

学号 2001818

**语言分析和机器翻译2020**

**多语言机器翻译系统**

**设计与实现**

学 院 名 称：计算机科学与工程学院

专 业 名 称：计算机科学与技术

学 生 姓 名：王老虎

指 导 教 师：肖 桐 教授

二○二〇年十二月

# 

# 1 主要工作

（1）Web系统的搭建。

（2）语种与数据集的确定。

（3）数据集的处理。

（4）翻译系统搭建和接口封装。

# 2 主要技术

## 2.1 Transformer

### 2.1.1 网络结构

Transformer是Geogle团队在2017年提出的自然语言处理领域的经典之作，通过注意力机制直接获取全局信息，不像RNN需要逐步提取信息，也不像CNN只能获取局部信息。它在序列到序列的机器翻译任务上的表现超过了RNN和CNN，使用了编码器-解码器和注意力机制并且达到很好的翻译效果，可以高效并行化。

在解决序列到序列的任务时，Transfomer的输入是一段序列，输出是另一段序列，如图2.1所示。

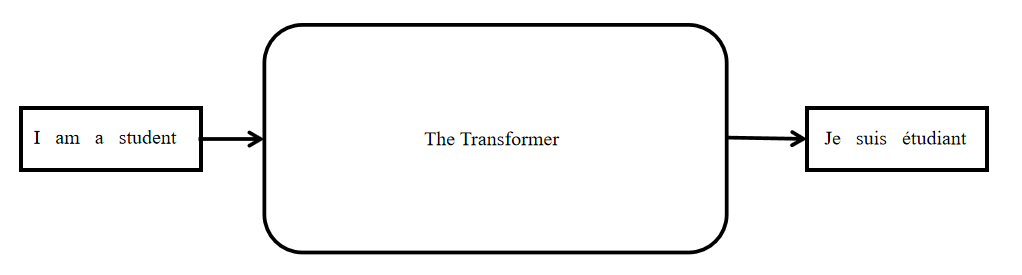


图2.1 Transformer简略图

由于Transformer模型采用了编码器-解码器机制，因此上图可进一步拆解，将其中的The Transformer拆分为两部分，一部分是编码器部分Encoders，一部分是解码器部分Decoders，如图2.2所示。其中的编码器部分和解码器部分将在下面的分析中进行拆解。

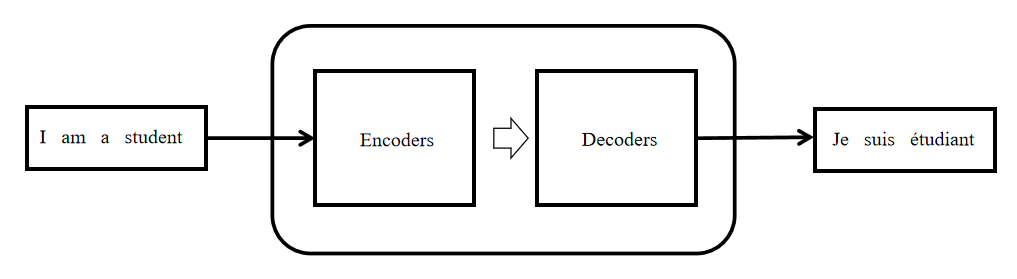


图2.2 Transformer 拆分图

在Transformer中，编码器部分由6个完全相同的编码器子层组成，解码器部分同样由6个完全相同的解码器子层组成。实际应用中，编码器和解码器的个数是需要根据具体实验得来的，但是不仅仅局限于上述所提到的6个。在序列到序列模型中，编码器负责把源语言编码成一种句子表示形式（如向量），解码器负责利用这种表示逐词生成处理目标语句子。具体而言，编码器把输入网络的长度不等的句子进行词嵌入Embedding，变成一个定长的背景向量，并在该背景向量中编码输入位置序列信息，在编码器经过特征抽取变换之后得到隐藏状态，解码器对隐藏状态进行信息提取，并一步步生成最终的单词。将图2.2中编码器部分和解码器部分拆分成如图2.3所示。该图具体表现出了Transformer的模型结构特点。

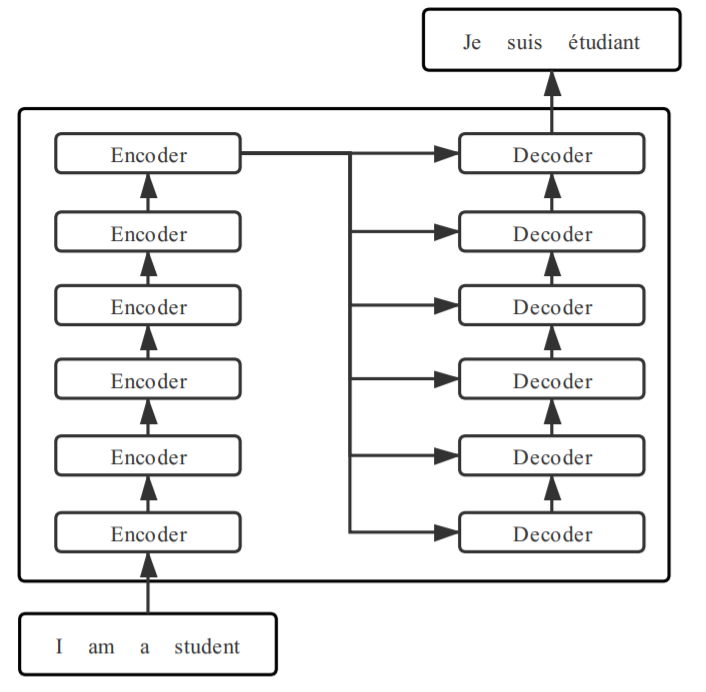


图2.3 Transformer细分图

而在编码器部分中的6个编码器的基本结构都是相同的，但是它们并不共享权重。每一个编码器都被分成两个子层，自底向上第一层是自注意力（Self-Attention）层用于提高某个单词对其他单词的关注度，第二层是前馈神经网络层（Feed Forward Network）用以进行维度变换，而每个子层传递完又会进行残差（Add）和层正则化（LayerNorm）的操作，如图2.4所示。而这些层之中所包含的运行机制以及数学原理，将会在本节的后续内容中进行说明。

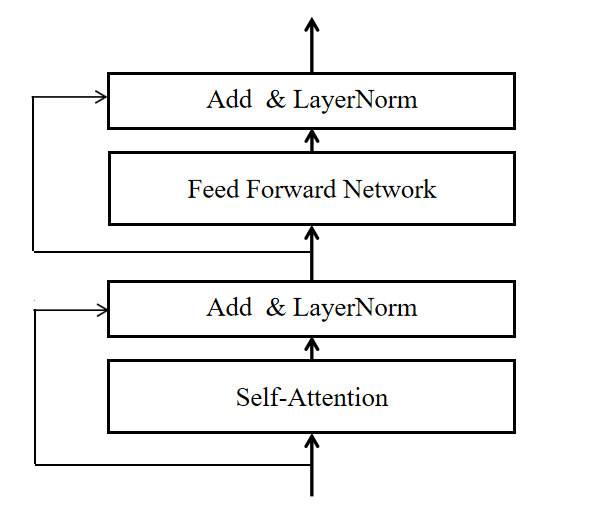


图2.4 编码器细分图

解码组件在基本结构上也是相同的，每一个组件都被分成三个子层，自底向上分别是自注意力层、编码器-解码器注意力层（Encoder-Decoder Attention）和前馈神经网络层，每个子层之上也存在残差和层正则化操作，如图2.5所示。

