

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 副教授

二○二〇年十二月

# 

# 1 绪论

## 1.1 研究背景

让机器协助语言之间的相互翻译是人类长久以来的梦想。从广义上讲，“翻译”是将一个事物转化成另一个事物的过程，其核心在于对序列的转化。机器翻译（Machine Translation, MT）是利用计算资源把源语言序列转变为目标语言序列的过程。它主要研究不同语言对之间的翻译，如英文和中文、英文和法文之间的相互翻译。上世纪九十年代，在深度学习应用于机器翻译任务之前，机器翻译处于统计建模的时代，使用传统的基于统计的方法，主要有以下特点：基于离散空间的表示模型、自然语言处理问题的隐含结构假设、以特征工程为主但是特征、规则的存储又消耗资源。

伴随着深度学习的不断发展以及应用于机器翻译任务中，神经机器翻译（Neural Machine Translation，NMT）成为机器翻译方法中的“宠儿”,它在许多自然语言对的相互翻译上表现出非常良好的性能。神经机器翻译使用一个单一的大型神经网络来模拟整个翻译过程，它的成功主要归功于语言分布式表示的使用，从而能够实现机器翻译系统端到端的训练。神经机器翻译系统不像统计机器翻译那样需要很多单独的组件，比如单词矫正器、翻译规则提取器和其它特征提取器，它的主要机制模式是词嵌入-编码器-注意力机制-解码器。循环神经网、卷积神经网络以及具有自注意机制结构的Transformer模型等就是建立在这种模式上的极为经典方法。这些方法不仅在机器翻译中取得巨大成功，在自然语言处理领域的许多其他任务中也取得突出的成绩。

鉴于神经机器翻译的强大优势，它在许多商业场景中得到了迅速应用。但是，到目前为止，几乎所有这样的系统都是为单个语言对构建的，还没有一种足够简单有效的方法来使用单个模型处理多个语言对，并且不需要对基本的神经机器翻译网络结构体系结构进行重大更改。同时，高质量的平行语料库被限制在了特定的领域内。在低资源的情况下，普通的机器翻译系统在特定的领域内翻译效果都很差。由于我们可以将每个领域视为一种语言，所以神经机器翻译的语言适配和神经机器翻译的领域适配有很多相似之处和共同的方法[6]。 此背景下，本课题尝试进行利用单一网络解决多语言多领域机器翻译任务。

## 1.2 研究意义

人工翻译尽管进度高，但是费时费力。当需要大量的文本翻译且精度要求不是那么高的时候，比如浏览器上的海量数据，机器翻译的优势就体现出来了。对于人工作业无法完成的事情，使用机器翻译可能仅花费几个小时甚至几分钟就能完成[9]。

在机器翻译中，多语言机器翻译（Multilingual Neural Machine Translation, MNMT）系统是一种非常可取且有效的方案，因为使用来自多个语言对数据的训练模型可帮助模型从多个“源”获得“知识”。此外，由于得到不同的语言对进行训练，多语言神经翻译系统往往具有更好的泛化能力，从而能够提高翻译质量。这种独特的现象也被称为知识迁移。在低资源语言之间的翻译中，知识迁移现象非常明显。低资源语言的平行语料库很少，但却从其他语言对的数据中获益。在零资源语言之间的翻译中，两个语言没有平行语料库，也能实现翻译，或是在没有对应领域的数据情况下实现翻译。并且，由于多语言机器翻译系统是单个模型处理多种语言，它也是一种极为紧凑的系统。这可以有效减少部署所带来的机器硬件消耗成本和人工成本，从而简化翻译系统的大规模部署任务，提高系统收益。最重要的是，研究多语言机器翻译系统可以帮助我们更好地地理解和回答自然语言处理中的一个重要问题:我们如何构建分布式表示，使跨语言的相似文本具有相似的表示?在将不同领域的语料库视为不同语言的情况下，迁移学习必须既跨越语言又跨越领域。因此，多语言多领域机器翻译系统的研发是一件非常有意思的工作并且具有很大的研究和应用价值。

## 1.3 主要工作

（1） Web系统的搭建。本课题搭建了界面友好简洁的交互式Web系统用于交互式翻译演示，并且针对用户的异常操作进行处理。

（2）语种与数据集的确定。关于数据集的确定，本课题所遵循的原则是：数据量具有一定的初步规模、涉及三种语言和两个领域、与国内外学者的工作具有一定的可比性，以便后续科研工作的继续开展。最终确定为WMT14德英、WMT14法英、IWSLT17德英三个数据集。

（3）数据集的处理。本课题使用的数据处理工具如下：python、linux shell编程、Moses分词工具、BPE分词工具。由于不同的数据集具有不同的特点，因此需要据此对数据集采用不同的处理策略。尝试在数据集中加入目标语标签和领域标签、长度过滤等多种处理方案。

（4）翻译系统搭建和接口封装。使用Fairseq训练多语言多领域机器翻译模型，因为考虑到Tensor2Tensor开源工具调试代码的复杂性，并且Fairseq最大的优势在于调试方便、训练速度快、内存占用率低。并且还需要封装翻译接口，使得Web系统得以调用从而实现交互式翻译。

（5）系统性能的提升。本课题从数据处理、加入字典数据以及预训练模型三个方面尝试提升系统性能。

## 1.4 难点分析

本课题所研究的多语言多领域机器翻译系统所要实现的最主要的功能是：使用单一的Transformer模型完成英德法三个国家的语言在口语和新闻领域之间的互译任务。下面将阐述本文的难点同时也是本文的亮点工作：

（1）关于实现德语和法语之间的语言零资源翻译。

零资源翻译是指，训练数据中不包含某个语言对的数据，但是网络通过训练后却达到了进行该语言对翻译的能力。由于本课题采用的数据是WMT14德英、WMT14法英、IWSLT德英，可以实现德语到英语、法语到英语的翻译。为了达到三者互译的目的，首先翻转源语言和目标语言，实现英德、英法两对语言之间的相互翻译。之后以英文为中枢语言实现德法的互译，关键是要实现翻译接口的封装。

（2）关于实现英德口语的领域低资源翻译。

领域低资源翻译是指，在特定的领域内，某些领域的数据量较少，可通过加入该语种其他领域的大量数据进行训练，从而提高该语种在该领域内的性能表现。本课题采用的IWSLT德英数据集属于口语领域，数据量仅仅20w，而WMT14德英数据集属于新闻领域，数据量为190w。以此来提高德英之间在口语领域的翻译效果。

1. 关于系统性能提升。

本课题尝试从数据处理、多卡训练以及预训练模型这三个方面进行系统性能的提升，其中数据处理方面又包含长度过滤、对领域标签的处理以及加入字典数据。并最终将系统性能进行对比，分析这些尝试处理对系统性能的影响。本课题也是第一次尝试利用预训练模型提升多语言多领域机器翻译系统性能的工作，之前未曾有人进行过相关工作。

# 2 相关技术简介

## 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，可以实现高效并行运算。它在序列到序列的机器翻译任务上的表现超过了循环神经网和卷积神经网络，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，最大的优点是可以高效并行化。

