分类号_____ 密级 _____

UDC _____

学 位 论 文

基于神经机器翻译的蒙汉翻译系统的研究与实现

作 者 姓 名 ：阿敏巴雅尔

指 导 教 师 ：肖桐 副教授

东北大学计算机科学与工程学院

申请学位级别：硕士 学 科 类 别 ：专业学位

学科专业名称：计算机技术

论文提交日期：2018 年 12 月 论 文 答 辩 日 期 ：2018 年 12 月

学位授予日期：2019 年 1 月 答辩委员会主席：袁野

评 阅 人 ：张俐 战学刚

东 北 大 学

2018 年 12 月

A Thesis in Computer Technology

Research and Implementation of Neural Machine Translation Based Mongolian-Chinese Translation System

By Ambyer Han

Supervisor: Associate Professor Xiao Tong

Northeastern University

December 2018

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：

日　　期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □　　一年□　　一年半□　　两年☑

学位论文作者签名：　　导师签名：

签字日期：　　签字日期：

摘 要

随着技术的发展，机器翻译系统的性能得到了长足的进步，由传统的基于统计的机器翻译发展到了目前备受瞩目的基于神经网络的机器翻译。现如今很多公司都推出了自己的基于神经网络的机器翻译系统，而且支持了很多语言对之间的相互翻译，这给很多企业和个人用户带来了很大的便利性。

虽然国内很多公司的机器翻译系统极大的便利了客户，但是主要还是集中在汉语、俄语和英语等语言之间相互翻译的系统。而中国有 55 个少数民族，并且其中有一部分民族仍使用自己的语言和文字。所以研究少数民族语言与汉语之间相互翻译的翻译系统有着重要的意义。本文主要研究蒙语到汉语的翻译系统，以及实现蒙汉翻译系统。

通常情况下，训练一个性能较好的神经机器翻译系统往往需要大量的双语数据。本文所使用的蒙汉双语数据的特点即为数据量稀少，获取或标注双语数据成本较高，而获取大量单语数据成本低且容易。鉴于此，为使模型性能提高，本文分别在搭建神经机器翻译系统的三个流程——预处理、模型训练以及解码译文的过程中做一些改进。首先在预处理过程中使用高频词与子词混合的分词方法使双语数据保留语义的同时减少数据稀缺的问题；第二，通过反向翻译单语语料生成大量的伪数据，并使用预训练以及微调训练混合的方法训练模型；最后，在解码过程中使用神经语言模型与解码端融合的方法进行解码，以达到输出更流畅的译文的目的。

本文的实验在数据集 CWMT-2018 蒙汉平行双语数据上进行，该数据集有 26 万句对，并在 CWMT-2018 蒙汉校验集上进行检验。实验结果表明，本文提出的预处理方法带来 0.43 的 BLEU-5 值的提升，使用的训练方法则有 3.05 的 BLEU-5 值的提升，使用的模型融合的方法最终有 1.07 的 BLEU-5 值的提升。

关键词：机器翻译；神经网络；自注意力机制；伪数据

Abstract

With the development of technology, the performance of machine translation has made great progress. It went from the traditional statistical machine translation to neural machine translation. Nowadays, many companies have deployed their own neural machine translation system, and supported mutual translation between many language pairs, which brings convenience to many enterprises and individual users.

Although the machine translation systems of many companies in China greatly facilitates customers, it mainly focuses on systems that translate among Chinese, Russian and English. But there are 55 ethnic minorities in China, and some of them still use their own language and characters. Therefore, it is of great significance to study the translation system between the minority languages and Chinese. This thesis mainly studied the Mongolian to Chinese translation system, and implemented the translation system.

In general, a large amount of bilingual data is needed to train a neural machine translation system with good performance. The Mongolian-Chinese bilingual data is characterized by the scarcity of resource, and the cost of acquiring or labeling bilingual data is high, and obtaining a large amount of monolingual data usually is low in cost and always easy to acquire. In view of this, for improving the performance of the translation system, we have made some improvements in the three stages of deploying a neural machine translation system, and the stages contain the preprocessing the corpus, training the model and decoding. Firstly, the word segmentation method of mixing high-frequency words and sub-words have been used in the preprocessing stage to reduce the problem of data scarcity while preserving the semantics of the data. Secondly, a large amount of pseudo-data has been generated by back-translation using monolingual corpus, and then the model has been trained by hybrid method of pre-training and fine-tuning. Finally, in the decoding stage, the neural language model and decoder fusion method has been used to decode the source sentences to produce more smoother translation.

The experiments have been carried out on CWMT2018 Mongolian-Chinese corpus that contains 260 thousand sentence pairs, and the performance of the model has been tested on CWMT-2018 valid set. The results of the experiments have showd that the preprocessing

method has brought 0.43 BLEU-5 score improvement, and the training method can increase the value of BLEU-5 by 3.05, the method of model fusion can improve the value of BLEU-5 by 1.07.