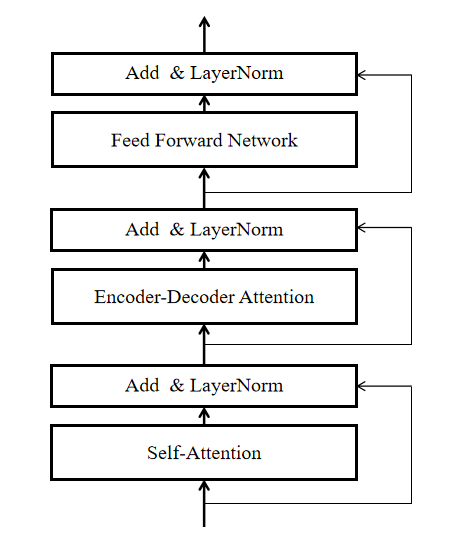


图2.5 解码器细分图

将编码器和解码器连接起来，便形成Transformer整体图，如图2.6所示。需要注意的是，图2.6中的Transformer整体图，只画出了编码器部分最顶端的编码器和解码器部分最顶端的解码器之间连接。下面将对其中的各部分机制运作进行简要介绍。

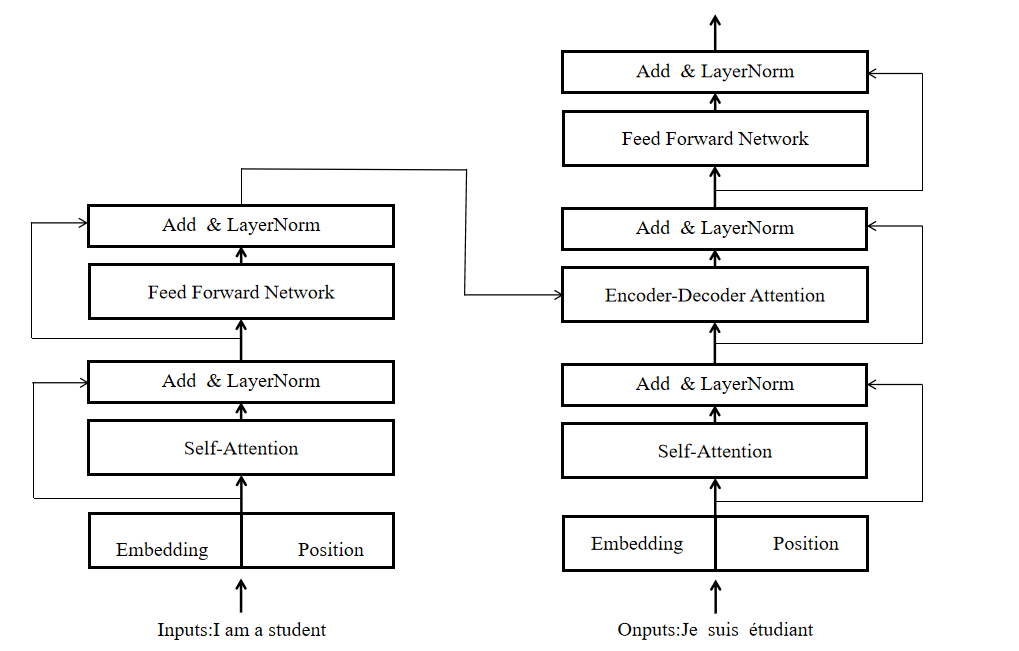


图2.6 Transformer整体图

### 2.1.2 主要机制

#### 2.1.2.1 位置编码

在循环神经网络中，注意力机制忽略了单词之间的顺序关系，即一句话中，相同词语在不同的位置时模型无法区分，为了解决这个问题，在Transformer中引入了位置编码，它使用正余弦函数来进行编码。见公式2.1和公式2.2，式中PE表示位置编码，pos代表第几个词，i代表词嵌入中的第几维，表示词向量的维度。位置编码将加到原有的词向量中去，即构造一个跟输入Embedding维度一样的矩阵，然后跟输入矩阵相加得到自注意力层的输入。

(2.1)

(2.2)

#### 2.1.2.2 自注意力机制

此机制主要作用在编码器和解码器中的自注意力层以及解码器中的Encoder-Decoder注意力层。Transformer使用点乘的自注意力机制来捕获句子内部各个位置之间的相似性。

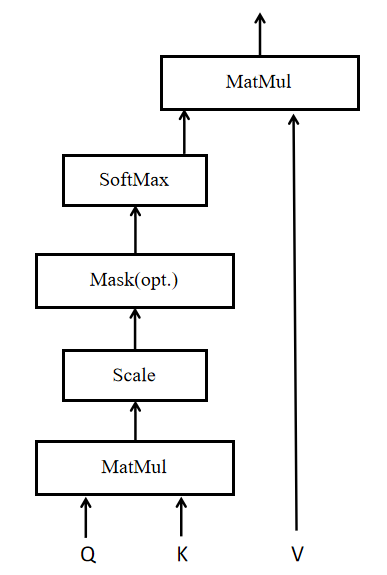


图2.7 基于点乘的注意力机制

如图2.7所示，图中的Q、K、V均来自同一句子，Q和K的转置进行点积（MatMul）得到句子内部各个位置的相关性，相关性矩阵在训练中反差变大，不利于训练，所以对其进行缩放（Scale）。Mask表示掩码，是对某些值进行取负无穷的操作，使其对某些信息进行屏蔽。在Transformer中存在两种Mask，对于源语和目标语的输入，由于需要进行batch处理，有些部分是填充的，需要Mask进行屏蔽。对于解码器来说，由于在预测的时候需要自左向右进行，为了保持训练解码一致，需要对未来信息进行屏蔽。见公式2.3。

(2.3)

在Transformer中首次提出了多头注意力机制，将Q、K、V沿着隐层维度切分成h个子集，分别进行注意力操作，它可以扩展模型关注不同位置的能力同时它提供了多个“表示子空间”，以取得更好的效果。

#### 2.1.2.3 残差&层正则化

在Transformer中，编码器部分、解码器部分分别由6层相同的编码器、解码器构成，每层的编码器或者解码器又包含多个子层。因此，Transformer深层神经网路的范畴，在训练过程中很容易出现梯度消失的情况，因此引入了残差网络在避免上述问题。如图2.8所示。

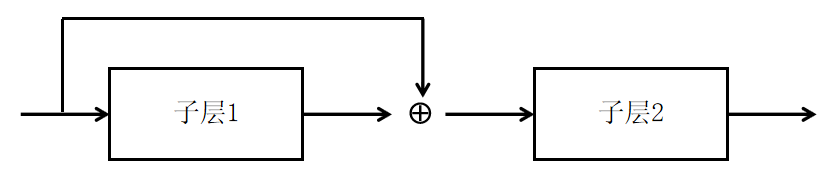


图2.8 残差原理图

其中，子层1的输入为，输出为，那么子层2的输入如公式2.4所示。

(2.4)

## 2.2 Flask框架

Flask是较为流行的Web“微”框架，其使用Python语言编写，简单、灵活、轻便、易上手，可协助开发者快速构建Web系统，它主要包括了Werkzeug和Jinja2两个核心函数库，其中前者负责业务处理，后者负责安全性能。同时，它可以和MVC模式完美结合协助开发人员分工合作。