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。

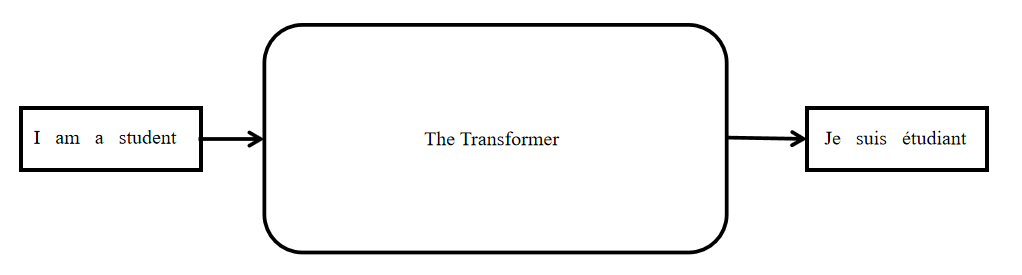


图2.1 Transformer简略图

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。

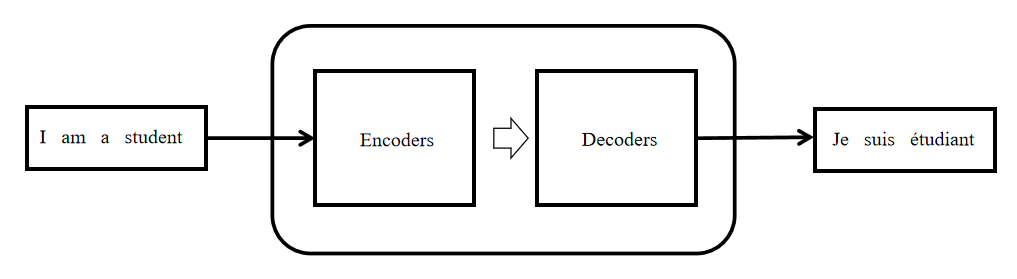


图2.2 Transformer 拆分图

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。

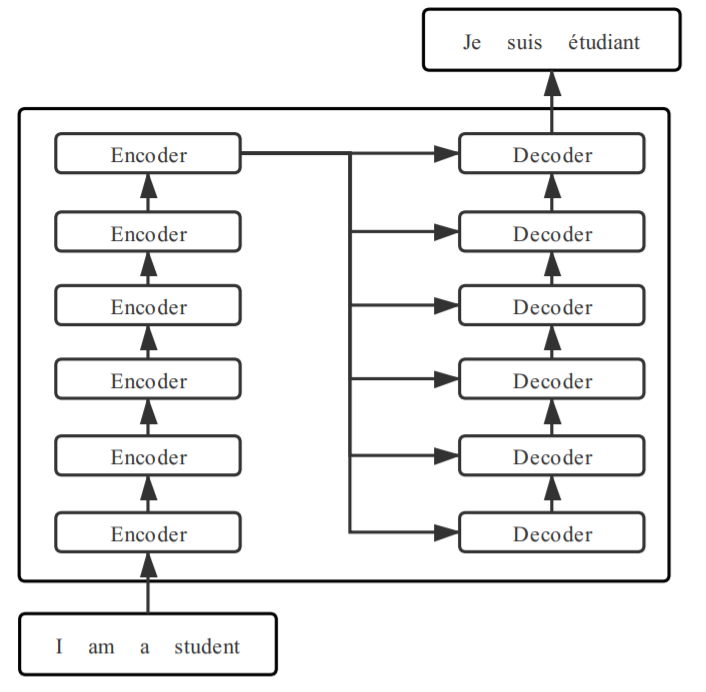


图2.3 Transformer细分图

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。

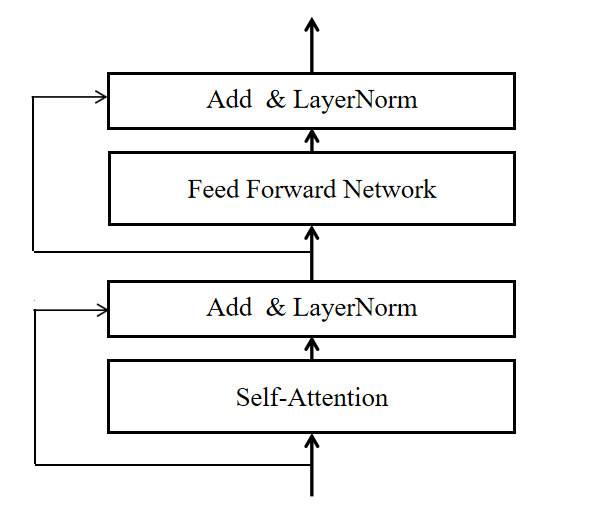


图2.4 编码器细分图

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。

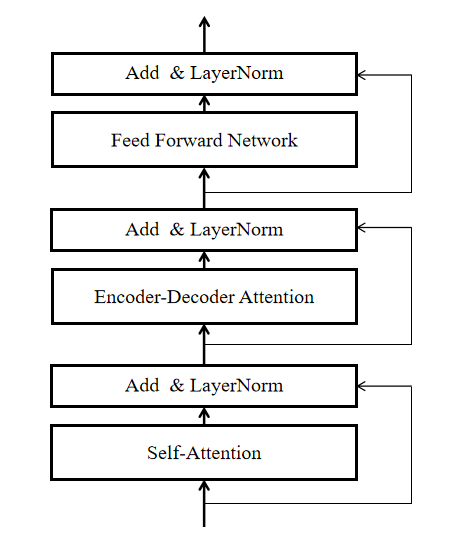


图2.5 解码器细分图

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。

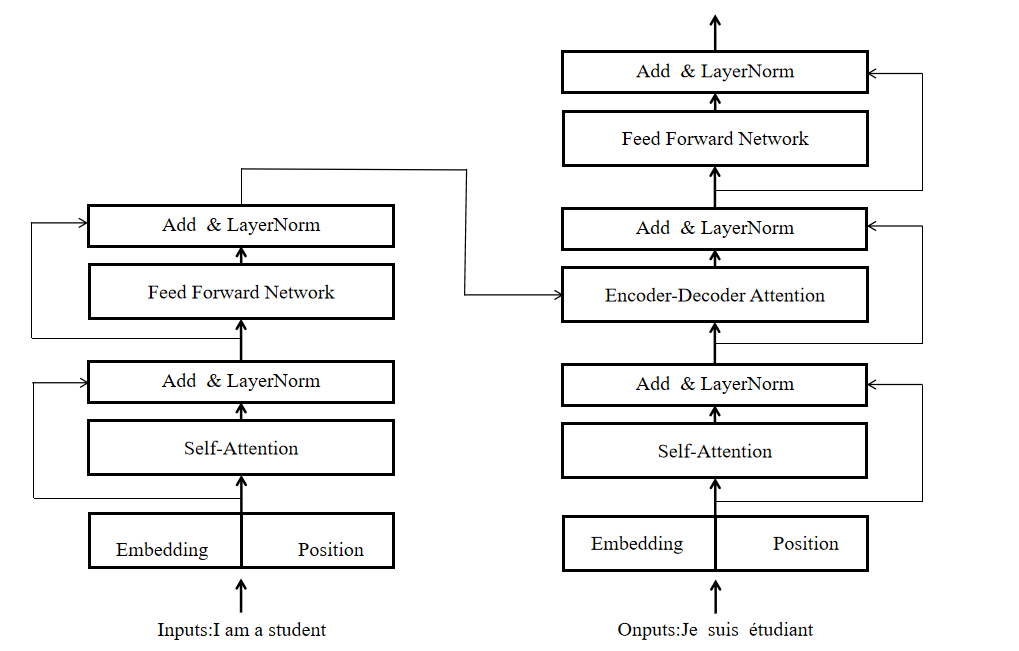


图2.6 Transformer整体图

### 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。

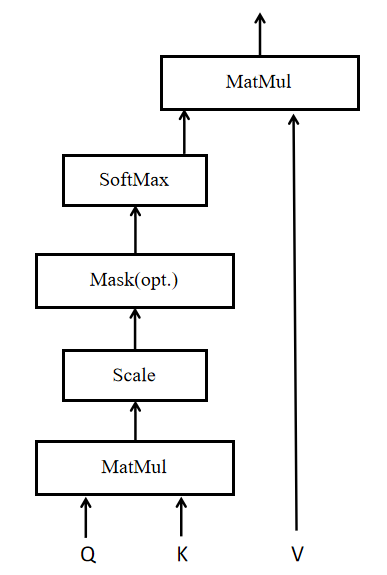


图2.7 基于点乘的注意力机制

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。

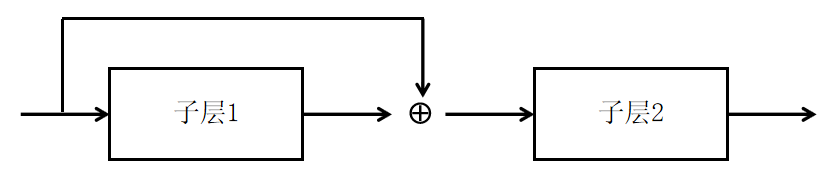


图2.8 残差原理图

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 Bert预训练模型

Bert的全称是Bidirectional Encoder Representation from Transformers，它是谷歌团队于2018年发表的论文中提出的预训练模型，并且在许多自然语言处理领域的任务上取得了较好的成绩。Bert本质是通过大量数据训练进行自监督学习，使得单词学习到好的特征表示，在自然语言处理领域的任务中，我们可以使用BERT预训练得到的特征作为该任务的词嵌入特征，从而提高该任务的性能表现，本文就是采用Bert预训练模型的方法尝试提高系统性能。

Bert的基础集成单元和Transformer种的编码器是一样的，如图2.9所示是Bert的单元结构图，改图反应说明了Bert模型是由一个个Transformer模型的编码所构成。

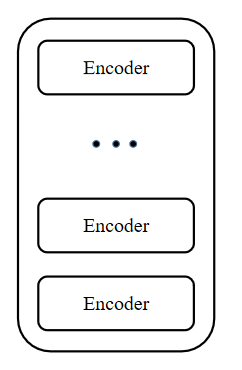


图2.9 Bert单元结构图

Bert模型和Transformer模型编码器部分的数据传递形式是相同的。首先需要将固定长度的句子作为输入，数据由最底层的编码器向上传递，每一层编码器也都包含自注意力层和前馈神经网络层。Bert的模型结构如图2.10所示。

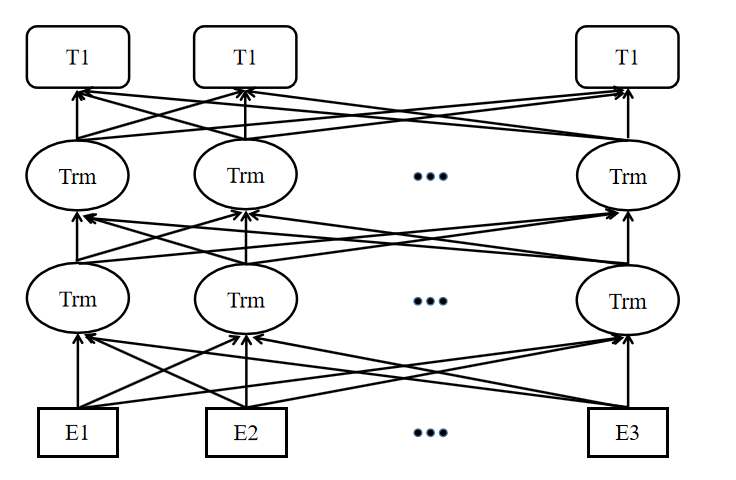


图2.10 Bert模型结构

其中，每个Trm表示Transformer种的编码器，E表示词嵌入Embedding，T表示Bert的输出。Bert采用两阶段模型，首先是预训练语言模型，之后是微调（Fine-Tuning）模式解决自然语言处理领域的各种下游任务。但是NLP领域下游任务种类繁多，比如序列标注类任务、分类任务、句子关系判断以及生成式任务，本课题所设计的多语言多领域机器翻译系统就属于其中的生成式任务。针对不同的下游任务，Bert需要改造输入输出使得不同的任务可以使用Bert预训练好的模型参数。