Key words: machine translation; neural network; self-attention mechanism; pseudo-data

目 录

第 1 章 绪 论

1.1 研究背景

机器翻译（Machine Translation 或简写为 MT）属于计算机语言学的范畴，是计算机语言学的一个分支，并且具有重要的科学研究价值。所谓机器翻译，简单来说就是利用计算机程序将一段文本从一种自然语言（即源语言）翻译成另一种自然语言（即目标语言）的过程。随着社会以及互联网的普及和发展，机器翻译开始有越来越多的需求。尤其最近几年由于神经机器翻译的出现，带来的翻译性能的提升，更是使其研究价值和使用需求推至又一个高峰。目前，国外国内如雨后春笋般有越来越多的企业开始做机器翻译相关的业务，他们或面向企业、政府机关单位，或面向普通用户。不仅如此，与机器翻译相关的产品也越来越多样化：在线上句子、篇章级别的翻译服务，与语音技术相结合的同声传译服务，与拍照结合的拍照翻译服务，以及手机端离线的翻译服务更是可以在不使用网络的情况下就可以完成文本或语音的翻译等服务。这些都很好的体现了现如今机器翻译的巨大研究以及商用价值、以及应用前景。

机器翻译系统可划分为基于规则的机器翻译[1-3]（Rule-Based Machine Translation 或 RBMT）和基于语料库的机器翻译[4-7]（Corpus-Based Machine Translation 或 CBMT）两大类。基于语料库的机器翻译又可以分为基于实例的机器翻译[8-10]（Example-Based Machine Translation 或 EBMT）、统计机器翻译[11-18]（Statistical Machine Translation 或 SMT）和最近几年迅速发展起来的神经机器翻译[19-22]（Neural Machine Translation 或 NMT）。从机器翻译问题的正式提出到 90 年代，机器翻译研究一直使用基于规则的方法，该方法在特定的条件下能够取得比较好的翻译效果。但是随着研究的深入，该方法遇到了一些难以突破的瓶颈问题。比如，人工书写的规则覆盖度有限、规则数量增加导致的冲突、规则的质量和效果依赖于语言学家的知识和经验，规则获取的人工成本较高而且维护大规模规则往往比较困难。虽然随后兴起的基于实例的方法可以一定程度上缓解以上问题，但是问题仍然没有得到根本解决。机器翻译的突破性进展开始于上世纪九十年代初。IBM 的研究人员提出了基于词的五个翻译模型，从此开启了统计机器翻译的时代。这种方法完全抛弃了对人工书写规则的依赖，而是把翻译问题看作是搜索翻译概率最大的译文的问题。在众多统计机器翻译模型中基于短语的模型（Phrase-based translation model）是最为成熟的模型。相比传统的基于统计的机器翻译，神经机器翻译（Neural Machine Translation 或 NMT）是一种使用深度学习神经网络获取自然语言之间映射关系的方法。

神经机器翻译的非线性映射不同于线性的统计机器翻译模型，使用了连接编码器和解码器的状态向量来描述语义等价关系。神经机器翻译凭借其较高的译文质量吸引了很多学者使用该方法对机器翻译任务进行研究[23]。

目前，在国内蒙古族有 600 万人，分布于内蒙古、青海以及东北三省等地，其中有很大一部分人依然以蒙语为母语并接受着蒙语教学，使用的课本、习题、甚至是中考、高考都是用蒙语文卷子。不仅仅在教育领域，在内蒙古法院、政府机关单位等机构都会使用蒙语或者使用蒙汉双语的文件。所以说，蒙-汉机器翻译系统在国内市场上还是有很大的需求。比如，法律文件、政府文献需要翻译，法庭、课堂的同声传译中的翻译系统以及试卷试题的翻译等等应用。但是，国内却有很少的企业或单位在蒙-汉机器翻译系统上做一些研究甚至是系统的实现。现在市面上，企业通常使用神经机器翻译系统作为线上系统的模型，而神经机器翻译模型通常需要大量的平行语料进行模型的训练。例如，线上提供翻译服务的模型通常在千万甚至上亿级别的平行双语语料进行模型的训练。而很多语种之间以及特定领域的双语语料则可能只有很少的双语语料，而且标注双语语料又需要很大的人工成本，这就直接导致很多语种之间或者特定领域的翻译任务很难得到效果好的机器翻译系统。而蒙语-汉语平行语料就属于这种稀缺资源，这也是导致目前市面上难有蒙-汉机器翻译系统和对应的研究的一个重要原因。因此，使用少量的蒙-汉双语语料的前提下怎样提高蒙-汉神经机器翻译系统的性能，并使其应用于实际市场中具有重要的研究和商业意义。