Flask在程序中将功能函数分配给某一个具体的地址定位符，当用户访问该地址定位符时，系统会执行绑定的函数，交互图如图2.12所示。

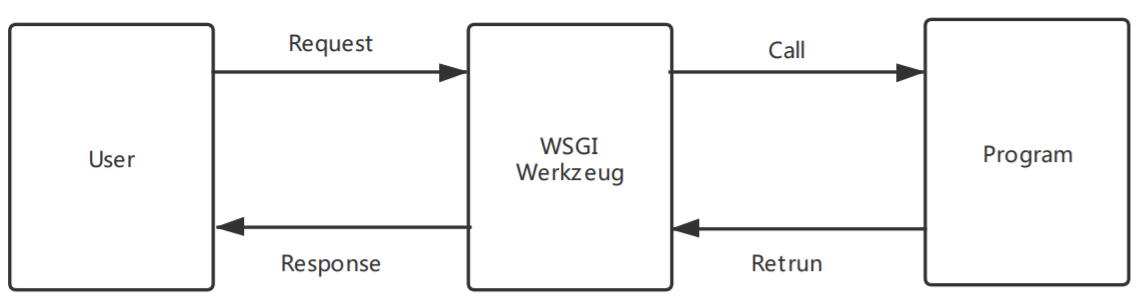


图2.9 Flask框架工作过程

WSGI是Web服务器和Web应用程序间通用接口的规范。Werkzeug是一个WSGI工具包，它实现了请求，响应对象和实用函数。 这使得能够在其上构建web框架。 Flask框架使用Werkzeug作为其基础之一，并使用Jinja2模板引擎。

为了降低系统的耦合度，本课题采用了前后端分离的方式，前端的HTML页面负责接收用户的输入，后端的代码将以HTPP GET方式接收用户发送过来的表单数据，表单数据中最主要的数据是用户在前端HTML页面所输入的待翻译的句子，并调用接口，返回结果。如图2.13所示，是对系统模块相互合作关系描述的泳道图。

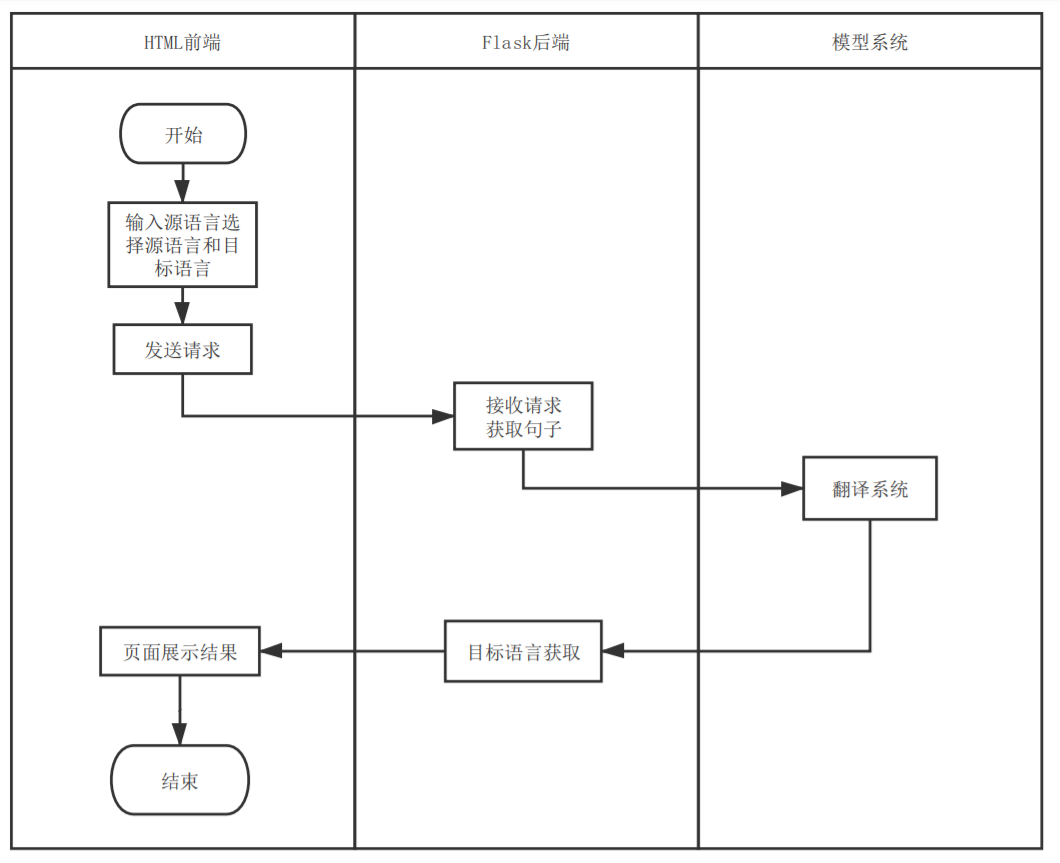


图2.10 系统活动泳道图

# 3 主要工作展开

## 3.1 Web系统搭建

Web系统的开发内容主要包括：前端交互式页面的开发、后端服务器开发。Web系统的开发主要在Windows系统的PyCharm集成开发环境中，使用Flask框架完成轻量级Web系统搭建。

### 3.1.1 前端页面

根据需求分析的结果，本系统需要做出可供用户使用的交互式Web系统。为此，本课题设计了一套较为简单且易于理解的部署策略。其中，前端页面包含输入文本框和两个下拉文本框用于获取用户输入的待翻译的句子以及源语言和目标语言，通过表单发送给后台服务器。考虑到界面的简洁性和美观性，以及用户在看到网页界面的时候，无需任何学习成本即可上手进行系统的使用。因此，综合参考小牛翻译、百度翻译、谷歌翻译等知名翻译网页端开发的样式，初步涉及本次课题的网页前端页面布局如下图4.2所示。

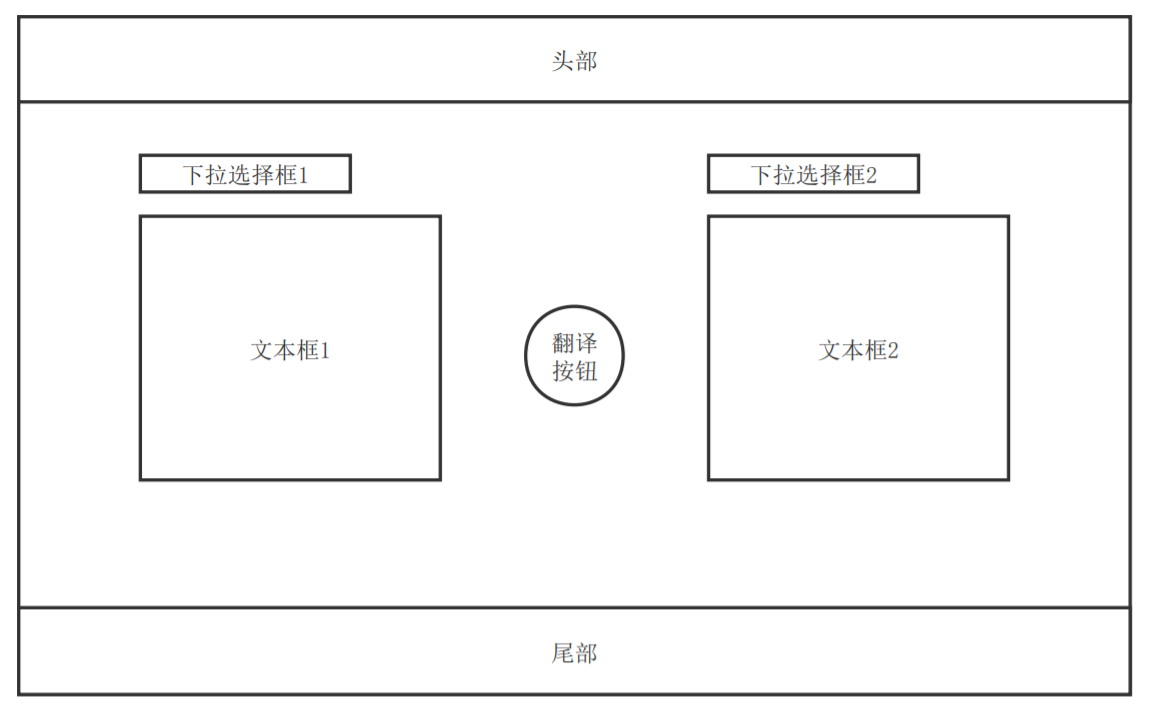


图3.1 Web前端页面布局设计图

### 3.1.2 后端服务器

后台服务器是采用Flask框架实现的Web后端系统，以GET/POST方式获取待翻译句子以及源语言和目标语言，之后进行相应特殊情况的判断和处理，调用翻译接口，完成翻译任务，并将结果返还给HTML页面，展示给用户。用户在看到翻译结果后，可以在此页面重新输入翻译结果并且重新选择源语言和目标语言进行翻译。可以看出，Web系统的核心在于Web后端服务区，它的任务总结如下：