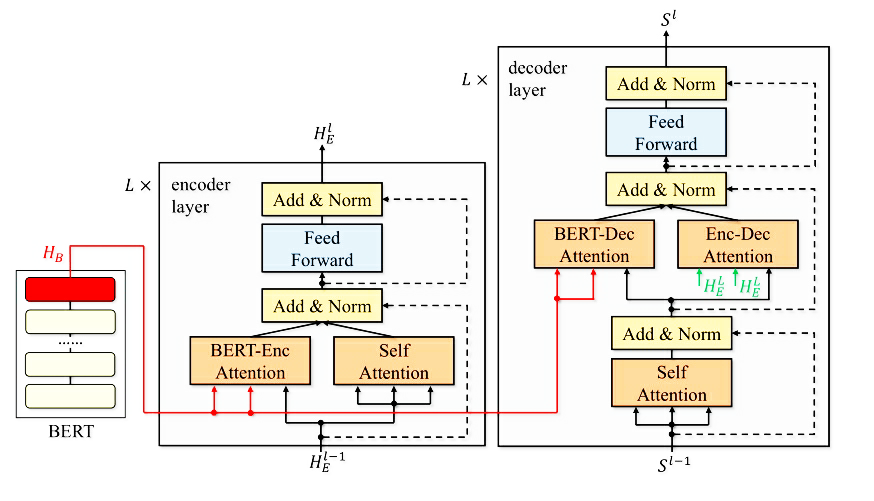


图2.11 BERT-fused模型结构图

本课题应用了微软亚研院提出的将Bert应用到神经机器翻译中的方法，进行系统性能的提升，它提出的BERT-fused模型算法使用预训练模型的输出作为翻译模型词嵌入的输入，如图2.11所示是BERT-fused模型结构图。

## 2.3 Moses开源工具

Moses是机器翻译采用统计（或数据驱动）方法实现的[12]。这种方法曾经是该领域的主导方法，谷歌和微软等公司部署的在线翻译系统也采用过这种方法。

Moses是一个很强大的开源统计机器翻译系统，而本课题在系统搭建的过程中，主要使用了mosesdecoder中的分词脚本tokenizer.perl。该脚本在英德法三国语言上的分词效果极佳，由于在欧洲语言中本来就是以空格分隔单词，因此该脚本最主要的作用是将标点符号和单词之间也以空格为分隔符分离单词。

此外，除了mosesdecoder中的分词脚本外，本课题还使用了Moses开源工具中的清理句子长度的脚本clean-corpus-n.perl，该脚本主要用于过滤平行语料库中句子长度过长的数据，长度超过规定的平行语料将会被舍弃，该脚本会同时检测源语言和目标语言。

## 2.4 BPE分词工具

在神经机器翻译中，通常会根据训练数据里面的单词或者短语形成词表，模型在训练或者解码过程中将使用它。在神经网络的训练过程中，需要对词表中每个词进行词嵌入Embedding，一个词向量对应一个单词，所以即便两个单词表达的是同一个意思，在训练过程词向量之间没有任何关联。这导致了一个单词因为拥有不同的形态而产生不同的向量表达形式。机器翻译的词表大小是固定的，但是需要实际翻译的时候可能会遇到词表之外的词，这种问题也被称为词表外（Out of Vocabulary，OOV）问题[19]。以前的做法是新词汇添加到词典中，但是过大的词典会带来两个问题：

（1）词频稀疏: 个别单词出现的频率很低，在训练过程中得不到充分的训练，导致最后的训练效果不好。

（2）计算量问题: 词典过大，也就意味着单词词向量Embedding的维度很大，在方向传播的过程中将会产生大量的梯度，也会因此而占用过大的计算资源。

与此同时，考虑到再大的词典也无法真正覆盖所有的词汇，这种单词级别(word-level)的处理方式并不能通过增大词表真正解决OOV的问题。为了处理这个问题，有人便提出了是将字符当做基本单元，建立字符级别的模型(character-level)。字符级别的模型试图使用比单词基本单元更小的单元符号来表示所有的词汇，比如26个英文字母，相比于word-level模型，这种处理方式的粒度变小，其输入长度变长，使得数据更加稀疏并且难以学习长远程的依赖关系。单词级别模型导致严重的OOV，而字符级别的模型粒度又太小，那么子词级别（subword-level）的处理方式就应运而生。

根据不同的策略，产生了3种不同的子词算法：BPE[9]（Byte Pair Encoding）、wordpiece和[Unigram Language Model](https://link.zhihu.com/?target=https://arxiv.org/abs/1804.10959" \t "https://zhuanlan.zhihu.com/p/_blank)。这三种算法均采用统计学的思路。本课题采用的是自下而上的BPE算法，它是NLP任务中最为简单有效的方法。本课题使用subword算法相关开源工具subword-nmt，其处理流程主要分为两方面，一是子词表的学习获取，二是子词表的应用进行切分子词。

BPE是一种数据压缩的方式，它将字符串中最常见的一对连续字符数据替换成该字符串中不存在的字符串，后续再通过一个词表重建原始的数据。BPE的处理过程可以理解为一个单词的再拆分过程。如"loved","loving","loves"这三个单词，其本身的语义都是”爱”的意思。BPE通过训练，能够把上面的3个单词拆分成”lov”,”ed”,”ing”,”es”几部分，这样可以把词的本身的意思和时态分开，有效的减少了词表的数量。

## 2.5 Fairseq开源系统

序列到序列模型已经成功地应用于多种文本生成任务，包括机器翻译，抽象文档摘要和语言建模。 因此，相关人员都可以从一个快速且易于扩展的序列建模工具包中获益。有几个具有类似基本功能的工具包，但它们在关注领域和预期受众方面有所不同。例如，OpenNMT是一个社区构建的工具包，用多种语言编写，强调可扩展性。Tensor2tensor关注生产准备度。Fairseq是用PyTorch编写的深度学习网络建模工具，允许研究开发人员训练用于自然语言处理任务的自定义模型，它是一个快速的，可扩展的，对研究和生产都有用的工具包。Fairseq开源系统工具具有如下特性：

1. 跨模型和任务的公共接口，可以用用户提供的插件进行扩展。
2. 高效的分布式和混合精确训练，能够在现有硬件上对具有数亿个句子的数据集进行训练。
3. 机器翻译，摘要和语言建模的最先进的实现和预先训练的模型。
4. 利用多种支持的搜索算法优化推断，包括束搜索，多样束搜索。

## 2.6 其他相关开发工具

### 2.6.1 PyCharm 2019社区版

PyCharm是一款Python集成开发环境，可以帮助用户提高开发效率。使用社区版本可以充分满足此次系统搭建的要求。PyCharm2019集成开发环境具有如下主要特点：

1. 具有代码补全、代码折叠、窗口规划、检查语法错误的功能。
2. 创建使用python解释器更加方便。
3. 自动进行Python库的更新。
4. PyCharm 2019带来了大量性能提升：Python 索引速度是原来的两倍。

### 2.6.2 Anaconda

Anaconda可以轻松获取Python语言的各种包且对包进行管理，同时对环境可以统一管理。Anaconda包含了conda、Python在内的超过180个科学包及其依赖项，具有简单、安装过程简单、免费的特点。Anaconda所含的包基本满足本课题需要。

