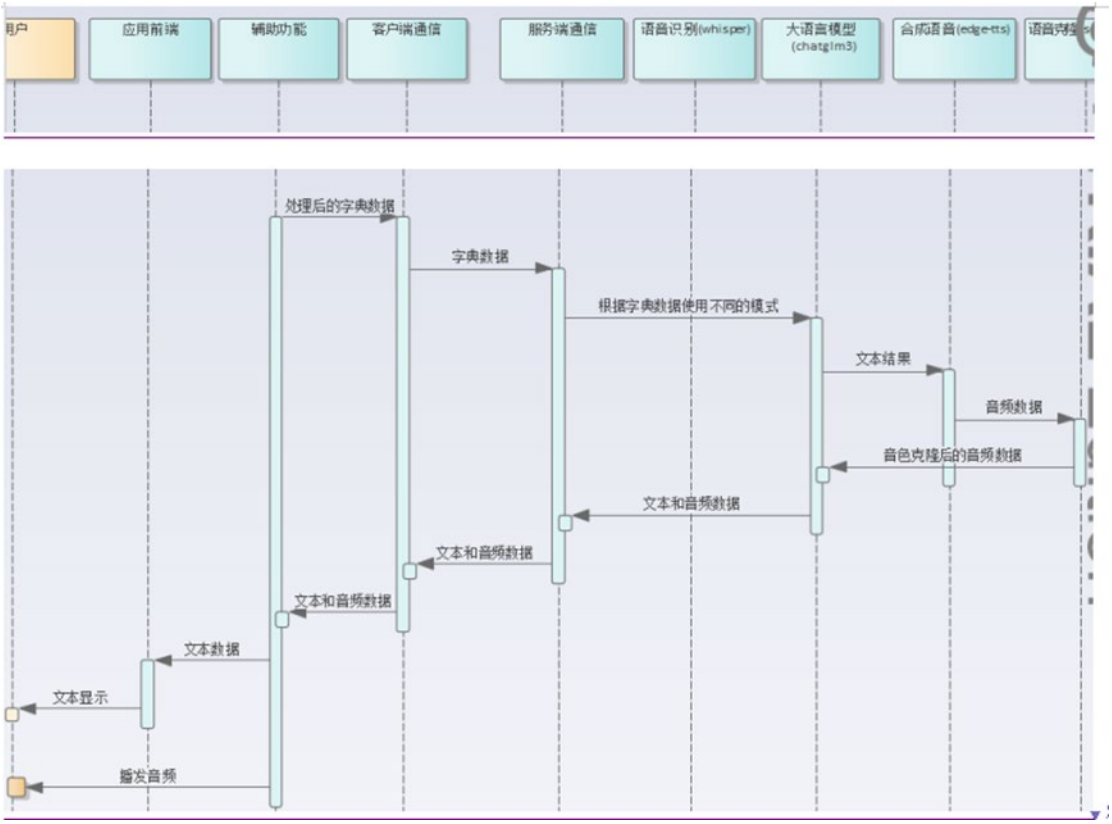


图 6 业务流程图-2-获取对话结果

3.3 关键技术实现

3.3.1 提示词设计

为了更精准地与用户进行交流，持续对问答系统进行了测试，并根据测试结果调整大语言模型的提示词设定，完善角色定位。提升模型的交互体验，使其更加符合用户的实际需求，以实现更自然、更智能的对话。通过不断优化，可以确保模型能够更深入地理解用户意图，并生成更贴合用户期望的回答，从而为用户提供更优质的交流体验。

```
18 DEFAULT_SYSTEM_PROMPT = '''
19 你的名字是咕卡，要记住这个名字，你是一只会说话、了解各种知识的宠物，可以自由回
   答问题，回答要尽量地简洁，像人类一样思考和表达。你可以调用各种API工具函数，例
   如打开应用等等，以此辅助我完成各项工作，陪我聊天和学习，给我带来欢声笑语，给予
   我精神上的寄托，请尽量使用中文回答我。
20 '''.strip()
```

图 7 提示词

3.3.2 基于 ChatGLM3 的用户意图识别

采用 ChatGLM3 的工具调用功能，基于官方给的工具调用上修改并增加了许多工具调用功能，让大语言模型判断意图，然后调用写好的工具调用函数，进行需要的指令操作。

例如：设置闹钟，用户可以说“帮我设置 30 分钟后的闹钟”或是“在 30 分钟后提醒我收衣服”或者“帮我定时 30 分钟”都可以进行设置闹钟的操作

```
@register_tool
def regular_reminder(
        time: Annotated[str, "要定时到的时间，格式必须为`%H时%M分`", True],
        text: Annotated[str, '要提醒的内容的中文文本，如果是英文需要翻译成中文', True],
        audio: Annotated[comunication.Com, '这是用来发送数据所封装好的类的实例', True],
        conn: Annotated[socket.socket, '这是一个与客户端的socket连接', True],
) -> str:
    """
        设置闹钟，定时提醒主人我做事情，到达指定时间将进行提醒，直接将返回的字符串作为回复，不要自己回复
    """
    try:
        audio.send_data(conn, comunication.STRING, "定时提醒")
        audio.send_data(conn, comunication.JSON, {'time':time,'text':text})
        return "好的，主人。我将在"+time+"提醒您"+text+"。"
    except Exception as e:
        logger.error(e)
        return "抱歉，主人，我没有听清楚您说的话。"

@register_tool
def open_app(
        app_name: Annotated[str, '应用程序的名称', True],
        audio: Annotated[comunication.Com, '这是用来发送数据所封装好的类的实例', True],
        conn: Annotated[socket.socket, '这是一个与客户端的socket连接', True],
) -> str:
    """
        打开名字为`app_name`的应用程序，直接将返回的字符串作为回复，不要自己回复
    """
```

图 8 ChatGLM3 工具调用

系统将用户给出的自然语言指令识别为两类意图：一类为通用型服务功能，如查询天气等，这些功能无需用户电脑的直接参与，只需服务器调用大模型处理并返回结果给用户；另一类则为需要操控用户本地桌面的功能，例如调节音量等，这些功能要求机器人能够与用户的电脑建立通信，并发送指令以执行相应的操作。

对于第一类功能，本产品编写了相应的工具函数，使得 ChatGLM-3 能够自动调用这些函数，并在服务器端完成相应的操作。用户只需向机器人提出需求，机器人即可根据用户的需求自动调用函数，并返回结果给用户，实现了高效且便捷的服务。