1. 与Web前端进行通信，以HTTP GET/POST方式获取Web界面用户输入的数据并且返回结果数据。
2. 启动模型文件，调用翻译接口。这部分内容的重点在于模型如何加载以及翻译接口如何封装。此部分内容属于翻译系统设计。
3. 特殊情况考虑，即用户没用选择源语言和目标语言、源语言和目标语言相同、用户没有输入待翻译的句子这三种情况。

因此，根据上述（1）（2）可以得出Web系统后端的信息流动情况如下图4.3所示。

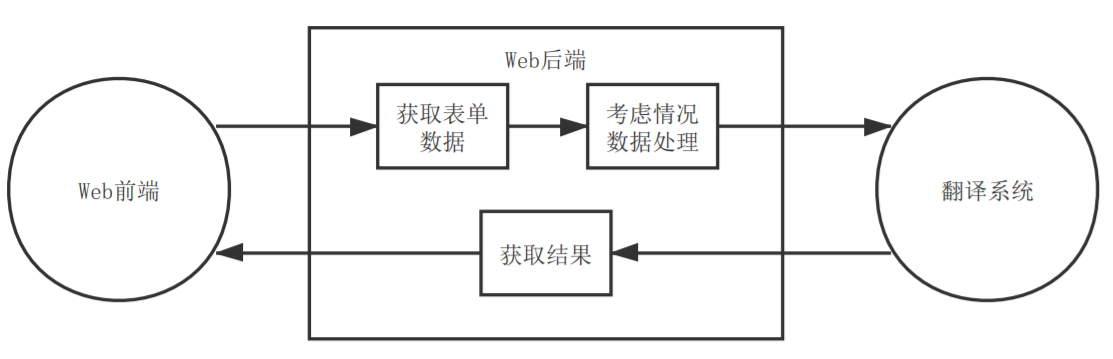


图3.2 Web后端信息流动图

## 3.2 翻译系统

翻译系统的开发设计工作主要包括：数据处理、调参训练、解码翻译、效果测试。将分为三部分内容介绍翻译系统设计过程，一是数据处理，二是模型训练、解码、测试，三是系统性能提升。

### 3.2.1 数据处理

本次课题搭建的多语言多领域机器翻译系统的核心就在于数据处理。数据处理是从事自然语言处理任务研究人员所必备的技能，是搭建效果良好的翻译系统的基石。它需要整理出高质量的“纯净的”双语平行语料库，用于在下一步训练和解码时使用。不同比赛所提供的数据集的特点不同，故需要先观察数据集的特点，之后在根据数据集的特点采用不同策略。

#### 3.2.1.1 WMT处理方式

来自WMT14德英和德法的训练集、来自WMT06shared-task的验证集和测试集以及来自WMT20的数据字典是质量较高的平行语料库，通过对数据的观察，发现针对WMT的数据集所要进行的主要处理步骤如下：首先用Python脚本将文本中的中文标点符号替换成英文标点符号，之后用Moses开源工具进行分词，最后用Subword-nmt对分词后的数据进行BPE分词处理。针对WMT数据的处理流程图如图4.4所示。

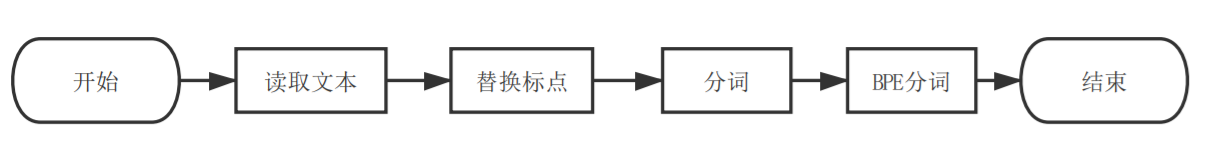


图3.3 WMT、验证集和测试集数据处理流程图

#### 3.2.1.2 IWSLT数据处理方式

来自IWSLT17德英的数据集来源于TED比赛的数据，官网所提供的数据是带有HTML标签的数据，通过观察，最终确定对来自IWSLT的数据所要进行的处理步骤描述如下：

处理（1）使用Linux shell脚本循环读取每一行文本，若该行含有指定标签则舍弃，否则去掉指定标签后保留该行于新文本中。

处理（2）读取行文本，将中文标点替换成英文标点后，先使用Moses开源工具进行分词，之后使用subword-nmt对分词后的数据进行切分子词。

针对IWSLT数据的处理流程图如图4.5所示。

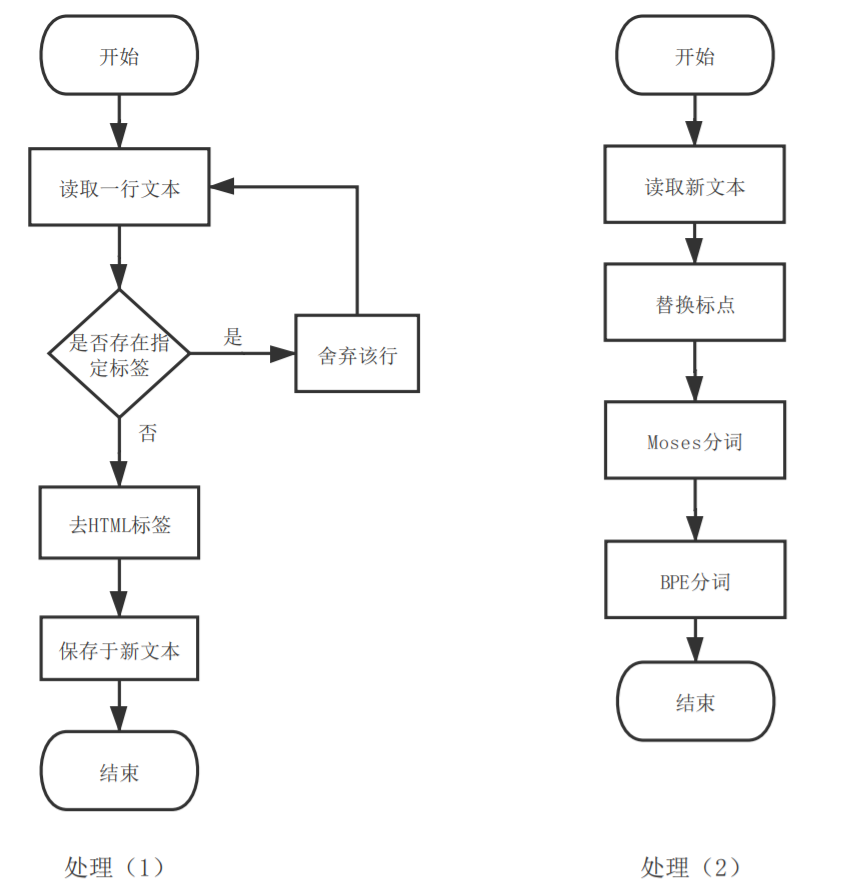


图3.4 IWSLT数据处理流程图

#### 3.2.1.3 多语言处理

因为要利用单一网络模型实现不同的源语言和目标语言的相互翻译工作，因此便涉及到关于网络模型如何辨别源语言要翻译到何种目标语言的问题。本课题的做法是在数据处理阶段于双语语料的源语言句子开头加入标签，该标签表明了源语言所要翻译到的目标语言。网络模型通过学习，便能获得对加入目标语言标签的句子翻译到所要求语言的能力。