### 2.6.3 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。

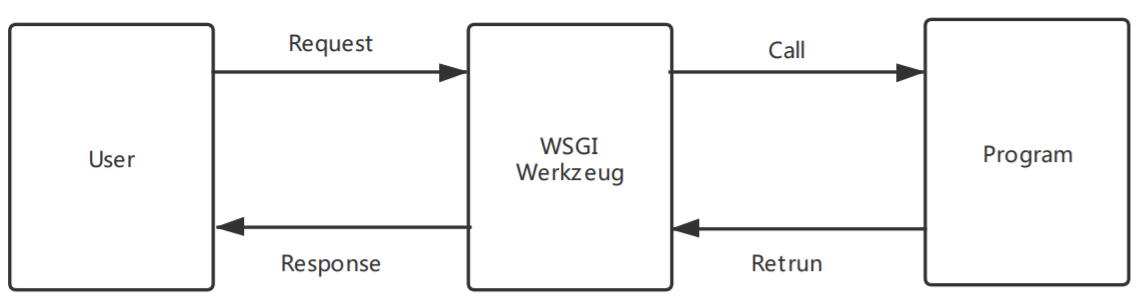


图2.12 Flask框架工作过程

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。

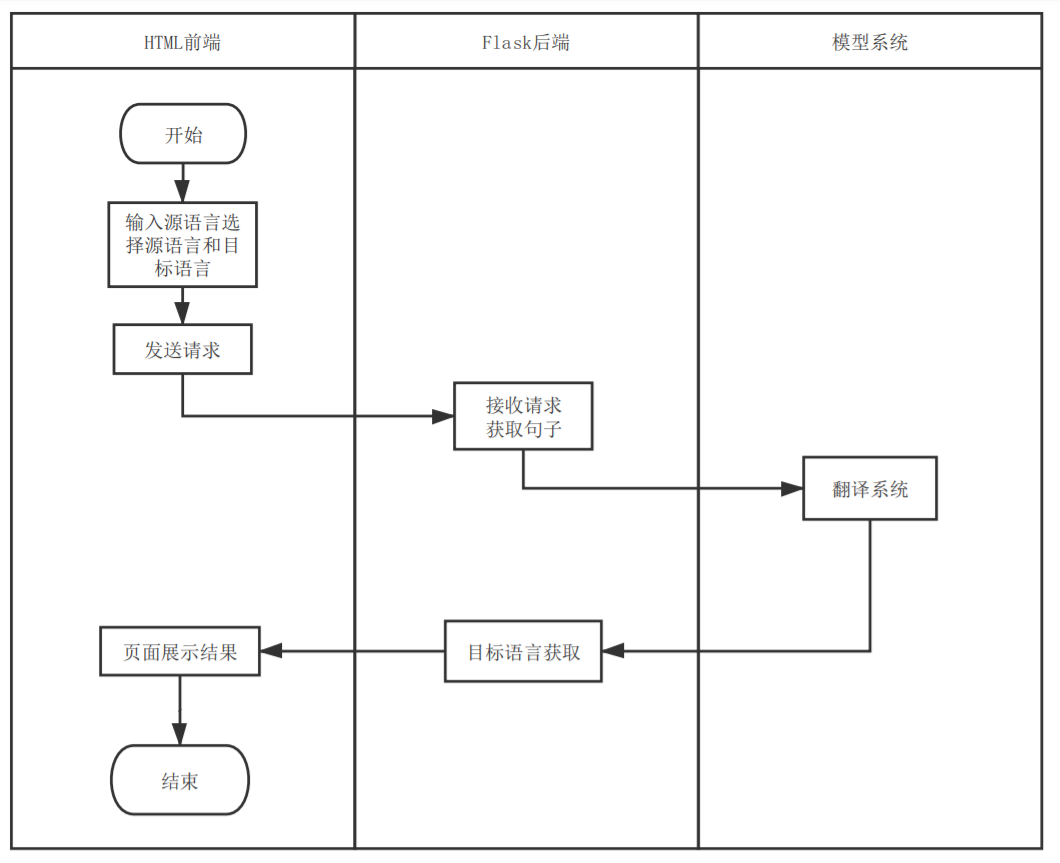


图2.13 系统活动泳道图

## 2.6 本章小结

本章以本课题在搭建整合系统过程中所依次使用到的关键技术和工具为顺序，分别介绍了翻译系统所使用的Transformer模型的网络结构和特点，数据预处理所用到的Moses分词工具和BPE分词工具，用于训练模型的Fairseq开源系统，最后介绍了搭建和整合Web页面和翻译系统的集成开发环境PyCharm、包管理工具Anaconda和Web框架Flask，为后续系统的详细实现做好充足的准备。

# 3 系统设计

## 3.1 系统总体设计

首先，根据本课题题目多语言多领域机器翻译系统设计与实现，并结合需求分析的结果，设计出多语言多领域机器翻译系统功能层次图。该层次图从较高的层次描述出系统包含的子系统，以及各个子系统所要实现的具体功能。如图4.1所示。

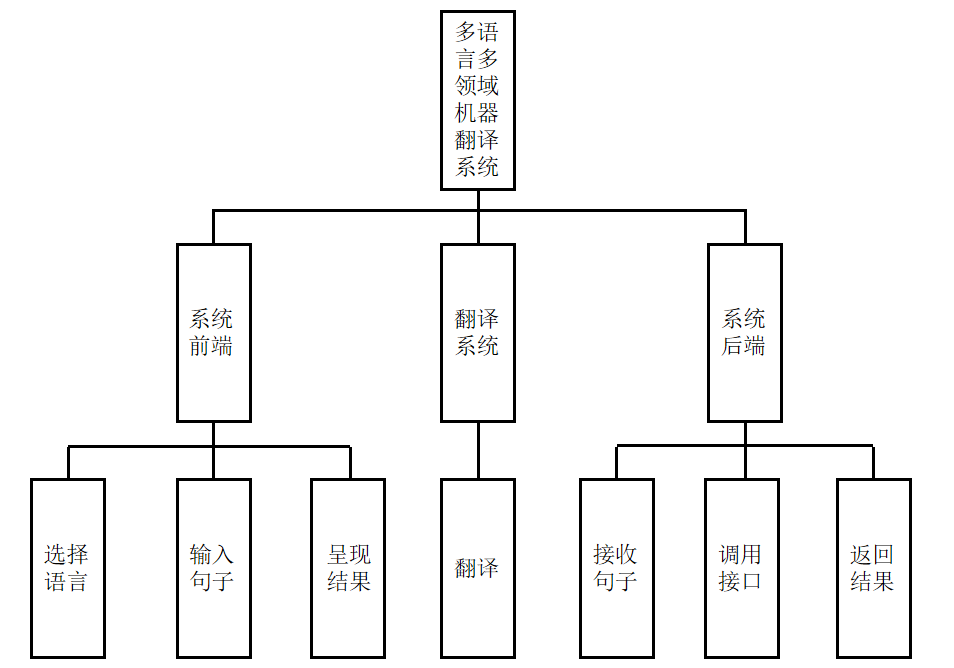


图4.1 系统功能层次图

根据层次图可以发现，本课题需要完成一个可供用户使用的交互式翻译系统，其核心工作包括三方面的内容，一是Web系统的设计，二是翻译系统的设计，三是系统的整合和完善。并且在完成最终系统后，需要考虑如何提升系统性能。