1.2 研究内容

最近几年，人工神经网络的研究工作不断的深入，并且已经取得了很大的进展，其在计算机视觉、自然语言处理等人工智能领域一一被深度学习攻破。在机器翻译任务中，神经机器翻译更是在自动评价表现和译文流畅性、可读性等方面都超过了传统的基于统计的机器翻译模型，是当前最热门的研究课题之一。目前，越来越多的企业开始搭建自己的神经机器翻译系统，并提供给客户使用。

然而，现在大部分的人工神经网络模型的学习都采用有监督的学习方法。即，这些模型想要达到比较理想的效果，都需要大量的、有标注的数据，这需要耗费大量的人力、物力和财力。所以，目前市面上大部分神经机器翻译系统都是那些有大量平行双语数据的语言对的翻译系统服务。而一些有较高需求，但是只有少量双语平行数据的语言对来说却没有太让人满意的翻译系统供客户使用。

故本课题主要使用基于自注意力（Self-attention）机制[24-25]的神经机器翻译系统来

研究并搭建蒙语-汉语神经机器翻译系统。谷歌提出的基于自注意力的神经机器翻译凭借其优异的性能表现以及较快的训练速度、收敛速度以及有效缓解了饱受诟病的梯度爆炸和消失问题，故在本研究课题中使用基于自注意力的神经机器翻译模型作为翻译模型。基于自注意力的机器翻译模型使用编码器-解码器结构，其中编码器和解码器分别使用基于自注意力的语言模型。该语言模型结构在训练阶段可以同时计算不同位置的信息，摒弃了基于循环神经网络[26-28](Recurrent Neural Networks 或 RNNs)所固有的当前状态要依赖上一时刻状态的特点，提高模型计算的并行性的同时也缩短了不同位置的词之间信息传输的距离。

在使用基于自注意力的神经机器翻译模型结构的基础上，我们使用了 26 万句对蒙语-汉语双语平行语料，以及 700 万中文单语语料，对于训练一个神经机器翻译模型来说，26 万句对属于数据稀缺的资源。为了在数据稀缺的情况下提高模型性能，本课题使用以下三个方面进行课题研究：

（1）蒙语语料预处理方法：蒙语文是一种黏着性语言，是由词根后接多种词缀构成蒙文文字。对于蒙语文来说，不同的后缀以及多个后缀的拼接可能导致文字时态、意义上的变化，是一种形态丰富的语言。对于这种情况，我们通过将高频、并长度较短的字模拟词根，并对词缀进行分子词的方法进行预处理，以达到减少语料多样性的目的；

（2）基于迭代生成伪数据的模型训练方法：蒙语-汉语可使用的双语平行语料只有 26 万句对，对于神经机器翻译模型来说，二十万级别的语料不足以生成可使用的机器翻译系统。本课题中通过使用大量中文单语语料迭代生成伪数据，通过迭代，可使生成的伪数据质量越来越高。通过此方法，可弥补平行数据不足的问题，并以此来提高模型性能；

（3）Decoder 与 LM 融合解码方法：训练语料的稀缺会导致模型训练不够充分进而影响模型性能。鉴于我们拥有大量单语语料可利用，故尝试使用单语语料训练语言模型，并在解码时使用神经机器翻译解码器与语言模型融合的方式进行解码。目的在于使用语言模型辅助解码器进行搜索。由于语言模型通过大量数据训练，相比只有少量训练数据训练的解码器表示能力更强大，可辅助解码。

以上三种方法，分别从数据的预处理方面使得在有限的语料上经过预处理以减少数据多样性的出现，使模型更容易训练；在训练方法上利用单语语料增加数据的方式训练模型，以使训练数据增加的方法提升模型性能；以及使用语言模型辅助解码器促使输出更高质量的译文的方法提升模型性能。通过这三种方法，可以有效的提高模型性能表现。

1.3 论文组织结构

本文的研究重点在于通过预处理的方法以及迭代训练的方法提高蒙汉神经机器翻译系统的性能。首先，我们通过对高质量语料进行预处理，通过此方法减少语料的多样性，使模型能更容易的学习；其次，我们通过增加伪数据的方法增加平行语料的量，以模拟大量平行双语语料，最终达到提高蒙汉机器翻译系统性能的目的。

本文主要内容如下：

第 1 章主要介绍了论文的研究背景以及研究内容。

第 2 章主要介绍了论文研究内容中所涉及到的相关技术，主要围绕语料预处理的一些方法以及神经机器翻译模型的结构、使用的相关技术以及参数迁移进行介绍。其中，语料预处理的方法有分词、编码对分词、Wordpiece[29]分词等；神经机器翻译模型使用的编码器-解码器框架、注意力机制以及基于循环神经网络的模型（RNNs）、基于卷积神经网络[30-33]（CNNs）的模型以及基于自注意力（Self-attention）的模型等。