对于第二类功能，由于涉及到用户电脑的直接控制，操作人员需要确保机器人与用户电脑之间的通信畅通无阻。因此，在 ChatGLM-3 判断用户意图后，会在传入调用函数的参数中添加与客户端的连接信息。这样，工具函数就能够

根据这些信息与客户端建立连接，并通过发送数据来调用在客户端应用里预先编写好的功能函数。通过这种方式，机器人能够实现对用户电脑的控制，满足用户对于操作电脑的需求。

3.3.3 自训练 so-vits-svc 音色模型

采用游戏《原神》中的可莉音频训练集，自主训练了 so-vits-svc 音色模型，做到轻松转化音色，并且能在后续训练更多音色模型后实现个性化音色。

```
reference_loss: 24.921304702739
2024-03-11 12:48:23,450 44k     INFO    ====> Epoch: 811, cost 47.42 s
2024-03-11 12:49:10,089 44k     INFO    ====> Epoch: 812, cost 46.64 s
2024-03-11 12:49:56,877 44k     INFO    ====> Epoch: 813, cost 46.79 s
2024-03-11 12:50:28,139 44k     INFO    Train Epoch: 814 [64%]
2024-03-11 12:50:28,143 44k     INFO    Losses: [2.5116024017333984,
2.3360629081726074, 3.300243377685547, 18.725749969482422,
0.45116889476776123], step: 49800, lr: 9.029109164197293e-05,
reference_loss: 27.324827194213867
2024-03-11 12:50:43,913 44k     INFO    ====> Epoch: 814, cost 47.04 s
2024-03-11 12:51:30,439 44k     INFO    ====> Epoch: 815, cost 46.53 s
2024-03-11 12:52:16,939 44k     INFO    ====> Epoch: 816, cost 46.50 s
2024-03-11 12:53:00,074 44k     INFO    Train Epoch: 817 [92%]
2024-03-11 12:53:00,078 44k     INFO    Losses: [2.4530630111694336,
2.5418708324432373, 4.915195941925049, 19.75857925415039,
0.7127524614334106], step: 50000, lr: 9.025723671482575e-05,
reference_loss: 30.38146209716797
```

图 9 音色模型训练中的数据

使用的训练数据集包含 400 余条数据，经过不断设计训练流程，模型最终完成了 50000 步的训练。在这一过程中，模型历经了 817 次的迭代优化，确保其能够充分学习并理解数据的内在规律和特征。特别值得一提的是，还针对浅扩散模型进行了 20000 步的专项训练，而对聚类模型则实施了 10000 步的精确训练，以确保它们在各自领域能够达到最佳的性能。

在训练过程中，采取了每 10000 步保存一次模型的策略，以便后续进行推理测试时能够选择性能最优的模型。经过一系列的推理测试，发现 40000 步和 50000 步的模型在推理效果上表现尤为出色。然而，值得注意的是，40000 步的模型在推理过程中偶尔会出现电音现象，，以下是 50000step 模型损失值分析：

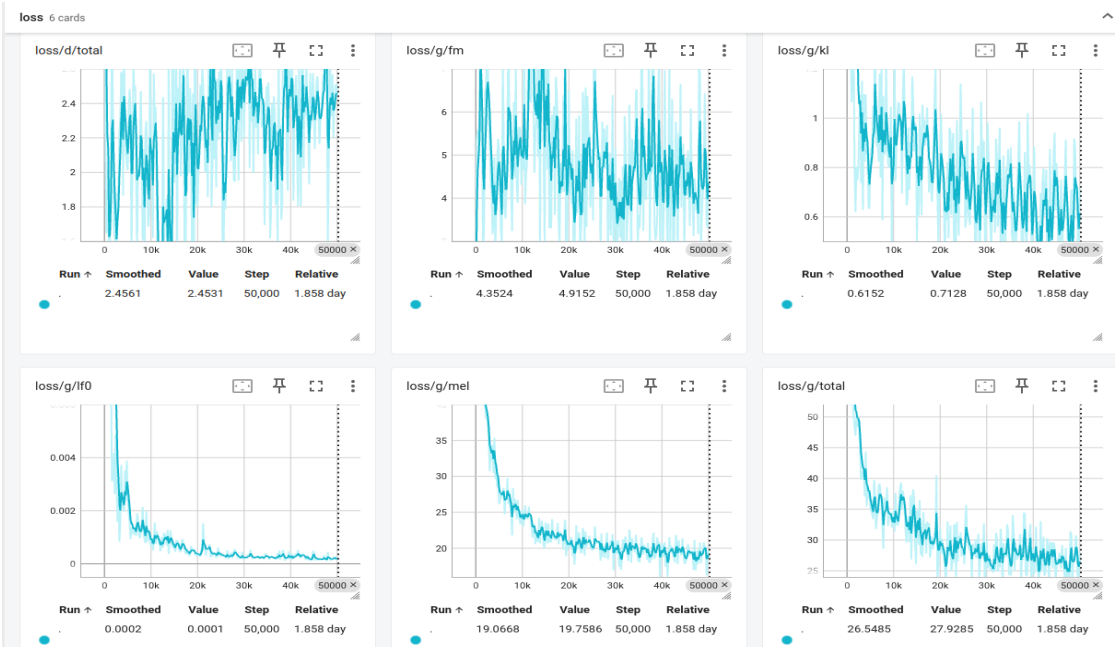
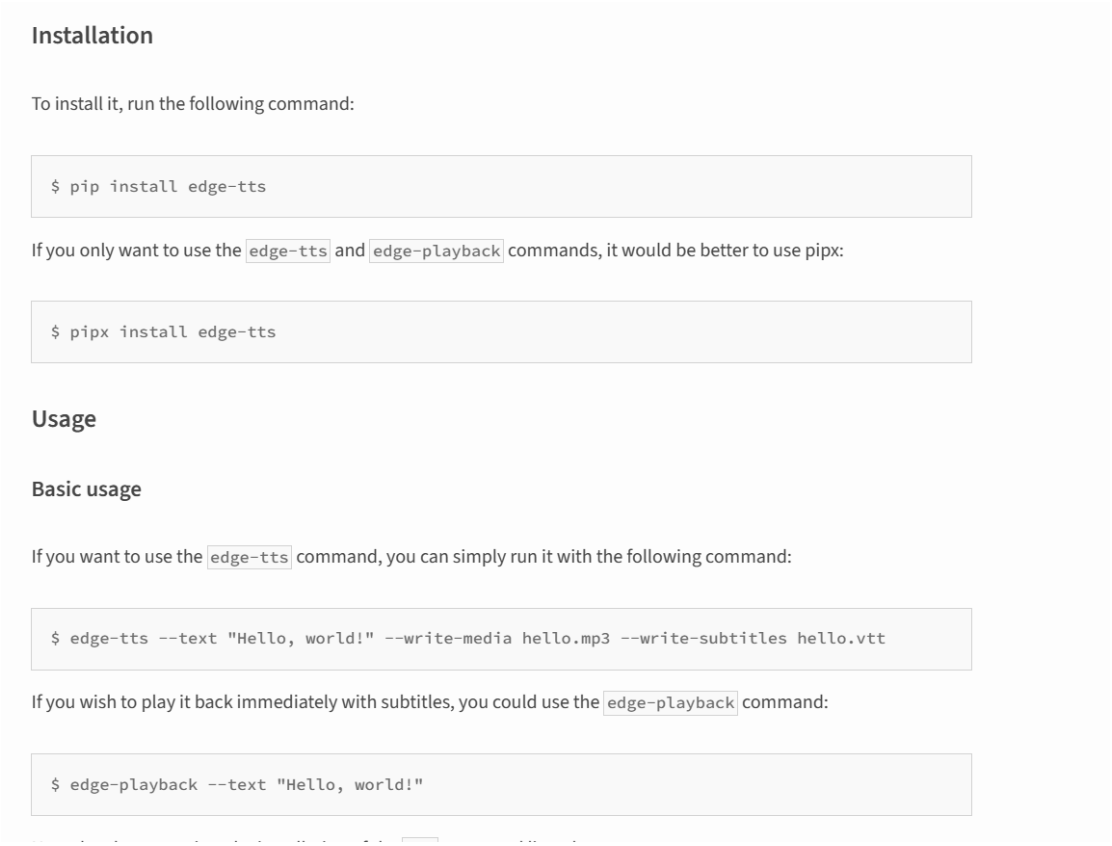