## 3.2 Web系统设计

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。

### 3.2.1 Web前端页面设计

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。

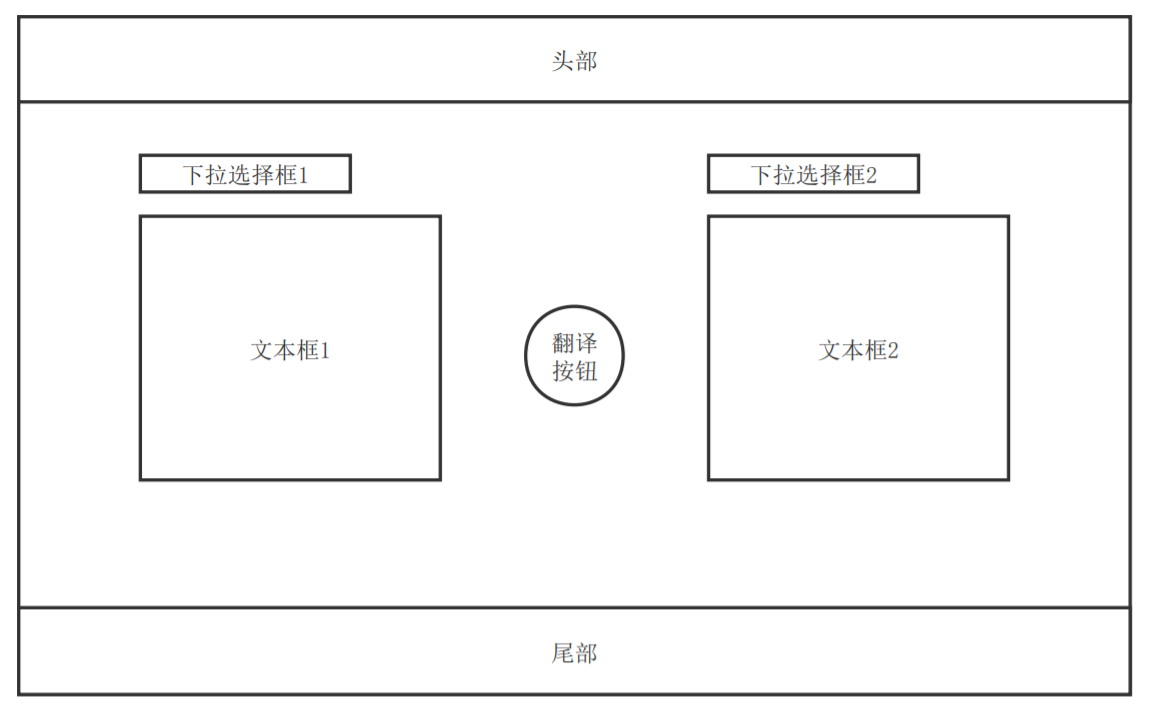


图4.2 Web前端页面布局设计图

### 3.2.2 Web后端服务器设计

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。

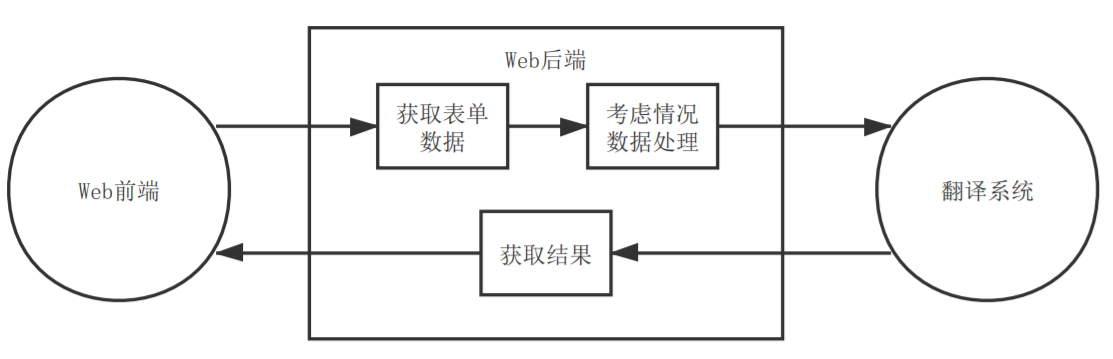


图4.3 Web后端信息流动图

## 3.3 翻译系统设计

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。

### 3.3.1 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。

#### 3.3.1.1 WMT和字典数据处理方式

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。

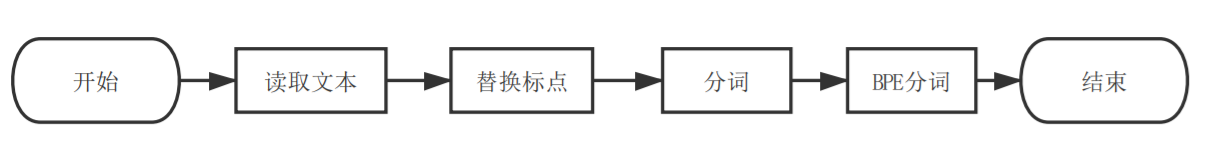


图4.4 WMT、验证集和测试集数据处理流程图

#### 3.3.1.2 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。

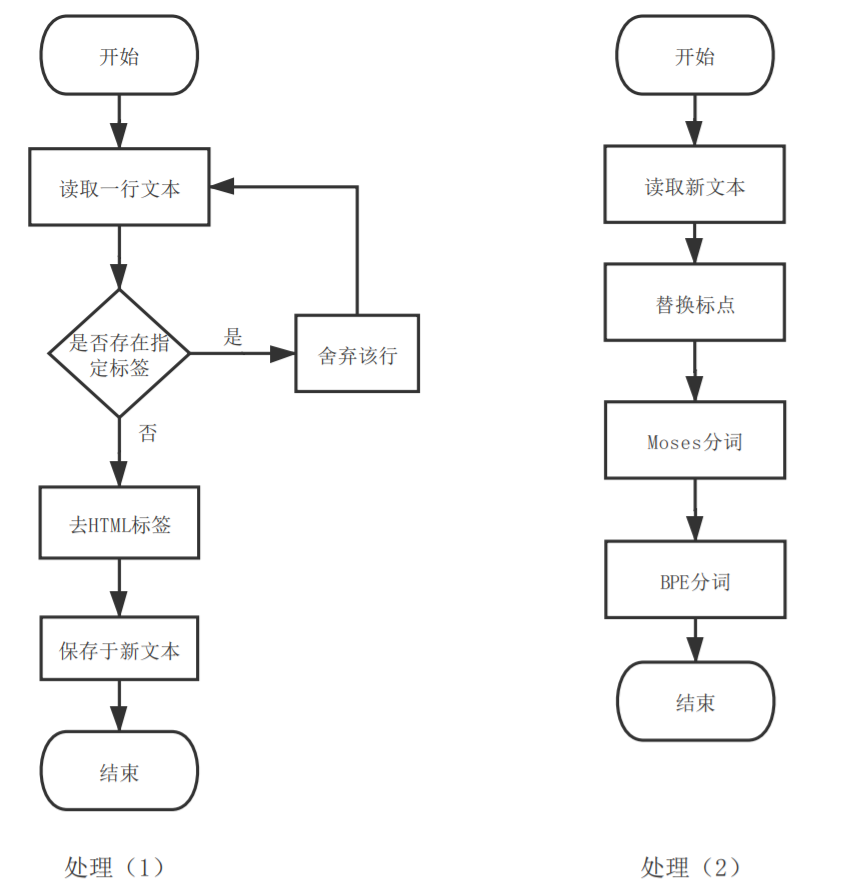


图4.5 IWSLT数据处理流程图

#### 3.3.1.3 多语言和多领域处理

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。领域标签处理同目标语标签。

### 3.3.2 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。

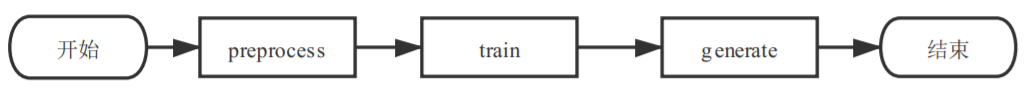


图4.6 翻译系统搭建流程图

## 3.4 性能提升设计

在翻译系统性能方面，本课题计划从以下几个方面进行系统性能提升。

### 3.4.1 数据处理方面

1. 句子长度过滤。训练语料中有些句子特别长，而这些句子实际上出现的次数极少，同时模型要想拟合超长句子特别难，所以可以把这种超长句子视为噪声数据，在训练之前过滤掉。同时，训练数据中存在着句子长度比过大的句子，比如源语言句子过长、目标语据种子过短，或是源语言句子为空的现象，这些也视为噪声数据进行过滤。
2. 领域标签。因为机器翻译中的语言适配和领域适配具有一定等价性和相似性，因此可以将不同领域的训练集等价于不同语言的训练集。故设计在数据处理阶段给在源语言句子开头加入目标语标签的同时加入领域标签，进行系统性能提升的实验结果观察。

有口语领域英语到法语的句子对：I am a student -> Je suis étudiant

那么源语言的句子将会被修改为：2fr ted I am a student -> Je suis étudiant

1. 尝试加入数据字典数据。从WMT20比赛中下载字典数据，字典数据是德英和德法的单词对应翻译，处理好可尝试后加入训练集尝试提升系统性能。考虑这种做法的原因是使得网络对于个别单词的表示形式进一步加强，从而提高具体单词的翻译效果。

### 3.4.2 Bert预训练模型

通过改造Bert的输入输出使得大部分自然语言处理领域的任务都可以使用

Bert预训练好的模型参数，使得这些下游任务性能提升。而且机器翻译属于自然语言处理领域的生成式任务，自然也联想到采用Bert预训练模型尝试提升性能。本课题初步设想采用微软亚研院2020年最新提出的Bert-fused模型使用预训练模型的输出作为翻译模型词嵌入的输入来尝试系统性能是否提升。

### 3.4.3 多卡训练

即多卡训练。多数情况下，采用梯度下降算法的训练过程往往需要较大的

Batch Size才可以获得较为良好的性能，考虑到本课题采用的数据较多、网络较为复杂且GPU资源有限，因此尝试三卡分布式训练以及预训练四卡分布式训练，进行系统性能的提升。

## 3.5 系统整合

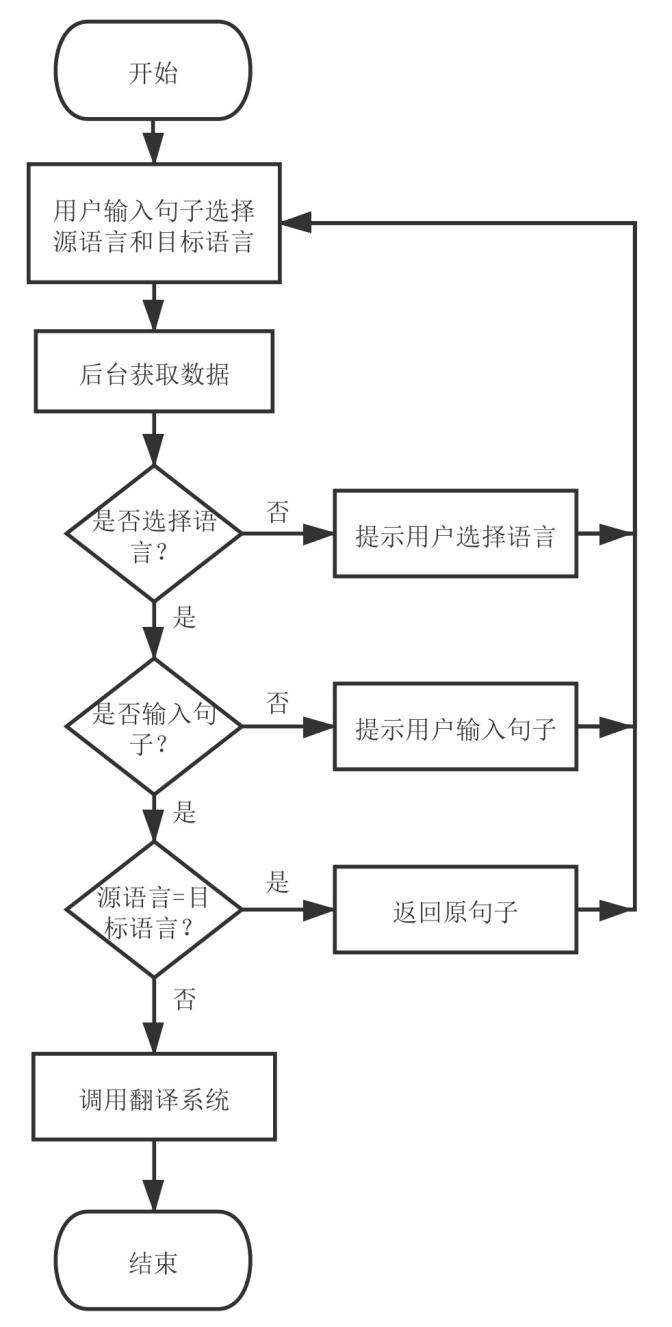


图4.7 系统整体工作流程图

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。

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。

## 3.6 本章小结

本章开头介绍了系统设计所要遵循的原则。在这些原则的指导下，采用自顶向下的设计准则将总系统拆分成两个个子系统：Web系统和翻译系统，其中Web系统又可拆分成Web前端页面和Web后端服务器。最后分别从Web系统设计、翻译系统设计以及系统整合三个方面详细展现了系统整体和局部的设计结果。本章是后续系统实现的“图纸”。

