Translator：设计文档

杨一凡
520021911080
上海交通大学

日期：2023 年 4 月 26 日

1 Translator 功能设计方案：

由 Translator 需求分析文档可知，Translator 可以对输入文本、txt 文件、wav 音频文件、png 图像、pdf 文件进行翻译以及对翻译内容进行朗读，下面具体描述各个功能的设计方案。

1.1 核心设计方案：

为了使得翻译结果的准确性，Translator 使用有道智云 AI 开放平台提供的 API 接口进行文本翻译、语音翻译以及图像翻译，根据官方文档，进行 3 个 class 的编写，将其进行类的封装，方便之后实现功能时的调用，实现了最为核心的翻译功能（由于作业的性质以及时间的限制，没有选择自己进行字典的建立，而是直接使用 API 进行翻译），其翻译的核心代码如下：

1. **文本翻译类：**

```
    class YouDaoTranslator_text():
    def __init__(self):
        #进行相应API的连接
        self.headers = {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
        self.data = {
            'q': None,  #表示待翻译的文字
            'from': None,
            'to': None,
            'signType': 'v3',
            'curtime': None,
            'appKey': '3f435b1aa827effc',
            'salt': None,
            'sign': None,
            'VocabId': "您的用户词表ID"
        }
        self.url = 'https://openapi.youdao.com/api'

    def translate(self, text, Source_lang, Dest_lang):
        self.data['q'] = text
        curtime = str(int(time.time()))
        self.data['curtime'] = curtime
        salt = str(uuid.uuid1())
        self.data['salt'] = salt
        signStr = '3f435b1aa827effc' + truncate(text) + salt + curtime + '1
            qcdQSgxK7OERGkrqGqTcOUDgdwIY0FM'
```

```
        sign = encrypt_text(signStr)
        self.data['sign'] = sign
        self.data['from'] = Source_lang
        self.data['to'] = Dest_lang
        res = requests.post(self.url,headers=self.headers,data=self.data)
        return res.content
```

2. **图片翻译类：**

```
    class YouDaoTranslator_picture():
    def __init__(self):
        self.headers = {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
        self.data = {
            'from': None,
            'to': None,
            'type': '1',
            'q': None, #表示需要翻译图片的base64编码
            'appKey': '3f435b1aa827effc',
            'salt': None,
            'sign': None,
        }
        self.url = 'https://openapi.youdao.com/ocrtransapi'

    def translate(self, q, Source_lang, Dest_lang):
        self.data['q'] = q
        self.data['from'] = Source_lang
        self.data['to'] = Dest_lang
        salt = str(uuid.uuid1())
        self.data['salt'] = salt
        signStr = '3f435b1aa827effc' + str(q) + salt + '1qcdQSgxK7OERGkrqGqTcOUDgdwIY0FM'
        sign = encrypt_picture(signStr)
        self.data['sign'] = sign
        res = requests.post(self.url, headers=self.headers, data=self.data)
        return res.content
```

3. **音频翻译类：**

```
    class YouDaoTranslator_audio():
    def __init__(self):
        self.headers = {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
        self.data = {
            'from': None,
            'to': None,
            'appKey': '3f435b1aa827effc',
            'q': None,
            'salt': None,
            'sign': None,
            'signType': "v1",
            'rate': None,
            'format': 'wav',
            'channel': None,
            'type': 1,
```

```
        }
        self.url = 'https://openapi.youdao.com/speechtransapi'

    def translate(self, q, sample_rate, nchannels, Source_lang, Dest_lang ):
        self.data['q'] = q
        self.data['from'] = Source_lang
        self.data['to'] = Dest_lang
        salt = str(uuid.uuid1())
        self.data['salt'] = salt
        signStr = '3f435b1aa827effc' + q + salt + '1qcdQSgxK7OERGkrqGqTcOUDgdwIY0FM'
        sign = encrypt_picture(signStr)
        self.data['sign'] = sign
        self.data['rate'] = sample_rate
        self.data['channel'] = nchannels
        res = requests.post(self.url, headers=self.headers, data=self.data)
        return res.content
```