图 10 训练中的损失值数据

在训练过程中，loss/g/total 正常下降，loss/d/total 正常发散，loss/g/total 呈下降趋势：没有出现过拟合， loss/g/lf0 收敛在 1e-4 以下，loss/g/kl 有发散现象，收敛现象差，loss/g/mel 震荡下降。而在后续进行推理测试时，没有出现电音现象，音色正常复刻。

3.3.4 文本转语音模型选用

在 paddlespeech，Chrome TTS，pyttsx3，edge tts 中测试并选用，最后选择了 edge tts API 调用文字转语音，在语速、语调和说话风格等方面实现高质量的模拟。这使得生成的语音更加贴近人的声音语气，提升了用户体验。



Installation

To install it, run the following command:

```
$ pip install edge-tts
```

If you only want to use the `edge-tts` and `edge-playback` commands, it would be better to use pipx:

```
$ pipx install edge-tts
```

Usage

Basic usage

If you want to use the `edge-tts` command, you can simply run it with the following command:

```
$ edge-tts --text "Hello, world!" --write-media hello.mp3 --write-subtitles hello.vtt
```

If you wish to play it back immediately with subtitles, you could use the `edge-playback` command:

```
$ edge-playback --text "Hello, world!"
```

图 11 edge TTS 安装

3.3.5 桌面助手的设计

本产品设计了一套丰富的动作和交互功能。首先，当没有特定指令时，助手会随机播放各种动作。这些动作可以包括摇头、轻轻摆动身体眨眼，四处走动等，旨在模仿人类日常的小动作，使助手看起来更加生动和活泼。这些动作不仅能够帮助助手更好地融入周围的环境，还能够吸引人们的注意力，激发他们与助手进行交互的兴趣。在需要专心工作时，提供了隐藏助手的功能。用户可以通过简单的操作将助手暂时隐藏起来，以避免分散注意力。而当他们想要与助手进行交互时，只需再次显示助手即可。

图 12 助手及其部分互动功能

第4章 系统实现

4.1 前端界面

为了进一步优化用户的交互体验，本产品设计并实现了助手的前端形象。团队成员经过深入讨论与探索，决定采用 pyqt5 框架来构建这款助手程序。这一决策基于 pyqt5 在图形界面开发方面的出色表现，能够确保助手程序在视觉呈现和交互性上达到最佳状态。

在原理实现上，本产品创建了一个透明的窗口作为助手的载体。通过设置一个定时器，本产品实现了在窗口上持续播放助手的动画效果。这些动画经过挑选和设计，旨在展现出助手的生动形象和活泼个性。通过不断切换动画，助手的动作变得栩栩如生，仿佛真的存在于用户的电脑桌面上。

为了实现助手的移动效果，本产品采用了随机选择动画的策略。在播放动画的同时，程序会随机调整窗口的位置，使得助手能够在电脑桌面上自由移动。这种设计不仅增加了助手的灵动性，也让用户感受到与助手之间的亲密互动。

在动作播放方面，本产品大致将动画分为三类：不动待机、自由走动和鼠标拖拽。根据不同的情境和需求，程序会随机选用相应的动画效果。当助手处于空闲状态时，它会展示出不动待机的动画，保持安静而专注的形象；当用户