# 4 系统实现

## 4.1 系统训练环境

表**5.1** 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**5.2** 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 系统开发环境

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5.3 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系统搭建

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 Web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5.1 Web页面实现效果图

### 4.3.2 Web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。

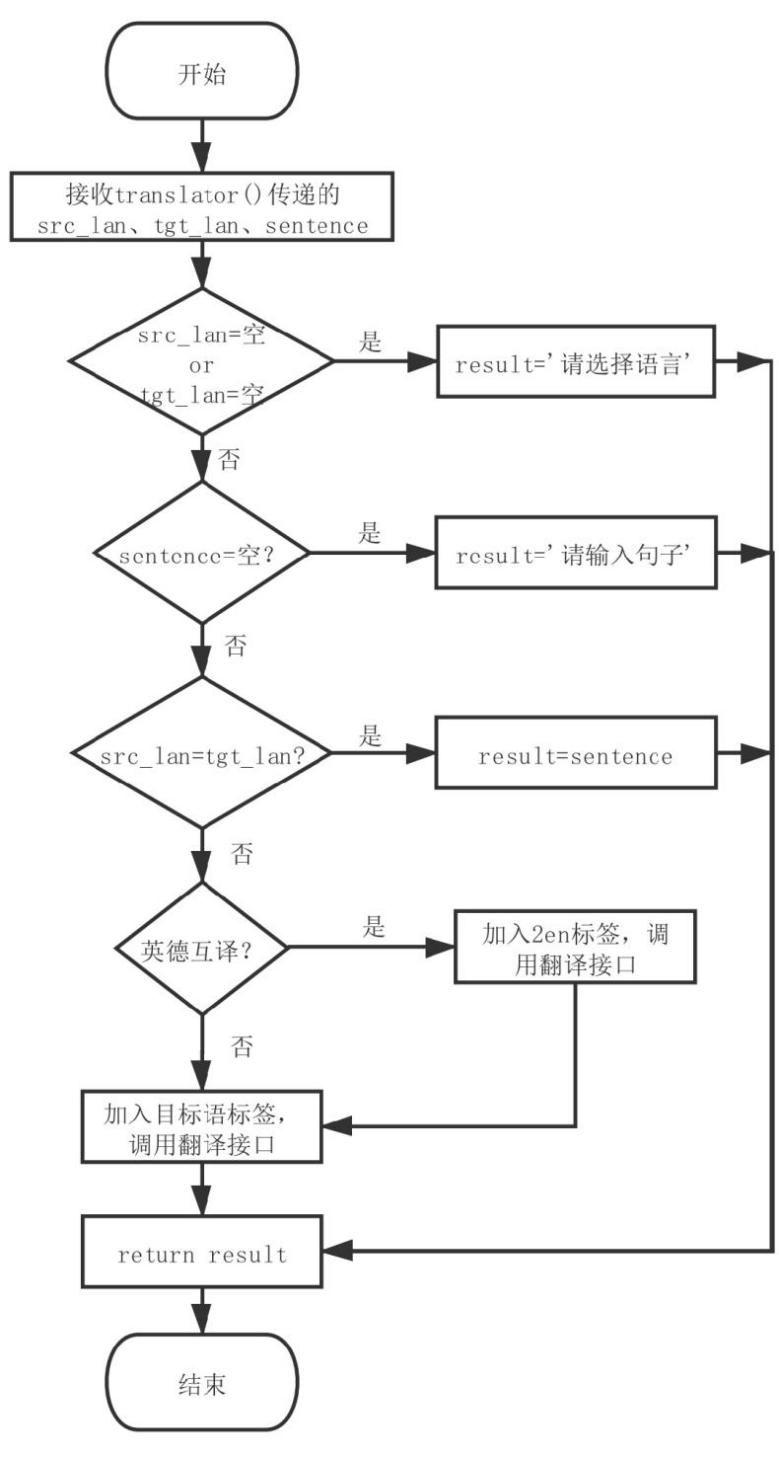


图5.2 result()函数程序流程图

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。

## 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。

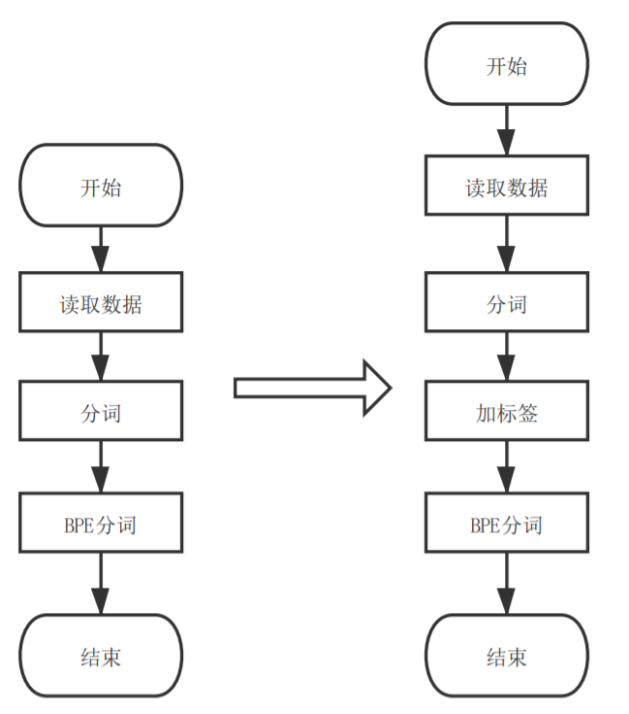


图5.3 公共处理部分流程图转变

### 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。

## 4.5 性能提升实现

在系统设计阶段初步计划从三个方面进行系统性能的提升。分别是数据方面、Bert预训练模型以及分布式训练，其中数据方面主要包括数据长度过滤、加入领域标签、加入数据字典数据。下面对采用这几种处理方式的实现过程和结果进行说明。

首先使数据处理方面进行系统性能的提升，对数据进行的额外处理工作主要包括以下三方面，其中需要注意的是，基准处理在5.4.1小节已经进行说明，如图5.4所示是各种数据处理的流程对比图。

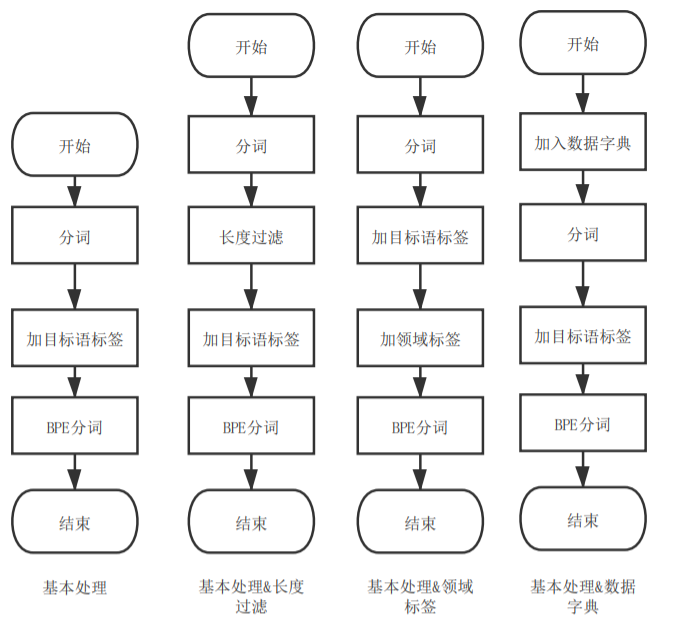


图5.4 各个处理方式流程对比

基准处理&长度过滤：在基准处理的基础上对训练集句子进行长度过滤，保留长度在1-100的数据。同时将源语言和目标语句子为空的语言对以及长度比过小或过大的句子对过滤掉。训练语料中有些句子特别长，而这些句子实际上出现的次数极少，同时模型要想拟合超长句子特别难，所以可以把这种超长句子视为噪声数据，在训练之前过滤掉。

基准处理&领域标签：在基准处理的基础上在训练集中加入领域标签进行训练。对于不同的语言对可以加入目标语标签，那么模型通过具有指定目标语标签的数据，便学习获得了将带有目标语标签的句子翻译到指定目标的能力。那么完全可以将不同领域的语言等价为不同语言对进行相同的处理，以提高模型在领域的翻译性能，也就是所谓的领域适配和多语言机器系统具有共通之处。