第 3 章主要介绍了不同语料预处理方法的性能对比以及使用字、子词混合的预处理方法在系统中的使用。并通过实验对比进行讨论和分析。

第 4 章主要介绍了通过迭代的反向翻译的方法来生成大量伪平行数据的方法来提高模型性能。并通过实验探讨不同迭代次数、伪数据量对于模型性能提升的影响。

第 5 章主要介绍利用语言模型辅助神经机器翻译模型解码过程的方法。

第 6 章介绍系统的结构、前端界面等内容。

第 7 章对本文的研究工作进行总结，同时对未来的工作进行了讨论。

第 2 章 相关技术概述

2.1 机器翻译流程与语料预处理方法介绍

神经机器翻译系统的整个流程，我们可以分两大部分——训练和解码。我们首先会使用标注的数据训练好模型，使其可以在训练数据的领域有较强的翻译能力。而所谓的解码过程就是只给定源语言句子或数据的情况下去翻译源语句子并生成对应的译文结果。

对于模型来说，不管在训练阶段还是解码阶段，对于输入数据都有一些预处理过程。通过预处理我们可以给后续的模型训练提供分好词的且格式恰当的双语语料。分好词的语料对模型的训练、系统的性能有着重要的作用。通常，我们可以将句子分成字、词、子词和字符等多种颗粒度的形态。同样不同颗粒度的语料对于模型的训练有着不同的影响。

2.1.1 神经机器翻译系统流程

搭建一个神经机器翻译系统，我们首先要对原始数据进行预处理。预处理包括过滤筛选数据（有乱码过滤，长度比过滤，长句子过滤，去除重复句子）以及分词的过程。其中训练模型时会进行过滤筛选过程和分词过程，而解码时通常只会进行对源语言进行分词过程，其详细的流程图如图 2.1 训练过程所示。

经过分词处理之后的数据则作为系统的输入进行模型的训练。神经机器翻译模型训练时通常以 batch 为一个单位进行一次向前计算后反向计算并根据梯度更新权重。一个 batch 通常有若干个句子或若干个词，而整个训练数据集有若干个 batch。当模型从头到尾学习过整个训练集可理解为学习了一轮。模型一般情况下都会训练 10 轮以上的次数才能达到较好的模型性能。

神经机器翻译解码过程相对训练过程比较简单，没有反向传播并更新权重的过程，只是向前计算，并预测译文。在解码时搜索过程有贪婪搜索以及束搜索，且束搜索过程具有更好的性能表现，是学者和产业界公用的方法。具体流程可以参考图 2.1 中解码过程。解码器中，模型体与训练过程中的模型结构一样，可以读取训练结束后保存的网络权重。而搜索器则会根据不同情况有不同的搜索方式，通常情况下是束搜索或者越来越多的学者提出的更多、更高效的搜索方法。