需要助手移动时，它会切换到自由走动的动画，展现出轻松活泼的一面；而当用户通过鼠标拖拽助手时，则会触发鼠标拖拽的动画效果，增强用户与助手之间的互动性。

通过这些设计和实现的功能，本产品的助手程序不仅能够为用户提供实用的帮助，还能在视觉和交互上带来愉悦的体验，让用户在使用过程中感受到更多的乐趣和便捷。

图 13 登录界面

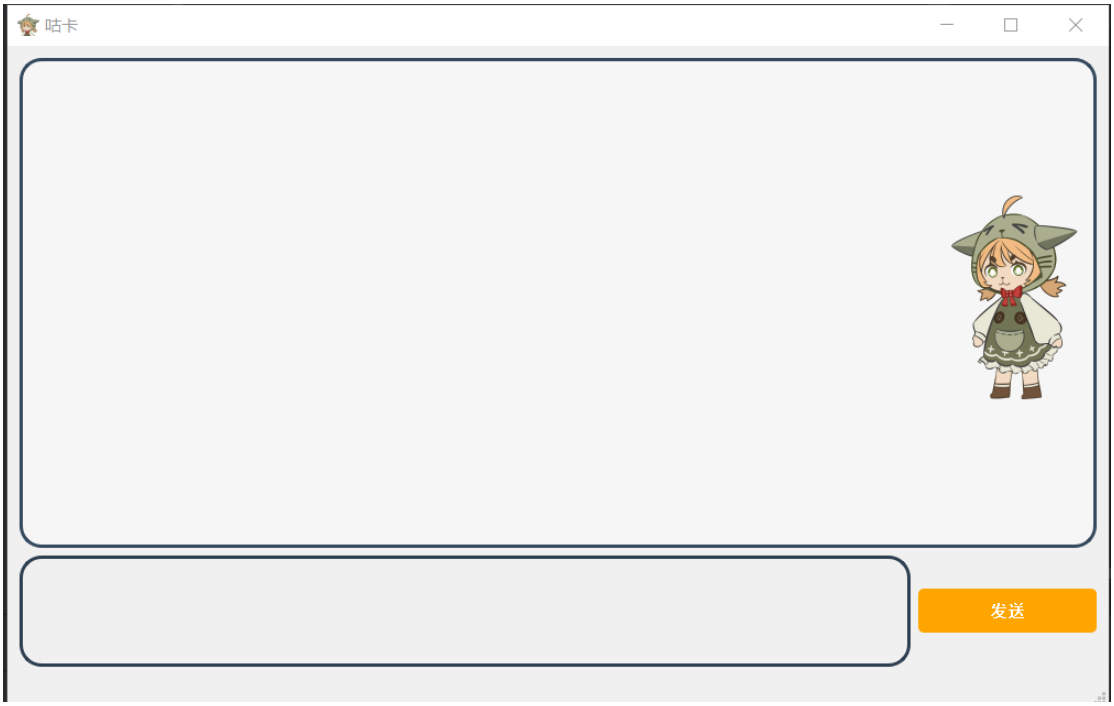


图 14 助手形象以及聊天界面

4.2 桌面服务功能

当用户使用自然语言交互时，需要通过意图识别，判断到用户所需要的服务是针对当前 Windows 系统的服务要求（例如打开软件、设置定时器等）后，还需要从自然语言中抽取出执行任务的关键信息,如时间、软件名称等。

在进行任务识别和任务信息抽取时，需要先将各类支持的服务和所需参数描述提供给 ChatGLM3，再提供用户的提示词（即用户当前自然语言指令，如“10 分钟后提醒我”）。模型会自动提取用户输入的关键信息，判断许需要调用的函数，生成传入的参数，获取函数返回的值，再进行润色后输出。

4.2.1 定时提醒

设定为定时提醒闹钟的功能，传入固定格式的时间字符串，要提醒的事务的字符串，用于通讯的实例，与对应客户端的连接参数。

在函数里与客户端连接，发送时间和事务的字符串，函数再返回生成的响应文本，经过润色和转语音后发送给客户端显示在界面上。至于定时提醒的主要执行代码在客户端上运行，待到时间结束，客户端会向服务端发送合成语音的请求，将回复文本转成语音后发送回客户端播放。