假如有英语到法语的句子对：I am a student -> Je suis étudiant

则源语言句子将会被修改为：2fr I am a student -> Je suis étudiant

其中2fr就是表明所要翻译到的目标语言的标签。该标签帮助网络模型学习到了如何翻译到期望的目标语言。

### 3.2.2 模型训练、解码与测试

Fairseq提供了用于模型训练和模型评估的命令行工具，主要包含以下六个方法。

1. fairseq-preprocess:进行Fairseq数据预处理，用于构建词汇表并对训练数据二进制化。
2. fairseq-train:在单个或者多个GPU上训练模型。
3. fairseq-generate:用训练好的模型翻译预处理好的数据。
4. fairseq-interactive:用训练好的模型翻译原始文本。
5. fairseq-score:生成翻译与参考翻译的BLEU评分。
6. Fairseq-eval-lm:语言模型评估。

本课题进行模型训练和解码将使用Fairseq中的fairseq-preprocess、fairseq-train、fairseq-generate、fairseq-interactive脚本。其中，fairseq-preprocess、fairseq-trainy以及fairseq-generate用于模型的数据二进制、模型训练和解码，即系统的搭建部分，如图4.6所示为翻译系统搭建流程图。fairseq中的fairseq-interactive用于翻译系统的整合。

至于在系统的整合方面，本课题便是初步计划深入fairseq中interactive.py代码，将其拆分封装为加载模型和翻译两个接口。

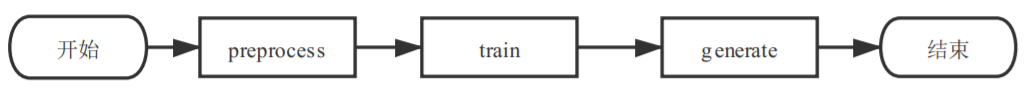


图3.5 翻译系统搭建流程图

## 3.5 系统整合设计

如图4.7所示为设计的初步系统工作流程图。

系统的整合工作涉及到Web系统前后端的整合、Web系统与翻译系统的整合（即翻译接口的包装）以及特殊情况考虑，下面将对这三个情况进行详细且全面的解释说明。

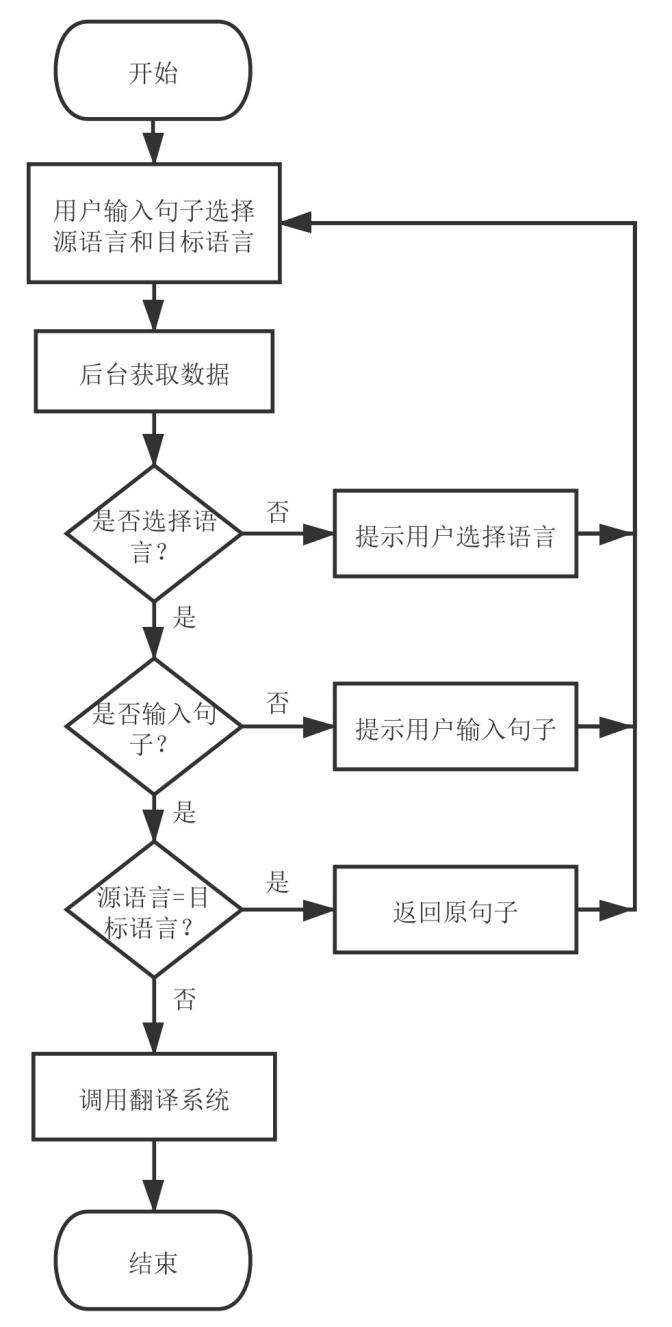


图3.6 系统整体工作流程图

1. Web前后端整合。

Web系统前端用户点击翻译按钮提交选择好的源语言和目标语言和待翻译的句子，Web系统后端以Get/Post方式获取表单内容，并且调用翻译接口获取翻译结果，并将翻译结果返还给前端。该工作将使用了Jinja2模板引擎以及Flask库中的render\_template()函数。

1. 翻译接口包装。

初步设计将Fairseq中交互式解码翻译的代码文件interactive.py拆分成加载模型和翻译两部分，加载模型实现在网页服务器启动的过程中加载训练完成的模型文件，翻译是封装好的翻译接口函数用以接收带翻译句子并返回目标结果句子。方便Web系统在启动时加载模型，在翻译时调用翻译接口。

1. 针对三种特殊情况所采取的处理方式。

如用户没用选择源语言和目标语言则提示用户在下拉框中选择语言，用户没有输入待翻译的句子则提示在文本框中输入句子，源语言和目标语言相同则返回原句子。

# 4 具体实现

## 4.1 训练环境

表4**.1** 训练集统计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 来源 | 语种 | 数据量 | 大小 | 领域 |
| WMT14 | de-en | 1920.2k | 188Mb | 新闻 |
| IWSLT17 | de-en | 209.5k | 42.7Mb | 口语 |
| WMT14 | fr-en | 2007.7k | 193Mb | 新闻 |
| 字典数据 | de-en | 15.9k | 300k | - |
| 字典数据 | fr-en | 15.2k | 280k | - |

系统训练数据量如表5.1所示。从图中可以推断出，系统在训练过程中将会占用大量的内存计算资源，笔记本在硬盘方面可以满足容量，但是在内存容量上无法满足训练的要求。同时，性能优越的显卡可以充分加速训练过程提高训练效果。因此，本系统的训练是在Linux服务器上完成，12GB可用显存充分加速训练，操作系统环境是Linux 64位 CentOS 7.6。本课题翻译系统的训练硬件配置如表5.2所示。