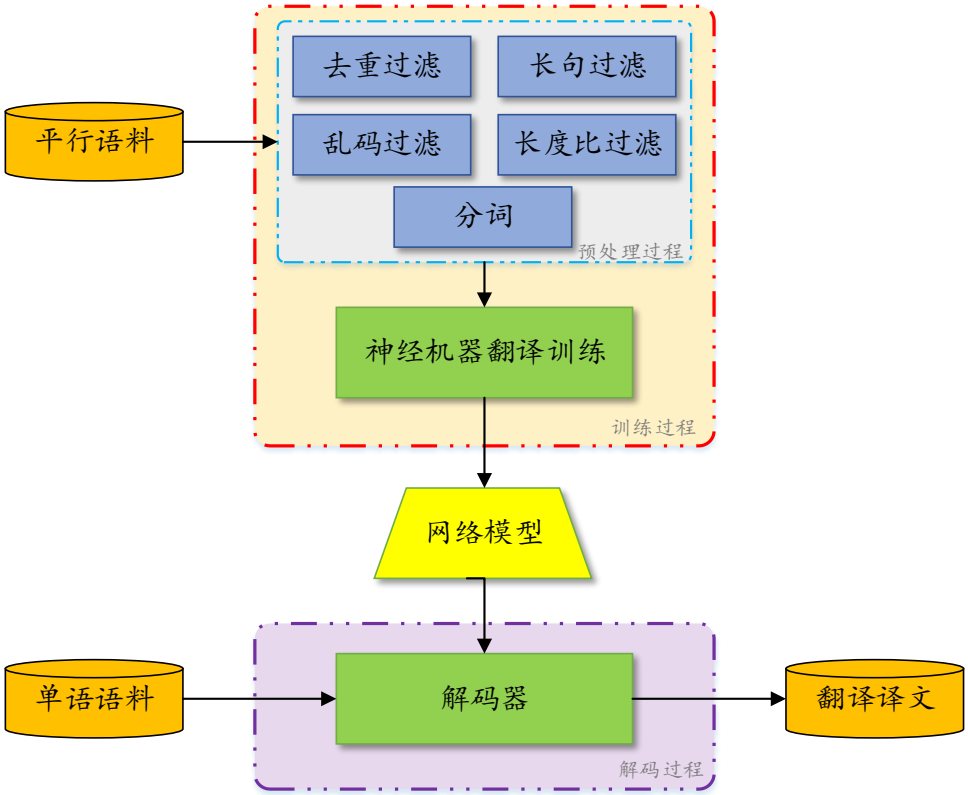

图 2.1 神经机器翻译系统训练与解码流程

Fig. 2.1 The training and decoding phase of neural machine translation system

2.1.2 基于 BPE 的分词方法

一些稀有的词会产生 OOV 问题，导致模型性能受到一定的影响，Sennrich 等人提出一种字节对编码[34]（Byte pair encoding 或 BPE）的方式对数据进行预处理。字节对编码技术最早是应用于数据压缩领域，是一种简单的数据压缩形式，算法描述为字符串里频率最常见的一对字符被一个没有在这个字符串中出现过的字符来代替的层层迭代过程。在论文中，这种算法应用于分词时，首先通过学习得到整个语料中相邻出现的字节对编码词典。这个词典中的每一条都是出现频率最高的字节对，词典大小由一个超参 operations（或称为 merges）结合次数来决定。然后再将此字节对编码词典应用于原始语料中，将原始语料中出现频率高的且存储在字节对词典里的字节对结合在一起，而不在词典里的字节对则会被分开。

通过 BPE 预处理，并在合适的合并次数的情况下，语料可以完全避免稀缺词的问题，使训练语料中没有 UNK 的出现。虽然分的词没有直接语义上的意义，但通过分子词的数据训练模型往往能得到更好的效果。且由于语料中从未出现过 UNK，所以一般情况下输出的译文中也不会有 UNK 的出现。

2.1.3 基于 Word Piece 的分词方法

与字节对编码一样，词片（Word Piece）分词方法也属于一种子词分词方法。字节对编码主要考虑出现频率最高的字节对，从而使频率高的字节对合并。而词片方法主要是将句子中的字符串先根据字以及标点符号，控制符分割。这样第一步时词片方法就将字和符号分割开，并将其中的空格使用下划线“_”代替。而字节对编码通常是在将语料分词之后再进行 BPE 分词。故一般情况下词片分词方法不需要像 BPE 一样先将句子分词，而是将整个句子进行词片分词。经过第一步之后，词片分词方法再将字内的字符进行合并，而合并方式不像 BPE 一样两两合并，而是将整个词以长度为 1 到长度为词长的子词分割。例如英文单词中的“word”，则将其分为子词“w”，“wo”，“or”，“rd”，“wor”，“ord”，“word”等。再根据语料中所有子词中频率高于某个特定阈值的判断为可合并。通过指定的迭代，就可以完成词片分词的训练过程。同样，将训练过的词片模型再应用于原始语料中则可将原始语料进行分子词。

2.2 神经机器翻译框架与模型结构介绍

传统的统计机器翻译是非限定领域机器翻译中性能较佳的一种翻译模型。其基本思想是通过对大量的平行语料进行统计分析，构建统计翻译模型，进而使用该翻译模型进行翻译。与统计机器翻译模型不同，神经机器翻译（Neural Machine Translation）是指直接采用深度学习的人工神经网络，并以端到端（end-to-end）的方式对翻译模型建模的方法。

早在上世纪九十年代初，西班牙阿利坎特大学的 Forcada 和 Neco 提出“编码器-解码器”框架进行翻译转换工作。在 2013 年，英国牛津大学 Kalchbrenner 和 Blunsom 提出了以序列表示为基础的神经机器翻译。在 2014 年，加拿大蒙特利尔大学的 Cho 等人以及谷歌公司 Sutskever 等人分别对该神经机器翻译模型进行了完善。Bahdanau 等人则在 Cho 等人的工作基础上增加了注意力机制，而正是注意力机制，将神经机器翻译模型带入到了一个新的时代。

2.2.1 编码器-解码器框架

神经网络（Neural Network）模型最初在图像领域有着举足轻重的地位，且性能优异。而对于序列形式的自然语言处理领域并没有能够很好的被利用到。最初 Bengio 等人使用前馈神经网络（Feedforward Neural Networks 或 FNNs）来实现 *n*-元语言模型[35]，

正式将神经网络带入自然语言处理领域。而前馈神经网络由于其固有的特性，无法完成不定长度的序列问题。而机器翻译中源语言句子和目标语言句子都是长度不定的序列到序列的问题。