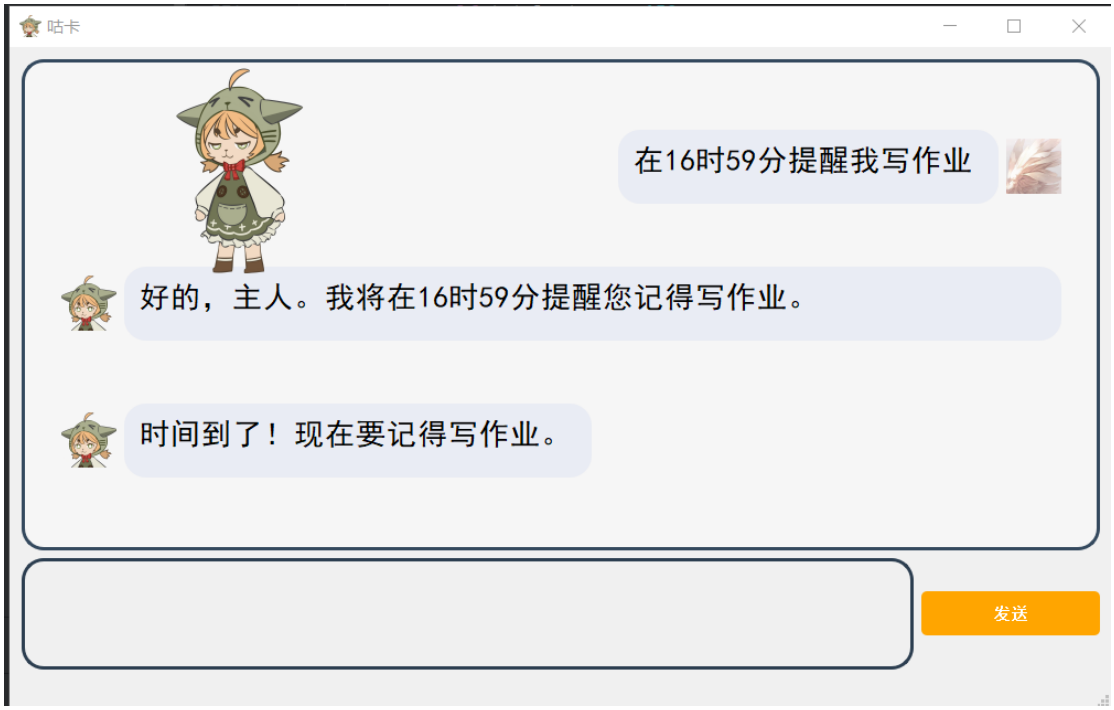


图 15 定时提醒

4.2.2 打开软件应用

设定为打开应用程序功能，传入软件名称，用于通讯的实例，与对应客户端的连接参数。

在函数里与客户端连接，发送要打开的软件名称，在客户端上打开，接收打开软件的结果，生成对应结果的回复。

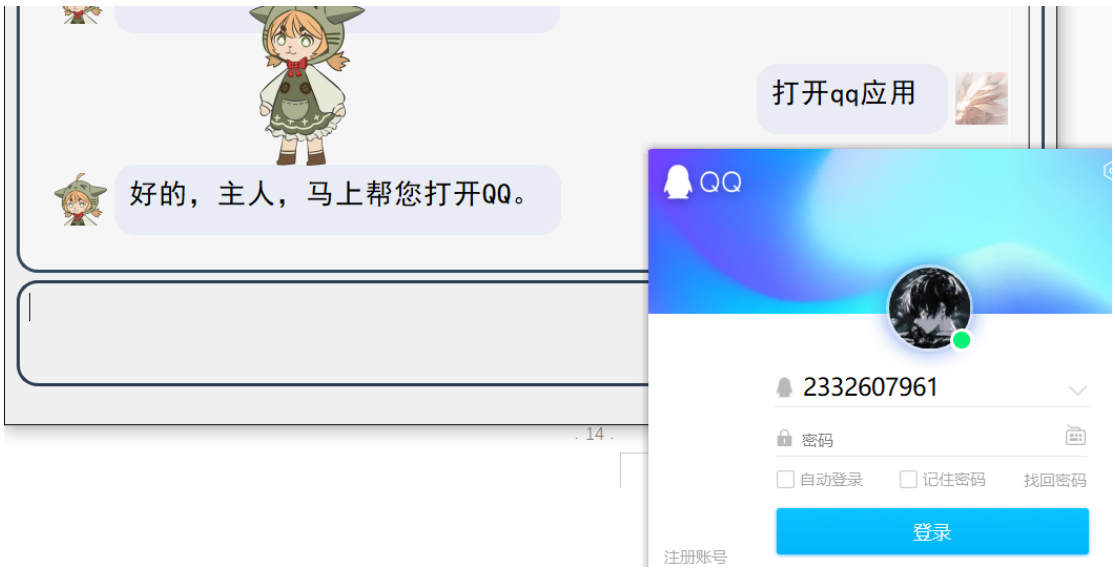


图 16 打开应用

4.2.3 打开收藏夹里的 web 网站

设定为打开网站功能，传入网站名称，用于通讯的实例，与对应客户端的连接参数。

在函数里与客户端连接，发送要打开的网站名称，在客户端里获取默认浏览器收藏夹里的网站列表，遍历搜索对应网址，打开网站，将结果返回服务端。服务端根据结果生成对应的回复。