实现完成核心的翻译功能之后，Translator 也可对翻译之后的文本进行对应语言的朗读，其同样使用了有道智云 AI 开放平台提供的 API 接口完成，同样将其进行类的封装，其核心的文本朗读代码如下：

1. **文本朗读类：**

```
class synthesis():
    def __init__(self):
        self.headers = {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
        self.data = {
            'langType': None,
            'appKey': '3516f28ea8ff5706',
            'q': None,
            'salt': None,
            'sign': None,
        }
        self.url = 'https://openapi.youdao.com/ttsapi'

    def synthesis(self,langType,q):
        self.data['q'] = q
        self.data['langType'] = langType
        salt = str(uuid.uuid1())
        self.data['salt'] = salt
        signStr = '3516f28ea8ff5706' + q + salt + 'HJGHJFjBDPV551PPi5uMDvwEm6N9jzFD'
        sign = encrypt_picture(signStr)
        self.data['sign'] = sign
        res = requests.post(self.url, headers=self.headers, data=self.data)
        return res
```

以上的 API 接口调用代码均是在官方文档参考下独立编写完成。

1.2 具体功能设计方案：

1. **对输入文本进行翻译：**

   在左侧的文本框输入需要翻译的文本，点击**输入完成**可以进行文本上传，选择原始语言和目的语言之后，点击**翻译**，即可将翻译之后的文本显示在输出框之中，当翻译完成的文本较短时，点击显示窗口下方的**输出朗读**，即可朗读对应的翻译信息。具体信息如图 1 展示。

设计要点：

- 点击**输入完成**可以进行文本上传，选择语言后进行 **Source lang** 和 **Dest lang** 的选择，之后创造类对象，使用接口进行服务的访问，并且对返回的信息进行筛选，在文本输出框中进行翻译结果的显示。
- 点击**输出朗读**，创建类对象，对文本输出框中的内容进行朗读。

以上功能的所有代码均为独立编写。

2. **对 txt 文件进行翻译：**

首先输入 txt 文件的路径，点击**输入完成**之后，txt 文件的内容会显示在下方的 **txt 文本内容**中，并且当 txt 文本内容较短时，可点击显示框下的**输入朗读**，即可朗读 txt 文本信息，选择原始语言和目的语言之后，点击**翻译**，即可将翻译之后的文本显示在输出框之中，当翻译完成的文本较短时，点击显示窗口下方的**输出朗读**，即可朗读对应的翻译信息。具体信息如图 2 展示。

设计要点：

- 读取 txt 文件，将其中的内容展示在输入框中，核心设计要点与输入文本的翻译是相同的。
- 其中存在对输入文本的朗读，其原理与输出朗读相同。

以上功能的所有代码均为独立编写。

3. **对 wav 音频文件进行翻译：**

首先输入 wav 音频文件的路径，点击**输入完成**之后，选择原始语言和目的语言之后，点击**翻译**，即可将音频的识别内容和翻译之后的内容分别显示在**音频识别内容**以及输出框之中，其中翻译对于 wav 音频的格式和长度存在限制，wav 文件的采样率需要为 16kHZ，音频时长不可超过 15s，点击**输入朗读**或**输出朗读**，会对音频识别内容或翻译内容进行朗读。具体信息由图 3 展示。

设计要点：

- 点击**输入完成**后，进行 wav 文件路径的上传，选择语言后进行 **Source lang** 和 **Dest lang** 的选择，之后创造类对象，使用接口进行服务的访问，并且对返回的信息进行筛选，将语音识别的内容和翻译的内容显示在**音频识别内容**和**翻译之后的文本**之中。
- 可对识别内容和输出文本进行朗读，其原理同上。

4. **对 png 图片进行翻译：**

首先输入 png 图片的路径，点击**输入完成**之后，对应的 png 图片会显示在**图像内容**下方，并且会对图片进行缩放处理，使其完全进行显示，选择原始语言和目的语言之后，存在两种翻译模式，一种为普通翻译，即输出的内容为图片中提取文本的对应译文，一种为对比翻译，即输出的内容为提取文本与对应译文，默认情况下翻译模式为普通翻译，若点击图片下方的**对比翻译**，即可进行对比翻译。