基准处理&字典数据：在训练集中加入数据字典数据，其中德英和德法的数据字典等价于句子长度为1的平行语料库，因此均采用和WMT德英和德法相同的处理方式。类似人类学习语言时先学习单词后学习句式的过程，在训练数据中加入字典，希望模型提高对于个别单词的拟合效果，从而提高对于句子的翻译效果。

预训练模型方面，本课题在如图5.4所示的基准处理的基础上采用Bert-fused模型首先单卡训练15轮，每轮102724步，使用的Bert模型是训练好的模型，模型名称为bert-base-multilingual-uncased。

多卡训练方面，本课题尝试在服务器上进行三卡训练，显卡的型号将在系统测试部分给出。

训练结果如下表5.4所示，表中的对于各种处理方式的流程说明如图5.4所示。

表**5.4** 训练结果对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 处理  方式 | GPU  数量 | 训练  步数 | BLEU |
| 基准处理 | 1 | 15轮\*71852步 | 30.50 |
| 基准处理&长度过滤 | 1 | 15轮\*67620步 | 31.05 |
| 基准处理&领域标签 | 1 | 15轮\*72913步 | 30.34 |
| 基准处理&数据字典 | 1 | 15轮\*71911步 | 30.46 |
| 基准处理&长度过滤&领域标签 | 1 | 15轮\*68943步 | 30.88 |
| 基准处理&预训练 | 1 | 15轮\*102724步 | 31.12 |
| 基准处理 | 3 | 15轮\*23951步 | 30.74 |
| 基准处理&长度过滤&预训练 | 4 | 15轮\*26627步 | 31.53 |

上各个处理方式均采用单卡训练15轮，其他的参数设置均相同。通过观察表5.4可以看出，基准处理的BLEU值为30.50。在基准处理的基础上对数据集进行长度过滤可以提高系统性能，BLEU值为31.05，提升了0.55。而在基准处理的基础上加入领域标签会使系统性能下降，BLEU值为30.34，下降了0.16。在数据集中加入数据字典将会对系统性能有略微的负面影响，BLEU值为30.46，降低了0.04。

可以看出，加入领域标签的处理方式对系统性能有较大的负面影响，为了验证加入领域标签的这种影响，进行基准处理&长度过滤&领域标签的处理，即在基准处理的基础上进行长度过滤保留句子长度在1-100的句子，同时加入领域标签。可以看出，相较于基准处理&长度过滤，BLEU值降低了0.17。

而采用预训练模型的输出作为NMT系统词嵌入的输入的方式，训练完成后，解码测试系统的BLEU值为31.12，系统性能提升效果最为明显。

采用多卡分布式训练对于系统性能有略微的提升，BLEU值为30.74，提高了0.24。

总之，在尝试提升系统性能的工作中，对训练数据进行长度过滤和多卡训练对系统性能的提升有一定的正面作用，但是多卡训练的效果有待进一步验证。加入领域标签或者数据字典将会略微降低系统性能。采用Bert预训练模型提高系统效果最为明显。

因此本课题最终在基准处理的基础上进行句子长度过滤以及预训练模型，采用四卡训练15轮，BLEU值达到了31.53。

## 4.6 本章小结

本章首先介绍了训练模型和开发系统的硬件和软件资源配置情况，之后以系统设计阶段的产出作为“图纸”，从Web系统搭建和翻译系统搭建两大主体内容出发，介绍了多语言多领域机器翻译系统的具体实现。其中Web系统搭建部分主要是前端HTML页面以及后端Flask函数代码的实现。而翻译系统搭建部分重点在于数据处理，以及改变数据处理策略实现系统性能提升。对比系统设计和系统实现可以发现，在系统设计阶段并未考虑到的德法无法互译的情况，在系统实现阶段得以发现并解决。

# 5 系统运行结果

系统测试关注系统是否“做对”，是对系统实现的功能和结果进行度量的关键活动。本章将从以下三大方面进行系统测试：一是对Web系统的基本功能以及稳定性测试，二是训练数据的大小对系统性能的影响，三是翻译系统性能对比测试。

## 5.1 Web系统测试

Web系统测试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进行测试。

### 5.1.1 测试方案

（1）测试对象

多语言多领域机器翻译系统Web服务器。

（2）测试目的

通过模拟用户操作，测试系统前端是否正确向系统后端提交表单数据，系统后端是否向前端返回结果数据，以及系统对于以下四种情况的处理：用户没有选择源语言、用户没有选择目标语、用户没有在文本框中输入待翻译的句子、用户选择的源语言和目标语言相同。

（3）测试过程

在本地服务器运行系统，加载训练好的翻译模型，之后打开系统的HTML页面，通过模拟用户的操作，分别对上述情况进行检测。

（4）评测标准

若用户正确选择语言，并且输入待翻译的句子，则返回翻译结果。

若用户没有选择源语言，则系统应提示用户选择语言。

若用户没有输入待翻译的句子，则提示用户输入句子。

若用户选择的源语言和目标语言相同，则返回原语句。

若用户既没有选择语言也没有输入句子，则系统需要提示用户选择语言并且输入待翻译的句子。

上述提示操作均在HTML页面的右侧输入框完成。

### 5.1.2 结果及评价

（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。

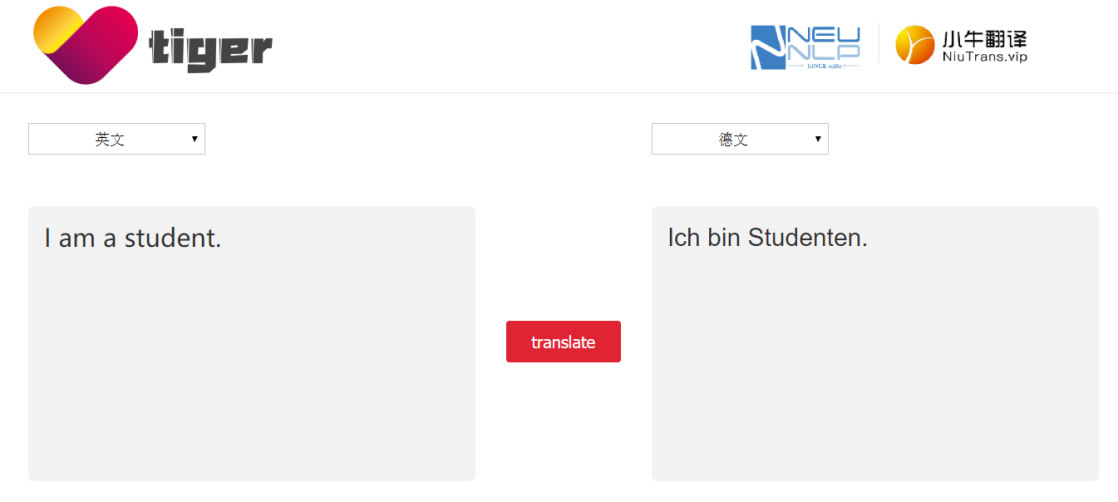


图6.1 翻译成功

（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6.2 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。

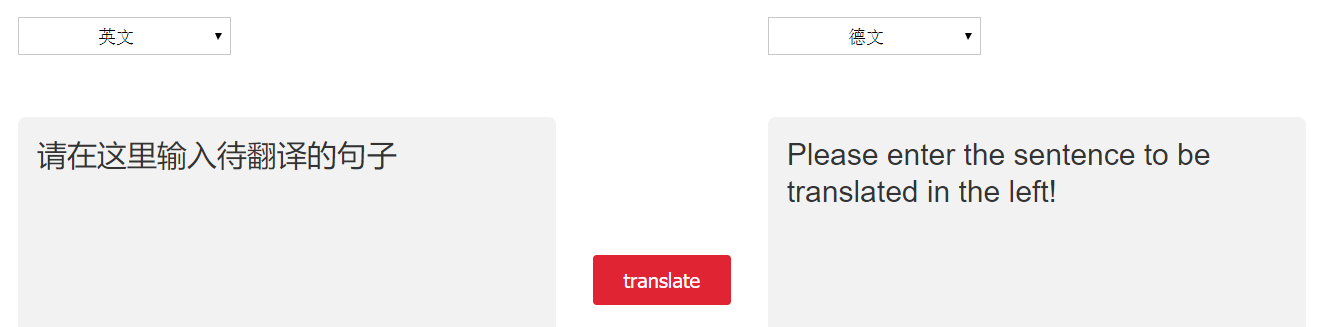


图6.3 未输入待翻译句子

（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。

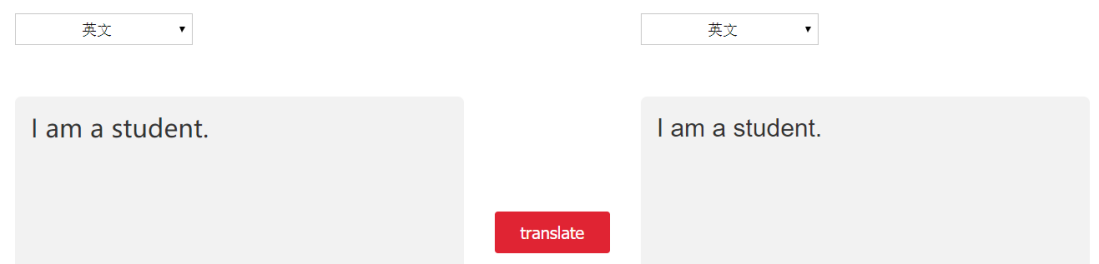


图6.4 源语言和目标语言相同

（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6.5 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 领域低资源翻译测试