表4**.2** 系统训练环境硬件配置表

|  |  |
| --- | --- |
| 硬件资源 | 配置信息 |
| CPU | 6核心 |
| 显存 | 12GB |
| 硬盘空间 | 50GB |
| GPU | GeForce GTX TITAN X |
| CPU最高主频 | 3600MHz |
| CPU最低主频 | 1200Mhz |

## 4.2 开发环境

多语言多领域机器翻译系统的开发主要是在Windows10 家庭版和Linux CentOS7.6操作系统上进行。软件配置及功能信息详情如表5.3所示。

表4.3 系统开发环境软件配置表

|  |  |  |
| --- | --- | --- |
| 名称 | 版本 | 功能 |
| Windows | 10 | 操作系统 |
| Linux | CentOS 7.6 | 操作系统 |
| PyCharrm | 2020.01 x64 | 集成开发环境 |
| VsCode | 1.45 | 跨平台编辑器 |
| Python | 3.7 | 编程环境 |
| Flask | 1.1.2 | Python Web框架 |
| Jinja2 | 2.11.2 | Flask 模板引擎 |
| Pytorch-GPU | 1.3.0 | Python 机器学习库 |
| Mosesdecoder | v 4.0 | 语言分词工具 |

Web系统前后端的开发以及翻译系统加载模型和翻译两个接口的封装主要使用Windows上的集成开发环境PyCharm 2019社区版，过程中以VScode作为辅助编辑器，使用的Web框架是Python Web Flask框架。

## 4.3 Web搭建

由4.3节Web系统设计部分可知，Web系统设计主要包含Web前端页面和Web后端服务器以及前后端通信的具体过程。下面将从这两方面分别介绍其具体的实现过程。

### 4.3.1 前端具体实现

Web系统前端主要采用HTML5 + CSS的主要组合，辅以少量JS实现动态交互。在HTML5中，页面所展现出的主要组件有<img>图片、<select>下拉列表、<textarea>文本输入框、<input>按钮。页面布局采用<div>画布进行整体规划分为头部、主题、底部三个部分。之后采取div嵌套的方式，规划主体中画布的具体分布情况，在将页面主要组件放入画布中并最终完成了页面搭建。

如图5.1所示，是Web系统前端的实现效果图。本效果图充分体现了系统设计阶段所要遵循的交互界面友好原则，使用者无需任何学习成本即可上手使用翻译系统。



图4.1 Web页面实现效果图

### 4.3.2 后端具体实现

Flask框架的基本模式是在后端程序里将一个函数分配给一个URL，每当用户访问这个URL时，系统就会执行分配给该URL的函数，获取函数的返回值并将其显示到浏览器上。为充分降低系统耦合性，实现前后端分离，方便后续系统维护，在Web后端服务器的实现中便充分利用到了该模式。具体表现为后端服务器设计并实现了两个关键函数，translator()和result()。下面将具体说明两个函数的实现。

1. translator()。该函数的作用是接收表单数据，具体而言，利用HTTP GET方式获取前端页面用户选择的源语言、目标语言以及输入的句子，此处使用的是flask库中的request库，关键函数是request.args.get()。与translator()函数绑定的url为127.0.0.1/translator，访问此url，系统便问调用translator()函数，从前端页面以GET方式获得表单数据。最后该函数会使用url\_for()动态构建函数result()的url。下面介绍result()函数。
2. result()。该函数是返回翻译结果的函数。该函数首先接收translator()函数传递的用户选择的源语言src\_lan、目标语言tgt\_lan以及输入的待翻译句子sentence。之后根据系统设计阶段考虑的用户操作的三种特殊情况，即用户是否选择语言、是否输入句子、源语言和目标语言相同构建翻译结果result，在Web页面右侧文本框对用户进行相应的提示。

因为训练采用的数据集是德英和德法的数据，因此系统通过训练便可以学习到在这两种语言对之间相互翻译的能力，但是本系统需要完成三种语言的互译任务。为了实现德法之间的相互翻译问题，则首先在原句开头加入2en标签翻译到英语，然后在翻译好的英语句子开头加入目标语标签翻译到目标。

result()函数具体工作流程图如图5.2所示。可以看出来，具体解决系统设计阶段考虑的特殊情况问题时，还可能会出现其他情况。

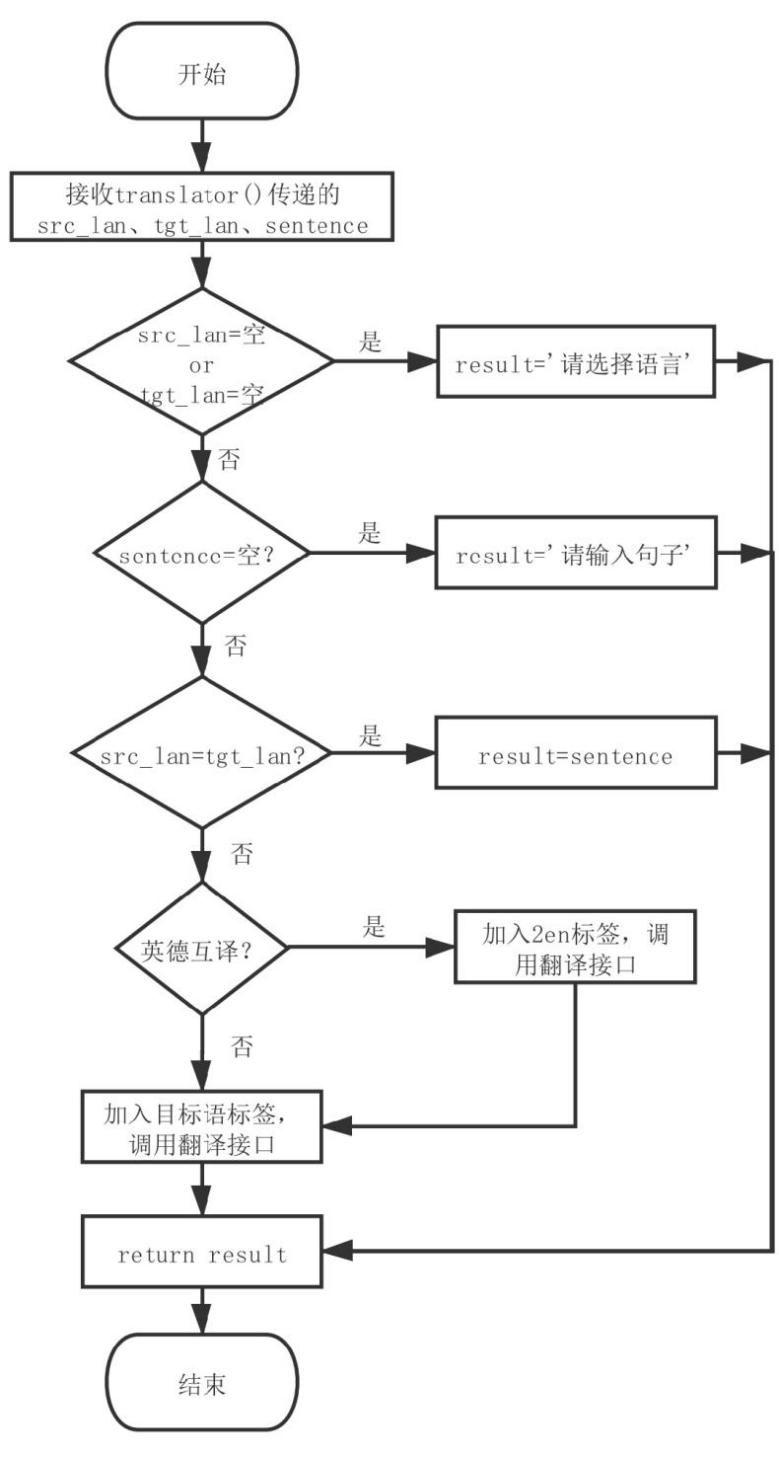


图4.2 result()函数程序流程图

## 4.4 翻译系统搭建

### 4.4.1 平行语料库的构建

根据系统设计时对数据的处理分析可知，对来自WMT或是来自IWSLT的数据处理的流程有所不同。不管是来自WMT14数据集的平行语料库还是来自IWSLT数据集的数据，都需要经过标点替换、分词、加目标语标签、BPE分词三个步骤。来自IWSLT的数据需要经过去HTML标签的处理。之后需要对处理完成的数据合并，通过subword-nmt中的learn\_bpe.py学习得到子词表，之后利用子词表将分词之后的数据切分子词。