设计要点：

- 点击**输入完成**后，进行 png 图片路径的上传，选择语言后进行 **Source lang** 和 **Dest lang** 的选择，之后创造类对象，使用接口进行服务的访问，并且对返回的信息进行筛选，将翻译的内容显示**翻译之后的文本**之中。
- 点击**输入完成**后，会对图片进行缩放，使其可以展示在特定区域之中，其中对于图像的操作使用了 PIL 库，其中图像处理的核心代码如下：

```
def picture_process(image_path):
    image = Image.open(image_path)  #打开指定路径的图片
    pix = image.load()
```

```
    width = image.size[0]  #获取图片的长宽像素
    print(width)
    height = image.size[1]
    print(height)
    middle = width / height
    print(middle)
    if middle > (250/210):
        resized_image_width = 250
        resized_image_height = int(250*height/width)
    else:
        resized_image_height = 210
        resized_image_width = int(210*width/height)
    resized_image = image.resize((resized_image_width, resized_image_height), )
    millis = int(round(time.time() * 1000))  #防止文件命名时的重复
    imgPath = '/Users/yangyifan/Documents/python/translate/picture/' + str(millis) +
        ".png"
    resized_image.save(imgPath)
    return imgPath
```

其本质上是通过像素的等比例缩放重新生成相应的 png 图片信息，并且将新生成的图片进行相应的展示。

- 翻译模式分为普通翻译和对比翻译，其中对比翻译会将图片的每个部分内容进行明确的标注，并且会将提取内容和翻译内容对比显示在输出框之中，其核心代码如下：

```
if result['errorCode'] == '0':
    if style == 0:
        text_output.delete("1.0", tk.END)
        for i in range(len(result['resRegions'])):
            print(result['resRegions'][i]['tranContent'])
            dataStr_Dest = ''
            for element in result['resRegions'][i]['tranContent']:
                dataStr_Dest = dataStr_Dest + element
            dataStr_Source = ''
            for element in result['resRegions'][i]['context']:
                dataStr_Source = dataStr_Source + element
            text_output.insert("insert", '第' + str(int(i + 1)) + '部分翻译结果: \n'
                )
            text_output.insert("insert", '原始内容: ' + dataStr_Source + '\n')
            text_output.insert("insert", '翻译结果: ' + dataStr_Dest + '\n')
            text_output.insert("insert", '\n')
```

图像处理代码在参考网络资料的基础上独立编写。

5. **对 pdf 文件进行翻译：**

首先输入 PDF 文件的路径，点击**输入完成**之后，对应的 PDF 文件会逐页显示在 **pdf 内容**下方，默认一个页面显示 pdf 内容的一页信息，左右滑动可以进行其他页信息的查看，选择原始语言和目的语言之后，同样存在两种翻译模式，普通翻译和对比翻译，默认情况下翻译模式为普通翻译，若点击图片下方的**对比翻译**，即可进行对比翻译，因为 PDF 文件内容普遍较多，很有可能超过朗读的最大字符数限制，因此可能出现不能朗读的情况。

设计要点：

- 点击**输入完成**后，进行 pdf 文件路径的上传，选择语言后进行 **Source lang** 和 **Dest lang** 的选择，之后创造类对象，使用接口进行服务的访问，并且对返回的信息进行筛选，将翻译的内容显示

翻译之后的文本之中。

- 点击**输入完成**后，需要将 pdf 文件逐页进行显示，首先需要将 pdf 转化为图片，使用到 pdf2image 库，之后需要将图片进行全部的展示，使用底部滑动条的方式将其进行展示，其中使用到 *picture_process* 函数，其核心代码如下：

```
images = convert_from_path(path_final)
    frame = tk.Frame(window)
    sv = tk.Scrollbar(frame, orient=HORIZONTAL)
    sv.pack(side=tk.BOTTOM, fill=tk.X)
    canvas = tk.Canvas(frame, bg='linen', width=250, height=210, relief='groove')
    canvas.pack()
    canvas.config(xscrollcommand=sv.set)
    canvas.config(scrollregion=(0, 0, 250 * len(images), 210))
    sv.config(command=canvas.xview)
    frame.place(x=50, y=215)
    canvas.config(xscrollincrement=1)
    global image_pdf
    image_file_update = []
    image_pdf = []
    global image_addr
    image_addr = []
    #将PDF进行相应的展示
    #以图片的方式在canvas上进行展示
    for i, image in enumerate(images):
        millis = int(round(time.time() * 1000))
        fname = 'image' + str(millis) + str(i) + ".png"
        image.save(fname, "PNG")
        addr = '/Users/yangyifan/Documents/python/translate/' + fname
        image_addr.append(addr)        image_file_update.append(picture_process(addr
            ))
        image_pdf.append(tk.PhotoImage(file=image_file_update[i]))
        canvas.create_image((125 + i * 250), 105, anchor='center', image=image_pdf[i
            ])
```