为了解决这一问题 Cho 等人在神经网络的基础上最早提出了“编码器-解码器”框架。这个框架将翻译过程建模为两个阶段：1）首先使用编码器（Encoder）将源语言句子读入并对其编码，将句子中的序列信息整合到一个固定维度的向量中；2）然后使用解码器（Decoder）对编码器得到的带有源语言序列信息的向量进行解析，并一个词一个词的输出，最终得到相对于源语言的译文序列。在 Cho 等人提出的方法中，编码器和解码器使用的神经网络都为循环神经网络。后续的科研工作者更是将循环神经网络（RNNs）以及其变种（如 GRU，LSTM 等）使用到编码器-解码器框架中。对于一个长度为 n 的源语言序列$X = \{x_0, x_1, \dots, x_n\}$，基于循环神经网络的编码器依次输入$x_t$,并通过循环神经网络单元输出当前的状态$c_t$并读取下一时刻的输入，直至将所有的序列读取完。我们将状态c_t理解为状态向量，亦称为隐层状态。隐层状态包含着从开始到当前位置的所有信息，可以理解为神经网络将序列的信息承载在隐层状态中进行传递。在时间步 t 时的隐层状态h_t的更新方式如公式 2.1 所示。

$$h_t = f(h_{t-1}, x_t) \tag{2.1}$$

其中函数 f 为非线性变换函数，可表示循环神经网络的循环单元（RNN、GRU 或 LSTM 等）。通过上述公式我们可以理解到，对于时间步 t 来说，其隐层状态h_t包含着从位置 0 到 t 时刻的所有序列的信息。那么序列的最后一个时间步的隐层状态则包含了整个序列的所有信息。我们可以将其理解为编码器将输入的源语言序列编码为一个固定的向量 c（即最后一个时间步的隐层状态），而这个向量包含了整个序列的信息，表示编码器将源语言编码为一个语义向量，并将其提供给解码器使用。

在解码阶段，解码器将从编码器得到的上下文向量 c 进行解析。解码过程与编码过程相似，也是每一个时间步生成一个隐层状态h_t。对于长度为 m 的目标语言序列$Y = \{y_0, y_1, \dots, y_m\}$，解码器隐层状态的更新方式如公式 2.2 所示。

$$h_t = f(h_{t-1}, y_{t-1}, c) \tag{2.2}$$

其中函数 f 为非线性变换函数。解码器经过 m 次循环，即可得到译文序列。整个“编码器-解码器”框架的神经机器翻译模型计算得到译文的公式如公式 2.3 表示。

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = g(h_t, y_{t-1}, c) \tag{2.3}$$

上式中的函数g表示非线性变换函数，通常为归一化函数（如 softmax 等）。在整个神经机器翻译的过程中，编码器第一个时间步上一个状态h_0定义为全为 0 的向量，表示

在编码前不具有任何信息。在解码器解码的第一个时间步的输入y_0为指定的起始标识符，通常使用字符串<sos>（start of the sentence）表示，而其前一个状态h_0则为编码器的输出向量 c，表示源语言的语义向量。解码器根据初始输入y_0，初始状态 c 开始预测目标语言句子的每一个词。每一个前一时刻的输入y_{t-1}和前一时刻的隐层状态h_{t-1}以及编码器输出的上下文向量 c 通过循环神经网络单元预测出当前时刻的输出y_t，以此类推，最终遇到结束标识符则停止循环，并结束搜索过程，输出最终的译文，表示整个翻译过程结束。其中结束标识符通常使用字符串<eos>(end of sentence)表示。而输出目标语言序列的公式如公式 2.4 所示。

$$Y = \text{argmax}_Y \prod_{i=1}^{m} Pr(y_i|\{y_0, \dots, y_{i-1}\}, X) \tag{2.4}$$

公式中$Pr(y_i|\{y_0, \dots, y_{i-1}\}, X)$表示着在 i 时刻的输出词的概率与源语言以及目标序列从开始到 i-1 时刻的所有状态有关。整个网络结构图如图 2.2 所示。

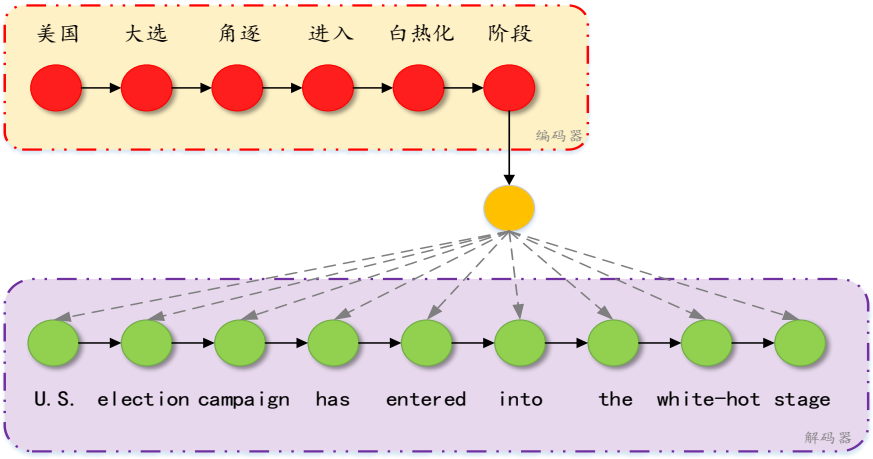

图 2.2 Cho 等人提出的“编码器-解码器”框架

Fig. 2.2 Encoder-Decoder framework by Cho et al.