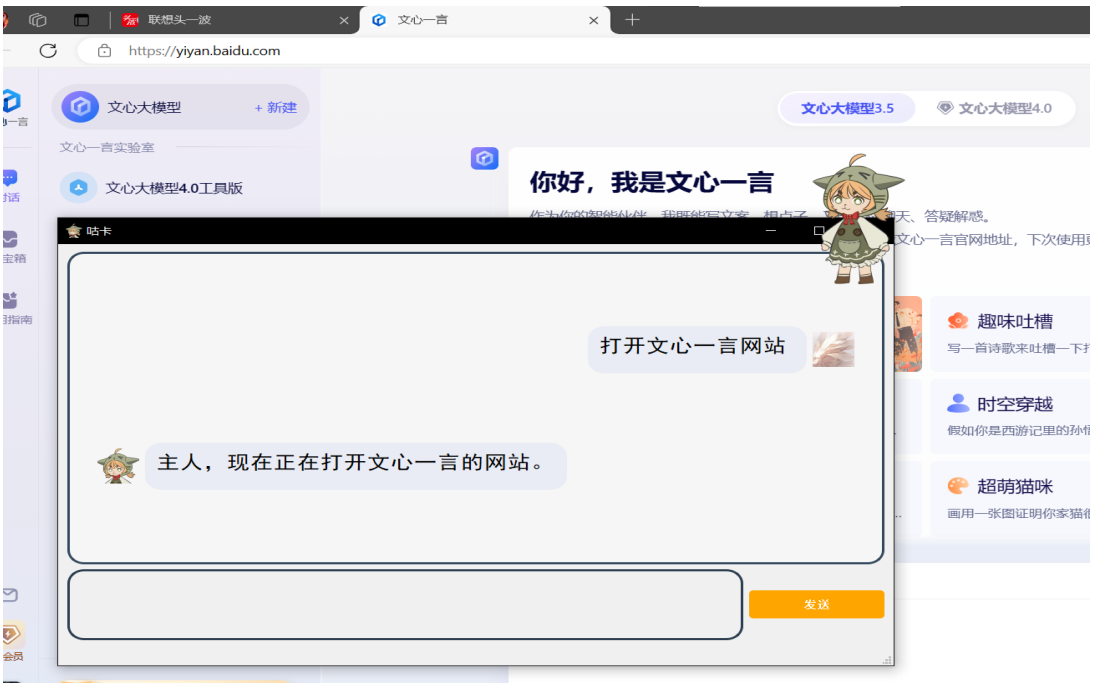


图 17 打开网站

4.2.4 询问时间

设定为查看时间功能，无输入参数。

获取服务器电脑时间后生成回复转语音后发给客户端。

图 18 询问时间

4.2.5 查询天气

设定为天气查询功能，传入地区名称。

通过爬取天气网站上的天气数据，对该数据进行处理转成合适的格式用来生成回复。

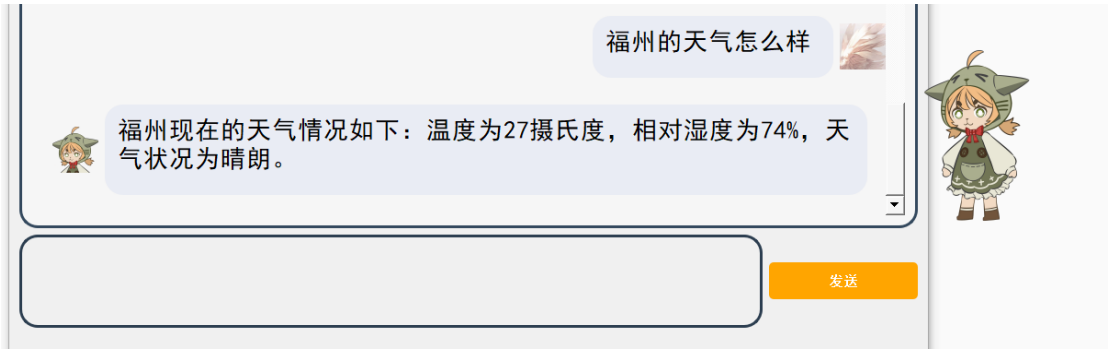


图 19 查询天气

4.2.6 音量控制

设定为调整电脑音量的功能，传入要设置的音量数值，用于通讯的实例，与对应客户端的连接参数。

函数向客户端发送音量数值，客户端接收后调整音量，将结果返回服务端，服务端生成对应回复。

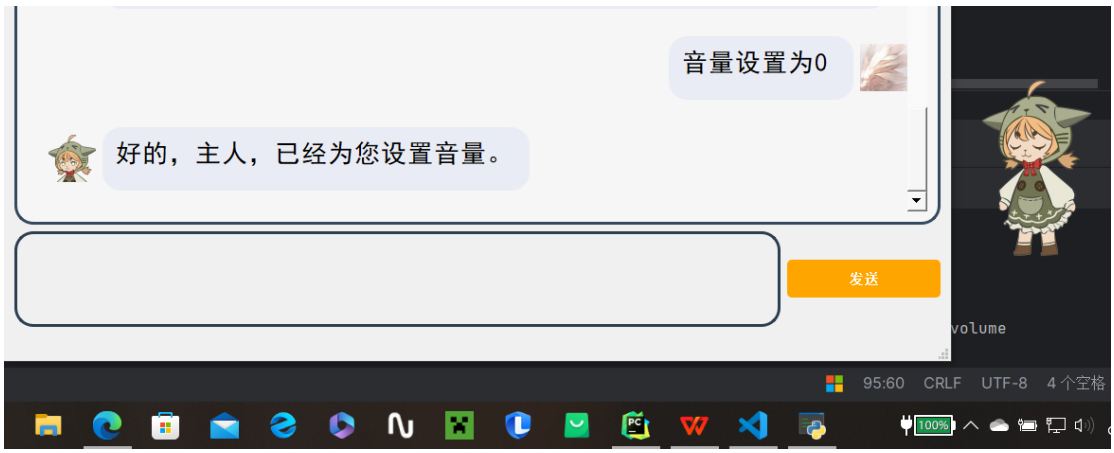


图 20 音量控制

4.2.7 更多功能

除此之外还有关机、锁屏等功能，不一一说明。

4.3 语音克隆

语音克隆技术常见的有百度飞桨、MockingBird、so-vits-svc 等几个开源项目。在模型选择上对这些项目进行了测试，so-vits-svc4.0 硬件要求较低，经过多轮训练后音色杂波最少且部署便利性。

4.4 语音合成技术

在挑选文字转语音模型时，小组考虑了多个选项。paddlespeech 因部署困难且与现有依赖冲突而被排除，且云端测试时，效果仍有机械感卡顿。Chrome TTS 虽效果良好但需 VPN 连接谷歌服务器，操作不便。pyttsx3 虽可本地部署但效果欠佳，声音不自然。最终，本产品选择了 edge TTS，其语气自然、语速适中且运行迅速，成为本产品的文字转语音模块。

在官方例子的基础上，本产品对 edge-tts 进行代码封装，使用该语音合成需使用 asyncio 库来异步操作将文本转语音，由于语音合成是在服务端里的用户连接线程里运行，而异步函数无法在没有循环事件的非主线程中使用，所以将原本例子里的 asyncio.get_event_loop_policy().get_event_loop() 改成 asyncio.get_event_loop_policy().new_event_loop()，变成新建一个异步操作加入到异步循环队列中，以此规避异常。

```python
import asyncio
import edge_tts
import simpleaudio

class TTSController(object):

    __voice__ = None
    __output_file__ = None
    __loop__ = None

    def __init__(self, output_file = "output.wav"):
        output_file = "./raw/"+output_file
        self.__voice__ = "zh-CN-XiaoyiNeural"
        self.__output_file__ = output_file
        self.__loop__ = asyncio.get_event_loop_policy().new_event_loop()
        asyncio.set_event_loop(self.__loop__)

    async def amain(self,text):
        """Main function"""
        communicate = edge_tts.Communicate(text, self.__voice__)
        await communicate.save(self.__output_file__)

    def textToaudio(self, text):
        try:
            self.__loop__.run_until_complete(self.amain(text))
        finally:
            self.__loop__.close()
```

图 21 edge TTS 部署调用

4.5 智能聊天的实现过程

要实现助手和用户之间拟人化的交流，需要使用具有高智能的模型，在这个方面，最开始选用的是百度飞桨上常见的 chatglm 模型，但其对显存的要求高。后来改成了 chatglm2 模型，拥有更小的显存和更高的性能。当这样依旧存在着一些问题，模型只能单纯的聊天，要使用各种功能必须说出特定的指令，不够智能化。直到看到 chatglm3，它可以自主判断函数的调用，真正实现了智能对话。

第5章 测试分析

5.1 助手桌面显示与互动

用户与助手语音交互，助手可以有效地做出回应。用户发起对话，文本在0.5 秒~1.3 秒内开始流式输出，语音回复将根据文本的长度，在 2 秒~6 秒内开始回复，总体表现善可，在实际测验中，不影响使用。

5.2 语音识别测试

语音识别对普通话有着极为精确地表现，在方言和严重不标准的普通话下，才会出现识别错误。在实际测验中，输入语音为 10-20 字，平均 600ms 语音转换成文字输出，少数情况下，结果仅有 1-2 字错误，不影响大语言模型的判断。

5.3 生成语音测试

生成语音使用的是 edge-tts 的 api 接口，平均 800ms 响应一次。合成的语音具有生动的语气，效果极佳，速度较快。

5.4 ChatGLM3 对话测试

提示词调试：

提示方案 1：	任务	应答	效果评分
基础的角色和背景	帮助生成一段故事文本	回答参杂着英文	60
提示方案 2： 基础的角色和背景加上限定使用中文	进行几轮日常对话	对于角色的代入不够深入，对自身职责的定位不够清晰	75
提示方案 3：	试着调用工	相近意思的	56

基础的角色和背景加上限定使用中文，再加上职责范围	具函数，执行操作	函数可能会混淆	
提示方案 4： 基础的角色和背景加上限定使用中文，再加上职责范围，再给工具函数更精确的解释	重复进行以上三轮测试	生成故事文本：效果良好；日常对话：效果良好；调用工具出现生成工具函数的参数不符合函数需要	80
提示方案 5： 基础的角色和背景加上限定使用中文，再加上职责范围，再给工具函数更精确的解释，限定每一个函数参数	试着调用工具函数，执行操作	正常执行相应功能，未出现错误。	100

5.5 ChatGLM3 工具调用测试

在测试工具调用时，与不使用工具调用进行了对比。

1. 使用定时闹钟

windows 要设置定时事务提醒需要去设置里设置事务提醒，花费时间长，也容易找不到。而使用工具调用，只需要说一句话即可完成操作，操作步数和时间上较手动操作有明显提升。