其想法是创建一个 frame 固定显示区域的大小，之后再进行 canvas 的构建，使得图片可以进行展示。

- 其中对比翻译在图片的基础上需要多进行一层的嵌套，其处理函数如下：

```
#将pdf转化成为图片，并进行翻译
for i,addr in enumerate(image_addr):
    addr = image_addr[i]
    f = open(addr, 'rb')
    q = base64.b64encode(f.read()).decode('utf-8')
    f.close()
    result.append(json.loads(translator.translate(q, Source_lang, Dest_lang)))
    #其同样可以分为普通翻译和对比翻译
    if style == 0:
        text_output.insert("insert", '****' + '第' + str(int(i + 1)) + '页翻译结果'
            + '****' + '\n')
        text_output.insert("insert", '\n')
        if result[i]['errorCode'] == '0':
            for j in range(len(result[i]['resRegions'])):
                dataStr_Dest = ''
                for element in result[i]['resRegions'][j]['tranContent']:
                    dataStr_Dest = dataStr_Dest + element
```

```
17                  dataStr_Source = ''
18                  for element in result[i]['resRegions'][j]['context']:
19                      dataStr_Source = dataStr_Source + element
20                  text_output.insert("insert", '第' + str(int(i + 1)) + '页   ' + '第'
                        + str(int(j + 1)) + '部分翻译结果: \n')
21                  text_output.insert("insert", '原始内容: ' + dataStr_Source + '\n')
22                  text_output.insert("insert", '翻译结果: ' + dataStr_Dest + '\n')
23                  text_output.insert("insert", '\n')
```

pdf 显示代码在参考网络资料的基础上独立编写。

项目的所有代码，均是在原始文档的学习基础上，独立进行编写。上面仅仅展示了 Translator 所使用到的核心代码，并没有展示完整代码。

2 Translator 使用指南：

2.1 Translator 运行步骤指南：

1. **初始界面：**

   组件介绍：

图 1: 初始界面

- 在左上角的输入框输入文件的地址，其使用后缀识别法：
  - 当输入内容中包含**.png**，系统识别为 png 图片文件，显示出对应的新增组件。
  - 当输入内容中包含**.txt**，系统识别为 txt 文本文件，显示出对应的新增组件。
  - 当输入内容中包含**.pdf**，系统识别为 pdf 文件，显示出对应的新增组件。
  - 当输入内容中包含**.wav**，系统识别为 wav 音频文件，显示出对应的新增组件。
  - 除此之外，系统识别为输入文本翻译，显示出对应的新增组件。
- **输入完成**按键，在输入框输入地址之后，需要点击**输入完成**，进行相应内容的上传，进行页面的跳转。
- **源语言**复选框，进行原始语言的选择，目前支持五种，分别为中文、英语、日语、韩语以及法语，默认为中文。

- **终语言**复选框，进行目的语言的选择，目前支持五种，分别为中文、英语、日语、韩语以及法语，默认为英语。
- **翻译之后的文本**，工具会将翻译后的结果输出到对应框中。
- **输出朗读**按键，点击按键之后，会对输出文本框中的内容进行对应语言的朗读，但要求文本内容不可过长。
- **重置环境**按键，点击按键之后，会使得程序全局变量恢复为默认值，并且会清空输入框和输出框。