此时涉及到关于何时进行加标签的处理，考虑到BPE分词需要对训练数据进行学习得到子词表，因此在Moses分词后进行加标签的处理。此时，数据集所要进行的共同处理由原来的分词和BPE分词变为分词、加目标语标签以及BPE分词，如图5.3所示。

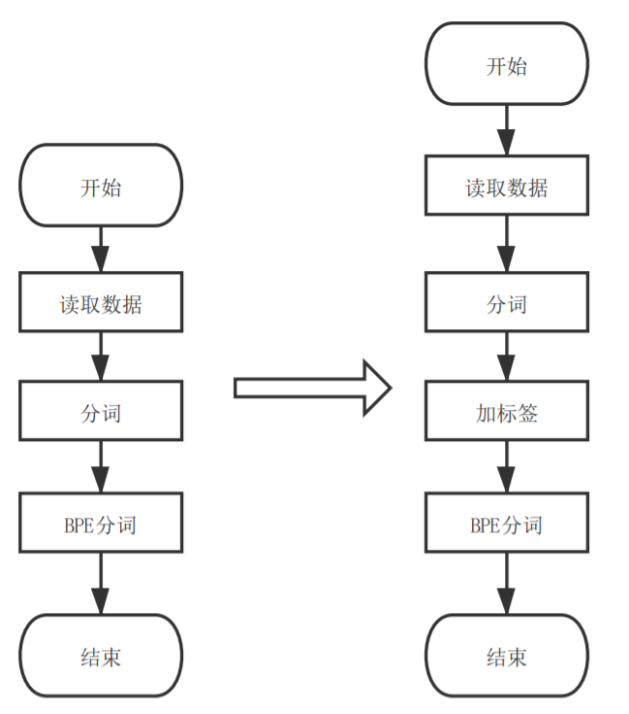


图4.3 公共处理部分流程图转变

### 4.4.2 翻译系统接口封装实现

fairseq目录—fairseq\_cli目录—interactive.py，该文件中存放fairseq中实现交互式翻译的关键函数main()。将main()函数拆分为load\_model()和translate()两个函数，方便Flask后端服务器系统调用。其中，load\_model()函数实现模型加载，translate()接收待翻译句子并返回翻译结果。

# 5 结果展示

## 5.1 Web系统

Web系统展示主要针对多语言多领域机器翻译系统的前端和后端的通信功能以及对于系统特殊情况的处理方案。

（1）用户选择源语言和目标语言且输入待翻译的句子，提示翻译成功，返回翻译系统翻译得到的结果，结果如图6.1所示。

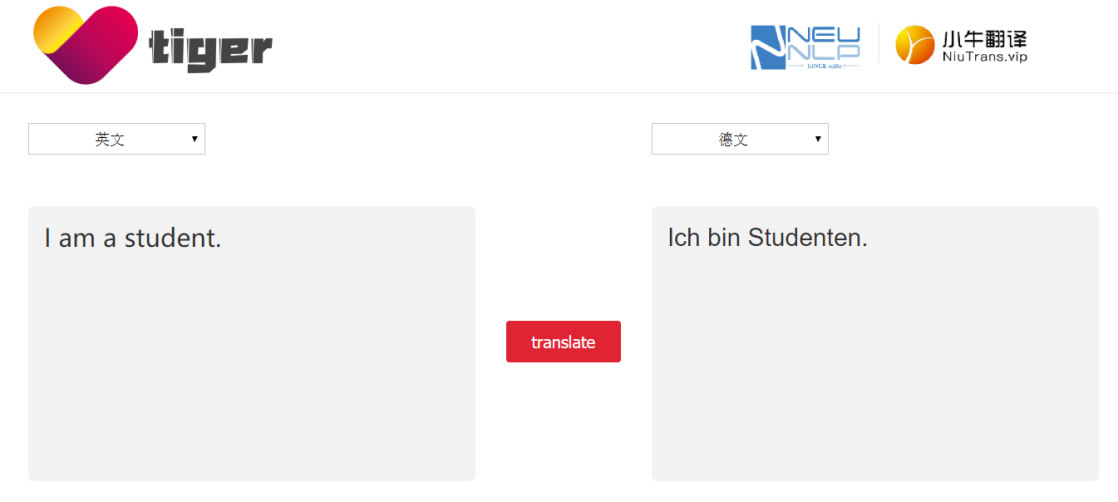


图5.1 翻译成功

（2）用户没有选择源语言或者目标语言，在右侧文本框中输出相应提示，提醒

用户选择语言，结果如图6.2所示。



图5.2 未选择源语言或目标语言

（3）用户没有输入句子，在右侧文本框中输出相应的提示，提醒用户输入待翻译的句子，结果如图6.3所示。

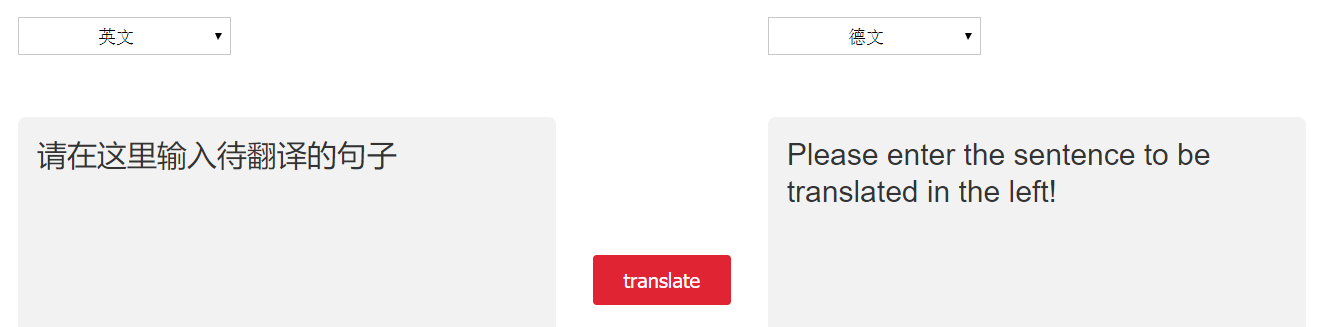


图5.3 未输入待翻译句子

（4）用户选择的源语言和目标语言相同，在右侧文本框中返回原句，结果如图6.4所示。

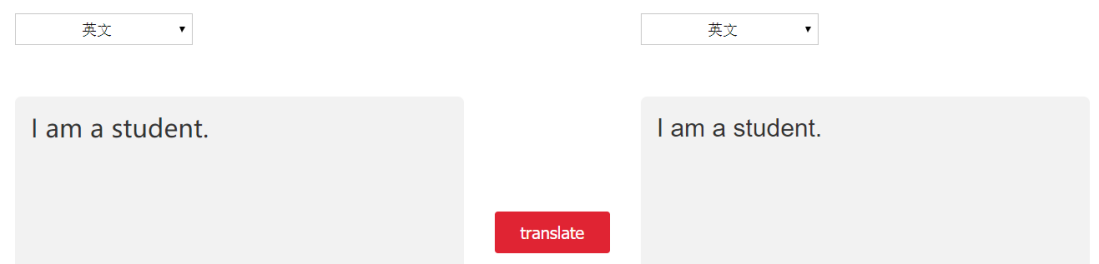


图5.4 源语言和目标语言相同

（5）用户既没有选择语言，又没有输入句子，结果如图6.5所示。



图5.5 未输入句子且未选择语言

从上面的具体测试过程可以看出，Web系统可以正确返回翻译结果且对特殊情况的处理合理妥当，并且反应速度较快，没有出现卡顿现象。由于网络稳定性原因，本服务器并未部署在Linux服务器上，选择部署在本地操作系统。这样充分保证了系统演示过程中不会出现因为网络问题而无法访问Web系统的情形。测试通过。