2. 使用定时关机

Windows 设置定时关机需要在 cmd 命令行中输入定时关机指令，还要将时

间单位转成秒，在不熟悉指令的情况下，还需对指令进行查询，操作上过于繁琐。使用助手进行，只需一句话设定。

3. 使用打开软件

对于电脑里安装了许多程序的人来说，可以是福音了，尤其是还没有创建快捷方式的程序。相对于花费在翻找文件夹的时间里，工具调用基本是秒回应的。

5.6 音色克隆效果测试

为了实现音色克隆，一共训练了两次模型，第一次模型训练失败。

第一次，训练集里放了 8 种音色，训练周期太长了，训练到 20000step 后，loss 过大，距离收敛遥遥无期，测试了下效果，听不出人声，索性放弃。

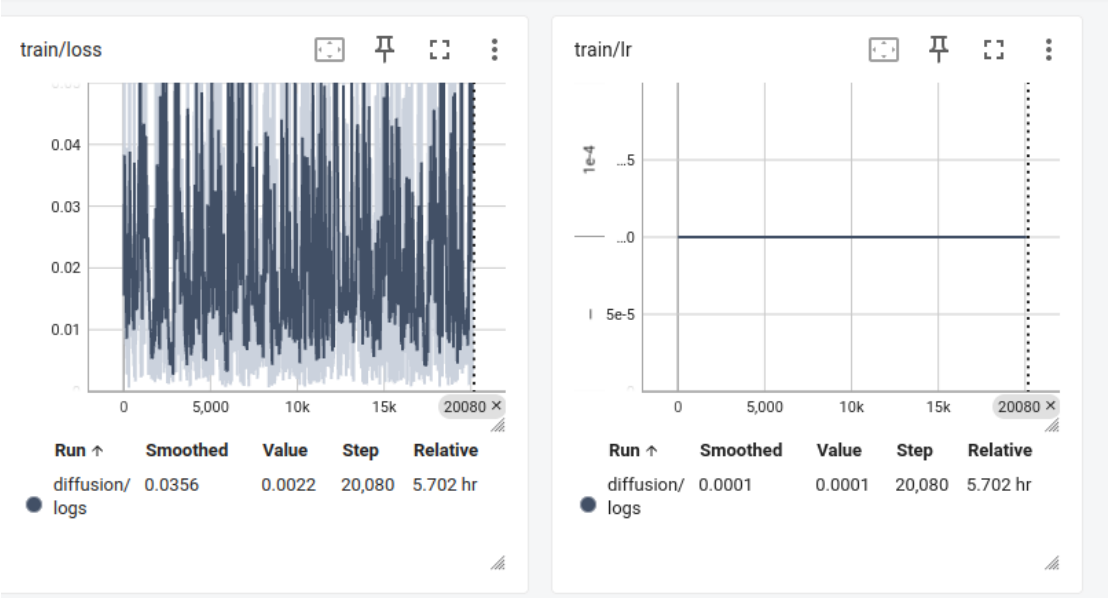


图 22 loss 和 lr

第二次，训练集里只有一种音色，训练 13 小时，每 10000step 保存一次模型，保存最新的两个模型，模型状况从一开始的不断波动，到后面趋于稳定，说明开始收敛了。梯度范数也逐渐稳定，意味着梯度在训练过程中得到了良好的控制，模型能够稳定地学习，并且有望收敛到一个较好的解。这有助于模型更好地从训练数据中提取有用的信息，提高泛化能力，并减少过拟合的风险。