之后 Sutskever 等人提出的“编码器-解码器”框架，则在 Cho 等人的基础上又进一步完善，使得隐层状态的更新方式从$h_t = f(h_{t-1}, y_{t-1}, c)$变为如公式 2.5 所示。

$$h_t = f(h_{t-1}, y_{t-1}) \tag{2.5}$$

即，每个当前隐层状态的值只跟上一个输出以及上一个隐层状态相关。而编码器输出的上下文向量只在解码器第一时刻使用，其结构图如图 2.3 所示。

在训练过程中，神经网络可通过反向传播可以更新网络内部的参数，并使用随机梯度下降法等优化方法进行更新。由于计算时完全使用了连续空间的表示，相比传统统计机器翻译的离散空间表示，模型的表示能力大大增强。以此来完成对机器翻译模型的训练以及解码过程。我们可以看到整个过程的输入和输出就是我们输入的双语平行语料，相对于传统的统计机器翻译，神经机器翻译可以直接端到端的训练和解码，简化了整个

机器翻译过程并使该领域的入门门槛降低了。

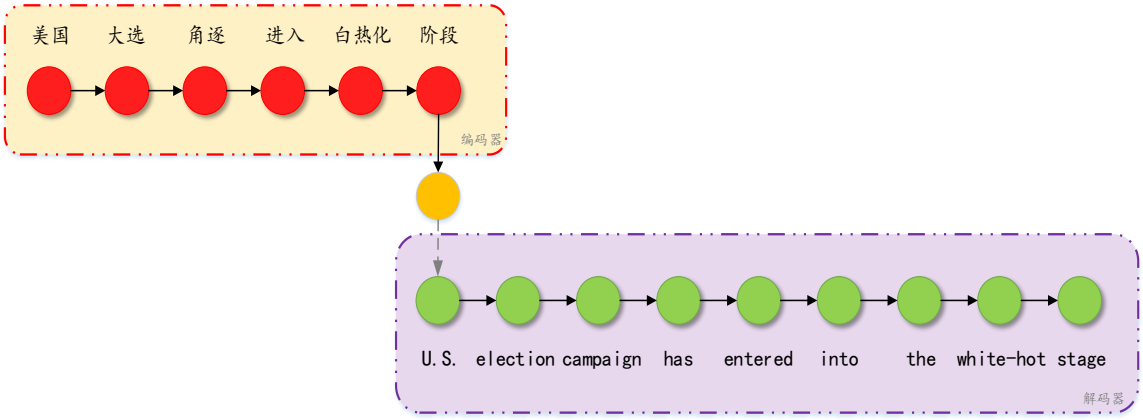

图 2.3 Sutskever 等人提出的“编码器-解码器”框架

Fig. 2.3 Encoder-Decoder framework by Sutskever et al.

2.2.2 注意力机制

在将注意力机制引入到神经机器翻译之前，基于“编码器-解码器”的框架的神经机器翻译系统翻译性能并没有想象的那么好。将源语言序列的所有信息都融合到一个固定维度的向量会导致信息丢失或无法有效的解析的情况。通常，源语言序列最开头的信息经过长距离的传递逐渐被削弱，导致解码器在解码时无法高效的解析上下文向量[36]，进而导致译文性能差的情况。

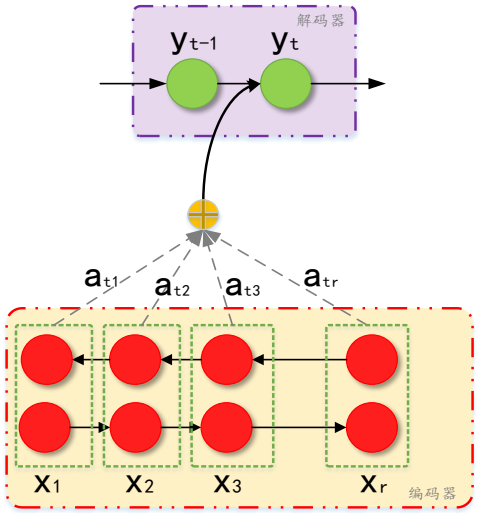

图 2.4 Bahdanau 提出的注意力机制

Fig. 2.4 Attention mechanism by Bahdanau et al.