2. **对输入文本进行翻译：**
   - **操作过程：**
     - 直接点击开始界面中的**输入完成**按键（或可输入任意字符，只要其中不包含有**.png**、**.txt**、**.pdf**、**.wav** 即可），弹出文本输入框组件，其下方的**输入完成**按键以及**翻译**按键。
     - 在输入框中输入需要翻译的原始文本，并点击输入框下方的**输入完成**按键。
     - 选择源语音（默认为中文）和终语言（默认为英文）。
     - 点击**翻译**按键，翻译的结果即可展示在输出框中。
     - 若需要进行朗读，点击**输出朗读**按键，即可朗读，但要求文本不可过长。
   - **注意事项：**
     - 选择源语言的类型应与输入文本实际的语言一致。
     - 若需使用朗读功能，对文本长度存在要求，不可过长。

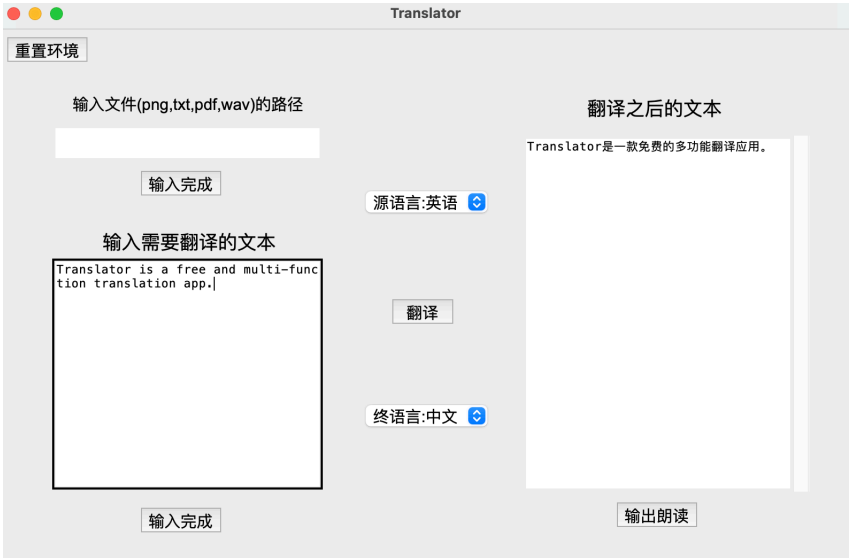

图 2: 输入文本翻译

3. **对 txt 文件进行翻译：**
   - **操作过程：**
     - 在开始界面中的输入框输入 txt 文件的地址，点击**输入完成**按键，弹出 **txt 文本内容框**并且其中展示出 txt 文件中的内容。
     - 选择源语音（默认为中文）和终语言（默认为英文）。
     - 点击**翻译**按键，翻译的结果即可展示在输出框中。
     - 若需要进行朗读，点击**输入朗读**按键或**输出朗读**按键，即可对原始内容或翻译后的内容进行朗读。
   - **注意事项：**
     - 选择源语言的类型应与 txt 文本实际的语言一致。

– 朗读功能的注意事项相同。

图 3: txt 文件翻译

4. **对 wav 音频文件进行翻译：**
   - **操作过程：**
     – 在开始界面中的输入框输入 wav 文件的地址，点击**输入完成**按键，弹出**音频识别内容**，此时，框内并没有展示出语音识别内容。
     – 选择源语音（默认为中文）和终语言（默认为英文）。
     – 点击**翻译**按键，翻译的结果和语言识别的结果分别展示在输出框以及输入框之中。
     – 若需要进行朗读，点击**输入朗读**按键或**输出朗读**按键，即可对语音识别内容或翻译后的内容进行朗读。
   - **注意事项：**
     – 所选择的音频文件为 wav 格式，并且采样频率为 16KHz，音频的长度不可超过 15s。
     – 朗读功能的注意事项相同。