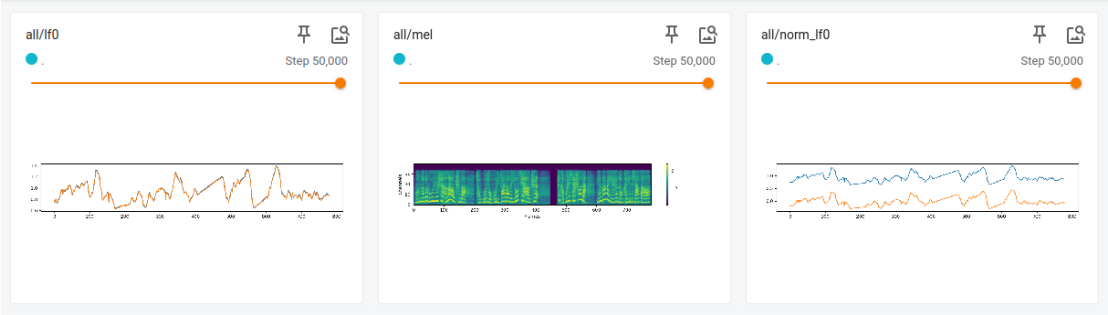


图 23 训练中的总体状况

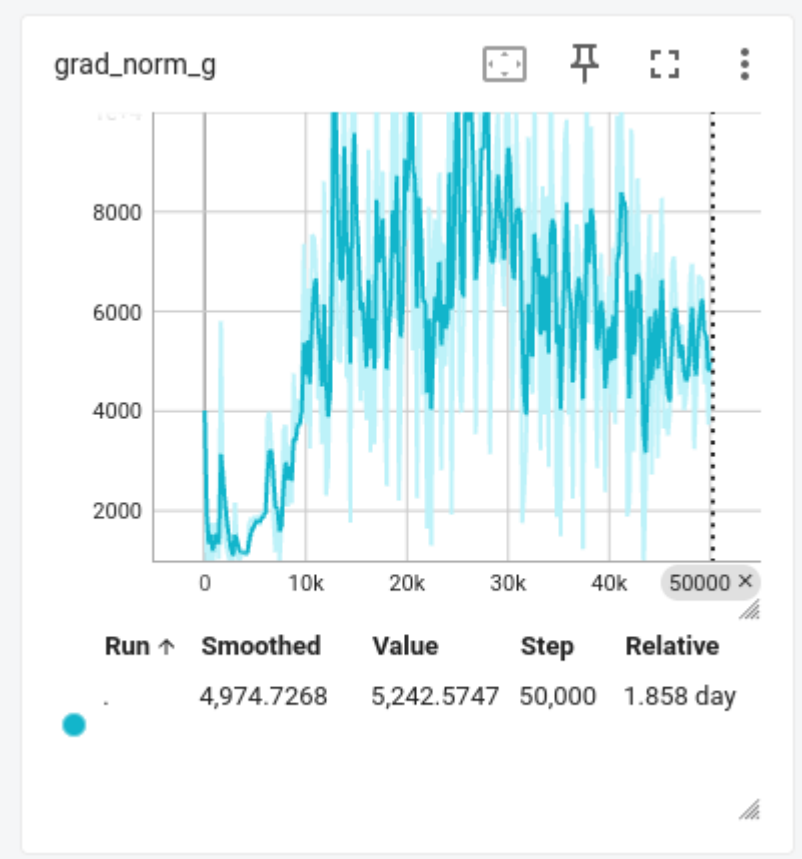


图 24 梯度范数

最后训练到如下状况停止：

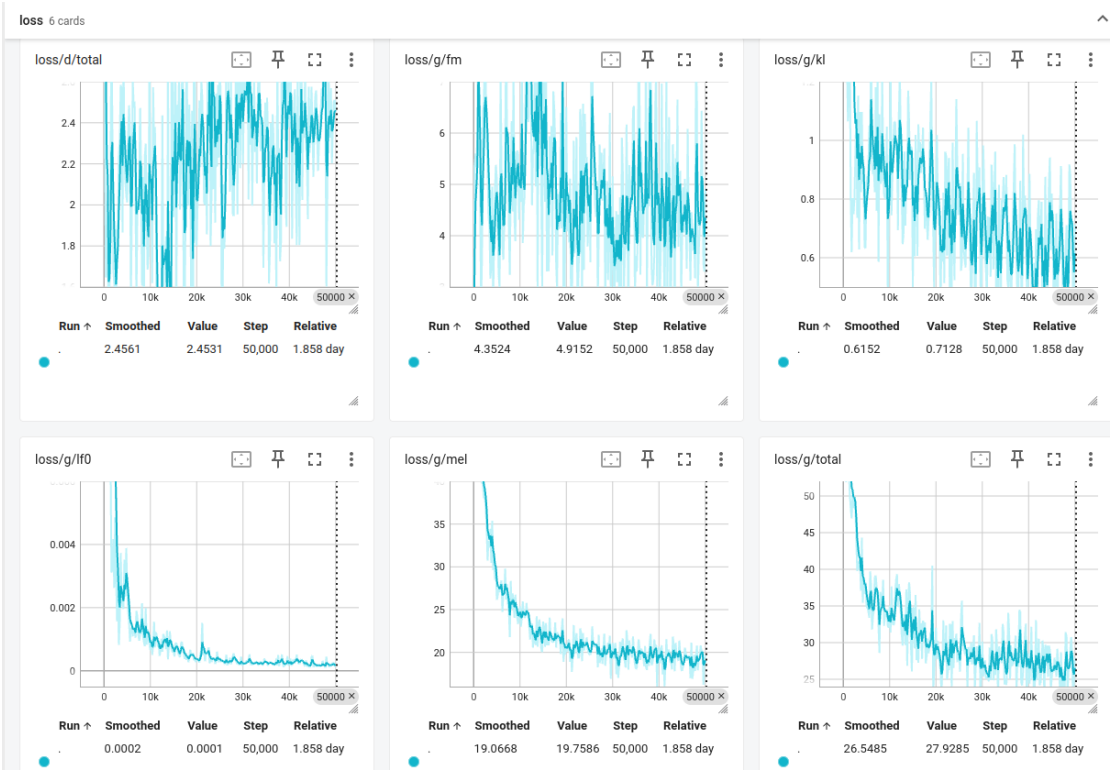


图 25 模型训练中的 Loss 各数值的变化

最终得到 40000step 和 50000step 的模型，经过测试对比，40000step 的模型偶尔可能出现电音，而 50000step 则更加地稳定，没有出现电音的情况，故选择 50000step 的模型。

第6章 作品总结

6.1 作品特色与创新点

6.1.1 高效率的桌面操控能力

相比用户使用鼠标点击 Windows 桌面翻找闹钟、打开收藏的网站链接，使用本产品只需要一句话完成，操作简便，无须记忆繁琐的操作细节，极大提升了工作效率。

6.1.2 AIGC 技术提供更流畅的用户交流

通过接入 ChatGLM3，能够使用户更自由地表达自己的意图，模型能够准确识别提供服务，并且给出更人性化的应答，全程语音交互更为流畅。

6.1.3 个性化互动新体验

语音助手以生动逼真的形象跃然桌面之上，不仅能够实时与用户进行对话交流，还允许用户根据个人喜好设定其行为模式，与之进行深度互动。这一创新设计旨在为用户打造个性化的互动体验，让每一次交流都充满趣味与便捷。

6.2 应用推广

1. 提供更多个性化部分，在使用中采集用户数据，使得更贴进用户日常使用

2. 添加更多的辅助功能，如清理桌面，换壁纸。

3. 增加桌宠形象，多变风格，对形象更细化更多样，满足用户体验感

4. 集成更多的办公工具和服务，如在线协作、会议管理、数据分析等，为用户提供一站式的办公解决方案。

5. 软件还可以与其他智能设备进行联动，实现跨平台的无缝衔接，提升用户的工作效率和便利性。

6. 更加注重用户隐私和数据安全。随着用户对隐私保护的关注度不断提升，软件将在设计和开发过程中严格遵守相关法规和规定，确保用户数据的安全性和隐私性。同时，软件还将提供用户可控的数据权限设置，让用户能够自主管理自己的数据。

7. 通过提供增值服务、广告合作或与企业合作等方式，软件可以实现商业价值的转化，为开发者带来经济收益。

6.3 未来展望

智能虚拟助理市场规模预计到 2024 年为 147.7 亿美元，预计到 2029 年将达到 608.3 亿美元，在预测期内（2024-2029 年）复合年增长率为 32.72%。深

度神经网络、机器学习和人工智能的其他进步使更多的虚拟助手成为可能。大语言模型为聊天解决问题和生成能力提供了无限可能。

大语言模型的引入将极大提升智能语音桌面助手的自然语言处理能力。大语言模型模型可以捕捉更丰富的上下文信息，生成更自然、更符合人类语言习惯的回答，这将大大提升用户的满意度和体验感。不仅使得智能语音桌面助手具备更强大的学习和进化能力。大语言模型模型可以通过不断的学习和调整参数，优化自身的性能。还可以与智能语音桌面助手的其他功能进行深度融合。其应用场景多样化、内容生成、问答系统、机器翻译、创意写作与艺术支持、语音识别、图像识别与分析，还可以应用于商业决策的数据分析等多个领域。

总的来说，需要结合大语言模型的市场，们将始终关注数据安全问题和用户需求，通过不断优化和改进软件功能，以提供高效的智能陪伴服务。

参考文献

[1] AI-Hobbyist. Genshin_Datasets: Genshin Datasets For SVC/SVS/TTS[DB/OL]. GitHub, https://github.com/AI-Hobbyist/Genshin_Datasets.

[2] 奇点.. 用户登录界面[DB/OL].CSDN 博客, https://blog.csdn.net/qq_37688204/article/details/100558278.

[3] 林林 zonzon. 用 Python 制作桌宠[DB/OL].CSDN 博客, https://blog.csdn.net/zujiasheng/article/details/124670676.

[4] Rrea. Pyqt5 聊天界面制作教程[DB/OL].博客园, https://www.cnblogs.com/Rrea/p/17584484.html.

[5] OpenAI. Whisper 模型[DB/OL].GitHub, https://github.com/openai/whisper.

[6] THUDM. ChatGLM3 模型[DB/OL].GitHub, https://github.com/THUDM/ChatGLM3.

[7] SVC-Develop-Team. So-vits-svc 模型[DB/OL].GitHub, https://github.com/svc-develop-team/so-vits-svc.