## 5.2 语言零资源翻译

结果如图6.6所示，系统正确进行了德法之间的翻译任务，并在后台有两次调用接口的提示输出。

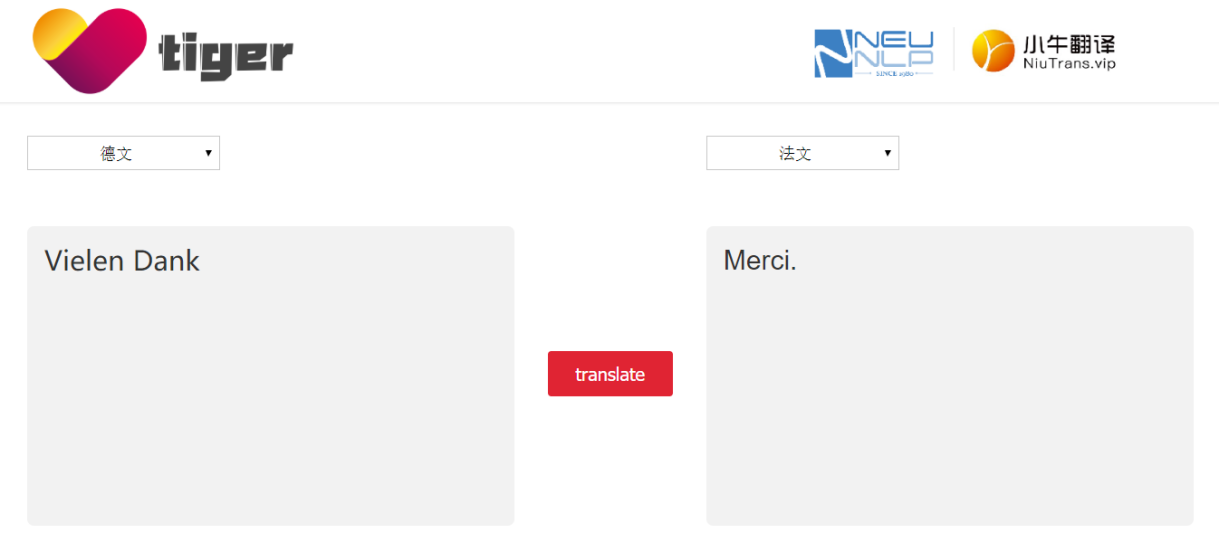


图5.6 德语翻译到法语

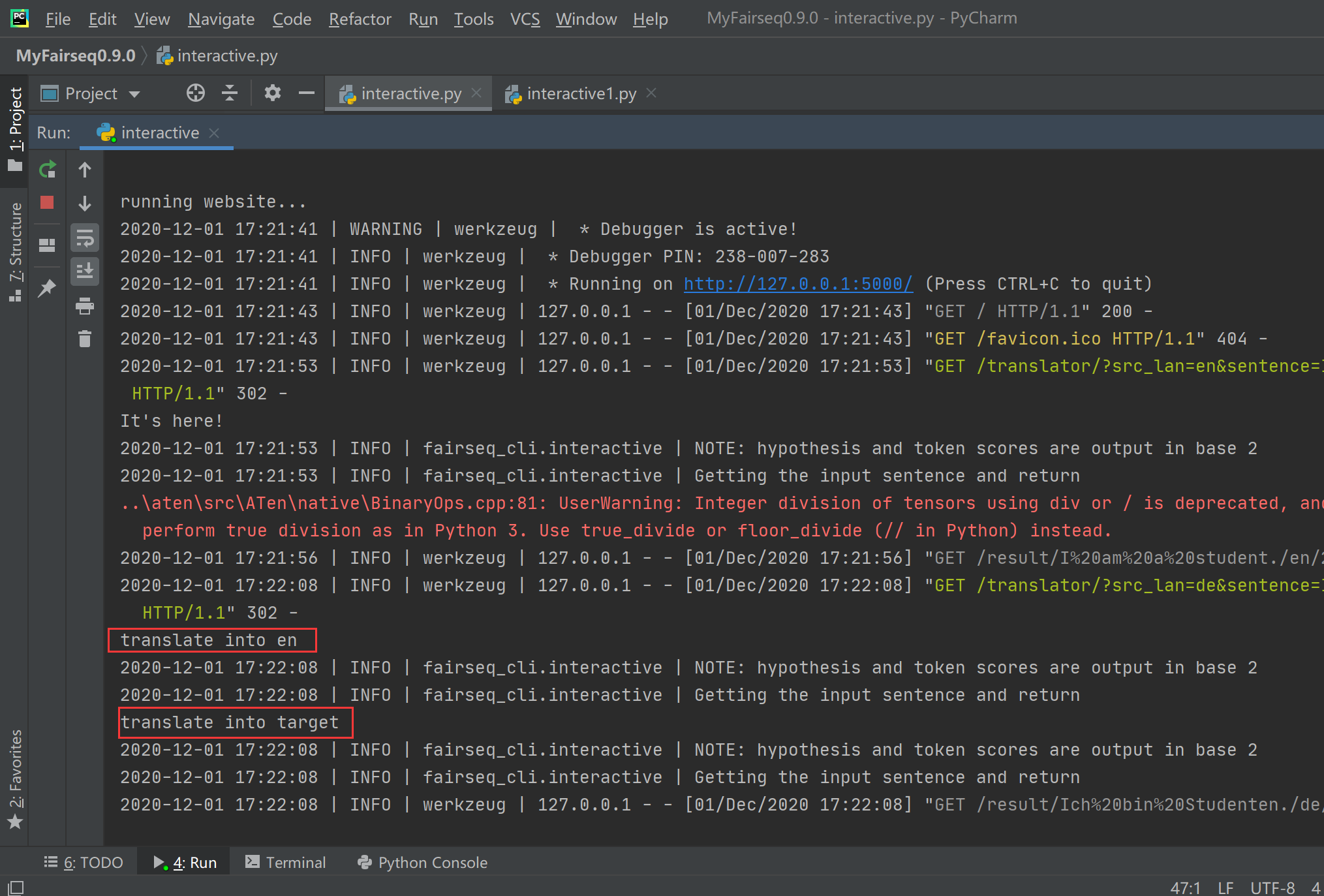


图5.7 后台输出提示

# 6 总结

本人对毕设设计的工作进行迁移总结，再一次完成了一个多语言翻译系统。非常感谢肖老师的指导，希望自己在以后的学术研究中真正做到心之所向、神之所往，以研究问题为基础、以解决问题为目的，努力提升自己的学术水平、自己对这个世界和对自我的认知。