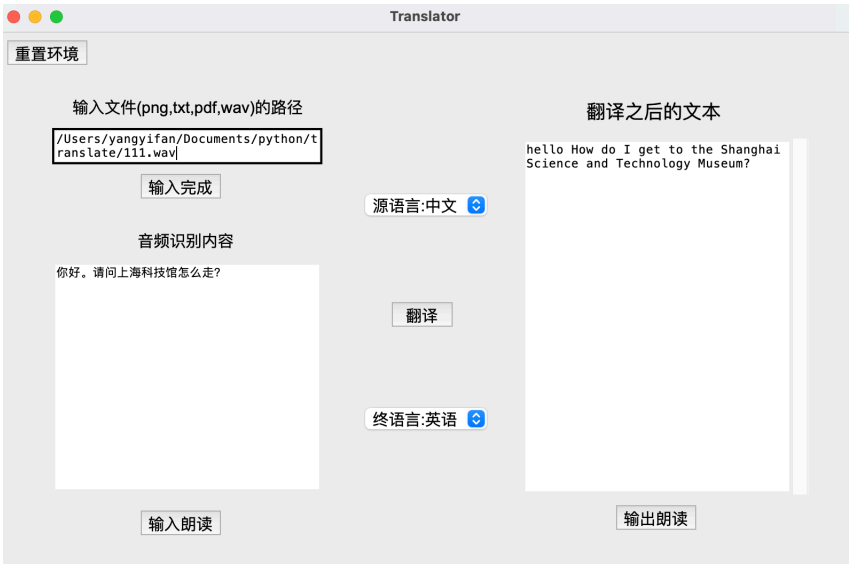

图 4: wav 音频文件翻译

5. **对 png 图片进行翻译：**
   - **操作过程：**
     - 在开始界面中的输入框输入 png 文件的地址，点击**输入完成**按键，弹出**图像内容**，会将识别出的 png 图像缩放后展示在对应区域。
     - 选择源语音（默认为中文）和终语言（默认为英文）。
     - 选择翻译模式，默认为普通翻译，若需要进行对比翻译，点击**对比翻译**按键，之后点击**翻译**按键，翻译的结果展示在输出框中。
     - 若需要进行朗读，点击**输出朗读**按键，即可对翻译后的内容进行朗读。
   - **注意事项：**
     - 进行图片缩放展示时，实际上时创建新的图片，并且将新的图片存储在源文件中的 picture 文件夹中，因此在进行检测时，需要确保源文件中存在 picture 文件夹。
     - 对比翻译模式下，不可进行朗读。
     - 当原始语言不为中文时，终语言只可为中文，不可翻译成其他语言。

图 5: png 文件普通翻译

图 6: png 文件对比翻译

6. **对 pdf 文件进行翻译：**
   - **操作过程：**
     - 在开始界面中的输入框输入 pdf 文件的地址，点击**输入完成**按键，弹出 **pdf 内容**，会将识别出的 pdf 内容缩放后逐页展示在对应区域。
     - 选择源语音（默认为中文）和终语言（默认为英文）。
     - 选择翻译模式，默认为普通翻译，若需要进行对比翻译，点击**对比翻译**按键，之后点击**翻译**按键，翻译的结果展示在输出框中。
     - 若需要进行朗读，点击**输出朗读**按键，即可对翻译后的内容进行朗读。
   - **注意事项：**
     - 对 pdf 进行翻译时，首先将 pdf 文件转化为图片，图片存放在源文件目录之中，并且对转化后的图片进行缩放处理，同样存储在源文件中的 picture 文件夹中，因此在进行检测时，需要确保源文件中存在 picture 文件夹。
     - 本质上为图片翻译的扩展，注意事项基本相同，并且一般因文本过长而无法朗读。