将多语言多领域机器翻译系统的数据分开单独训练，对比综合训练的结果，分析加入德法在新闻领域的数据集对其在口语领域的影响。

### 5.2.1 测试方案

（1）测试数据

测试数据如下：WMT14德英、法英以及IWSLT17德英三个数据集。

（2）测试目的

通过对比训练，分析数据量对于系统性能的影响。

（3）测试过程

将三个数据集六个方向的语言对数据放入同样的模型训练，训练步数、学习率等训练的参数设置相同，对翻译结果进行对比分析。

（4）评测标准

针对不同训练集所训练相同步数后的模型进行解码得到的BLEU值。

### 5.2.2 结果及评价

表**6.1** 训练集分开训练效果表

|  |  |  |
| --- | --- | --- |
| 数据集 | 数据量 | BLEU |
| IWSLT17英语→德语 | 20w | 11.86 |
| IWSLT17德语→英语 | 20w | 13.51 |
| WMT14英语→德语 | 180w | 25.62 |
| WMT14德语→英语 | 180w | 31.01 |
| WMT14英语→法语 | 190w | 31.30 |
| WMT14法语→英语 | 190w | 34.72 |
| 综合 | 780w | 31.05 |

训练数据解码测试结果如表6.1所示。训练集为WMT14德英及德法训练集以及IWSLT17德英数据集，验证集和测试集来自WMT06 shared-task对应语种。比如，若训练WMT14德英数据集，则验证集和测试集采用WMT06 shared-task中的德语和英语部分的Development Data和Test Data。综合是指将上述训练集混合在一起进行训练。训练集、测试集、验证集均经过长度过滤的操作。

观察可得，训练数据的数据量大小对于模型训练的效果具有很大的影响，例如在相同的测试集上，IWSLT17德英训练出的BLEU值是13.51，而WMT14训练出的BLEU值是31.01。同时，相同语种不同方向的训练对于BLEU值也有影响，英语和法语之间的BLEU值明显高于其他语种。

另外，单独的IWSLT英德口语领域的数据在测试集上的BLEU仅有11.86和13.51，当时加入了大量英德在新闻的数据集后，BLEU变为31.05，相较于单独的WMT14英德数据集训练BLEU值略微降低，但是对于提升英德在口语领域的表现却是巨大的。

## 5.3 语言零资源翻译测试

对整合后的系统进行测试，验证德语和法语之间是否可以进行正常的翻译任务。

### 5.1.1 测试方案

（1）测试对象

多语言多领域机器翻译系统Web服务器。

（2）测试目的

通过模拟用户操作，测试系统前端是否调用两次接口并进行了德法之间的翻译。

（3）测试过程

在本地服务器运行系统，加载训练好的翻译模型，之后打开系统的HTML页面，通过模拟用户的操作，进行检测。

（4）评测标准

正确返回德法之间的翻译结果。

### 5.1.2 结果及评价

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。

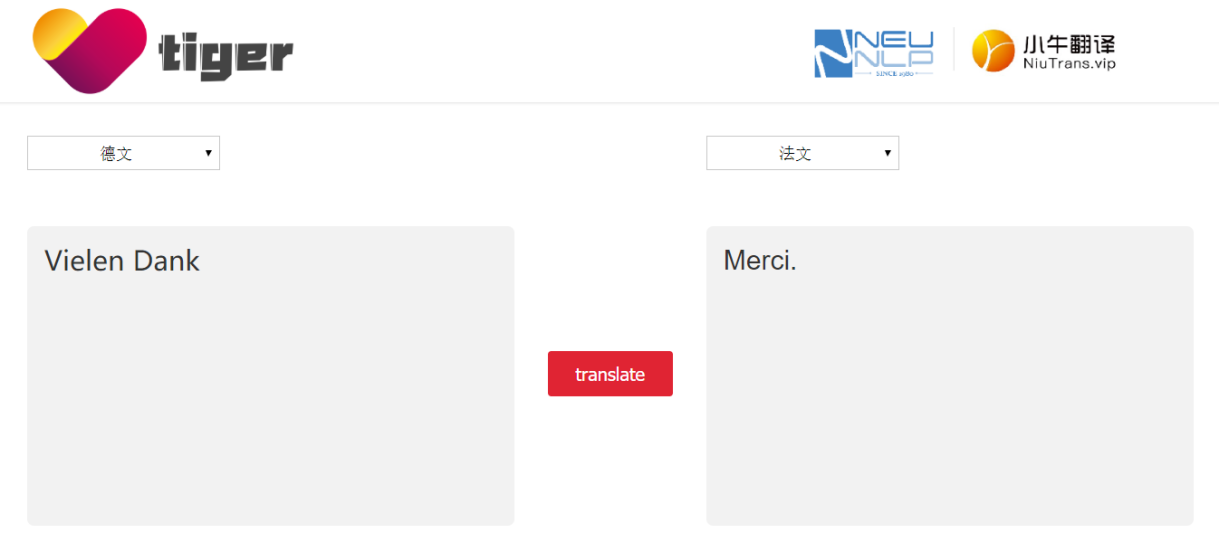


图6.6 德语翻译到法语

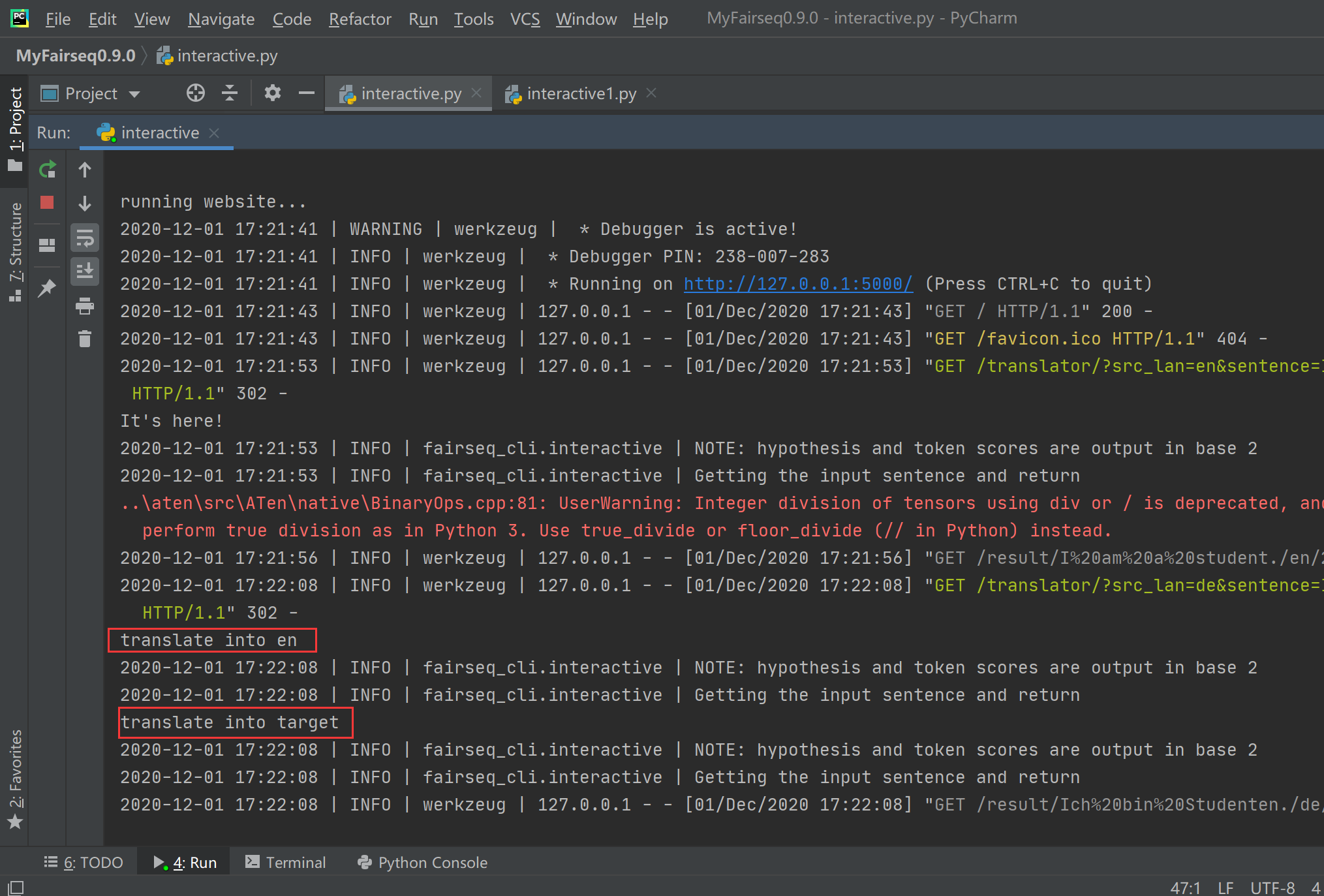


图6.7 后台输出提示