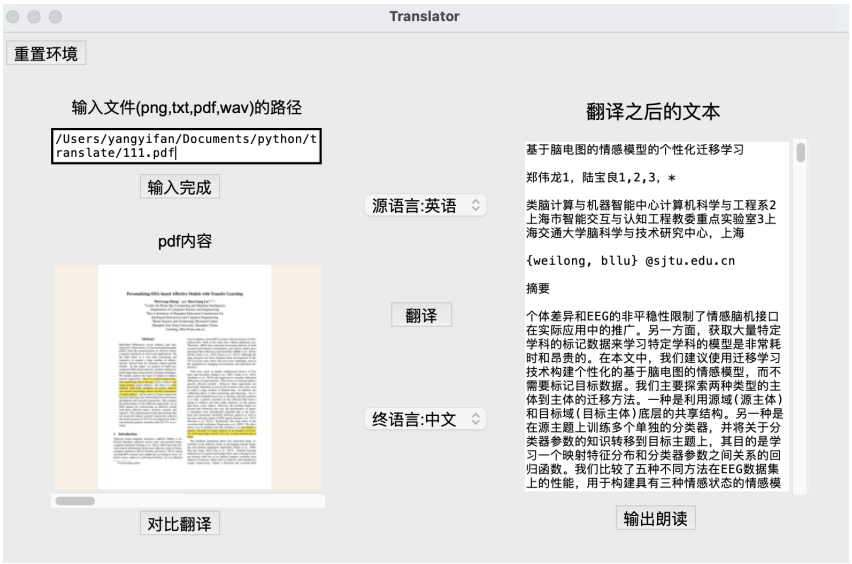

图 7: pdf 文件普通翻译

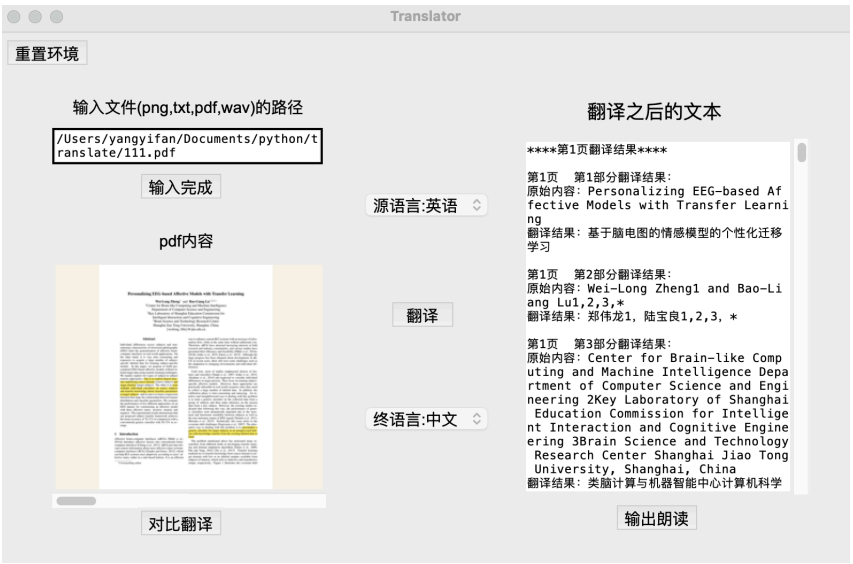

图 8: pdf 文件对比翻译

2.2 Translator 使用注意事项：

1. 在源文件目录下需要建立相应的文件夹，以保存程序过程中新产生的文件。
   - 建立 audio 文件夹。朗读时，首先需要生成对应的音频文件，之后在进行音频文件的播放，音频文件存储在 audio 文件夹中。
   - 建立 picture 文件夹。进行 pdf 文件以及 png 文件的展示时，均需要重新生成图片文件，新生成的图片文件存储在 picture 文件夹中。
2. 在源文件之中，存在 111.txt，111.png，111.wav，111.pdf 文件，可以进行工具功能的测试。
3. 在进行不同功能的检测时，测试时未发现任何 bug，但是为防止未知错误的出现，建议在切换功能时，点击**重置环境**按键进行环境重置。

2.3 Translator 测试环境：

1. 使用 python 版本为 python3.10.7
2. 均在 macos 系统下进行调试，在 windows 系统下可能会存在差异，因此尽量选择 macos 系统进行检测
3. 使用到的库有：

```
#使用到的库文件
import hashlib
import json
import time
import tkinter.messagebox
import requests
import sys
import uuid
import base64
import os
import wave
from importlib import reload
#相应的声音播放库文件
from playsound import playsound
#相应的图片处理库文件
from PIL import Image
#相应的pdf处理文件
from pdf2image import convert_from_path
from pdf2image.exceptions import (
 PDFInfoNotInstalledError,
 PDFPageCountError,
 PDFSyntaxError
)
#相应的GUI界面设计库
import tkinter as tk
from tkinter.constants import (HORIZONTAL,VERTICAL,RIGHT,LEFT,X,Y,BOTH,BOTTOM,YES,NONE,
    END,CURRENT)
import time
reload(sys)
```

其中 PIL 库，pdf2image 库，requests，playsound 需要安装。

- pip install requests
- pip install pdf2image
- pip install playsound

- pip install Pillow