为了解决这个问题，Bahdanau 等人于 2014 年提出在“编码器-解码器”框架中引入注意力机制的方法。该方法核心思想是想每输出一个译文结果时，都要与源语言每一个

时刻的隐层状态作用。这就保证了解码器中每一个词输出时都能将源语言中的每个状态考虑进来，并根据源语言中不同状态的值来决定当前时刻的输出。如图 2.4 表示提出的注意力机制。

从图中我们可以看到，不同于 Cho 等人提出的框架，此框架的上下文向量 c 不再是编码器中最后一个时间步的状态，而是编码器中每个时间步的隐层状态都有参与。随后的 Luong 等人提出了更高效的注意力机制[37]，并应用于“编码器-解码器”框架中。他们提出了与 Bahdanau 等人提出的注意力机制类似的全局注意力机制（global attention）。在全局注意力机制中，解码器当前时间步的隐层状态不再与上一个隐层状态直接相关，即将原有的公式 2.5 改为如公式 2.6 所示。

$$\tilde{h}_t = \tanh(W_c[c_t; h_t]) \tag{2.6}$$

公式中$\tilde{h}_t$表示当前时间步的经过注意力计算之后的状态，从公式中可以看出，它不再与h_{t-1}直接相关，而是与当前时间步的隐层状态h_t直接相关。从公式可以得出，当前时间步的输出与编码端输出的所有时间步的状态都需要进行计算，全局注意力机制的结构图如图 2.5 所示。

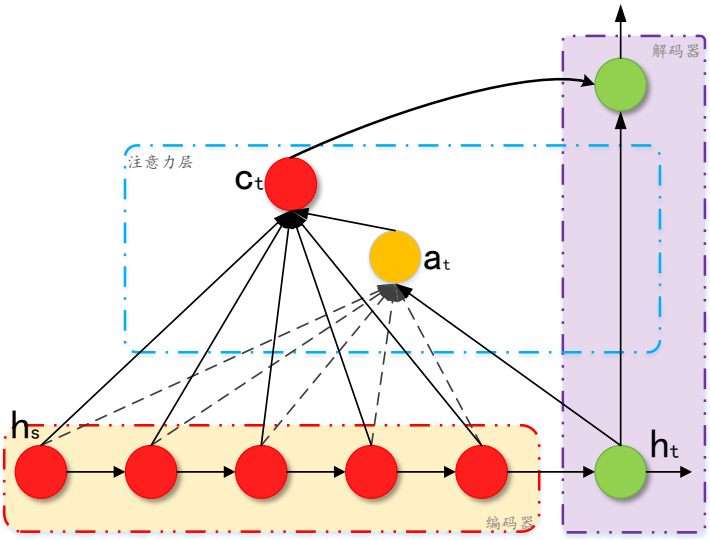

图 2.5 全局注意力机制

Fig. 2.5 Global attention mechanism

但是，通常某个时间步的输出并不与编码端所有时间步的状态都有直接的联系，或者影响很小。这就容易导致一些关系不大的编码端时间步状态会影响当前解码端输出的结果，且消耗计算资源。鉴于此，Luong 等人提出的第二种方法，即局部注意力机制很好的解决了该问题。局部注意力机制在每一次解码端的输出只根据一部分编码端隐层状态来计算。在局部注意力机制中使用一个长度大小为 $2D$ 的窗口来框取局部需要注意的

隐层状态信息。由于只需要部分的隐层状态，故更少消耗计算资源。那么下一个需要解决的问题则是求得一个位置p_t表示定位这个局部窗口，使区间$[p_t - D, p_t + D]$的局部信息作为需要注意的对象。论文中提出的方法中有两种方法来解决这个问题：1）使用当前解码时间步，即使得$p_t = t$；2）使用预测的方式求出p_t，如公式 2.7 所示。

$$p_t = S \cdot \text{sigmoid}(v_p^T \cdot \tanh(W_p h_t)) \tag{2.7}$$

其中W_p和v_p为可训练的参数，S 是源语言序列长度。由于 sigmoid 函数输出结果在 0~1 区间，故乘上源语言长度则$p_t \in [0, S]$。通过上述计算，模型可以在解码端 t 时刻时求出与其对应的编码端相关的区域，之后再根据上述公式计算相关程度。在 Luong 等人提出的方法中e_{tj}的计算方法有三种形式：1）点积运算；2）通用运算；3）级联运算。

我们对比可以看到，Loung 等人提出的注意力机制计算时只用当前时间步的隐层状态，即计算路径为$h_t \rightarrow a_t \rightarrow c_t \rightarrow \tilde{h}_t$，计算和实现时都相对简单；对应的 Bahdanau 等人提出的注意力机制则使用解码端上一个隐层状态计算当前隐层状态，即计算路径为$h_{t-1} \rightarrow a_t \rightarrow c_t \rightarrow h_t$，在实现上相对困难。并且 Luong 等人不仅提出了全局注意力机制，局部注意力机制等，而且还将计算注意力的方式扩展为三种，即点积运算、通用运算以及级联运算等方法。

2.2.3 基于循环神经网络的机器翻译

要了解基于循环神经网络的机器翻译模型，我们需要先了解循环神经网络（Recurrent Neural Network 或 RNNs）。不同于前馈神经网络和卷积神经网络，由于循环神经网络有时间步的概念，所以可以对任意长度且变长的序列进行建模。

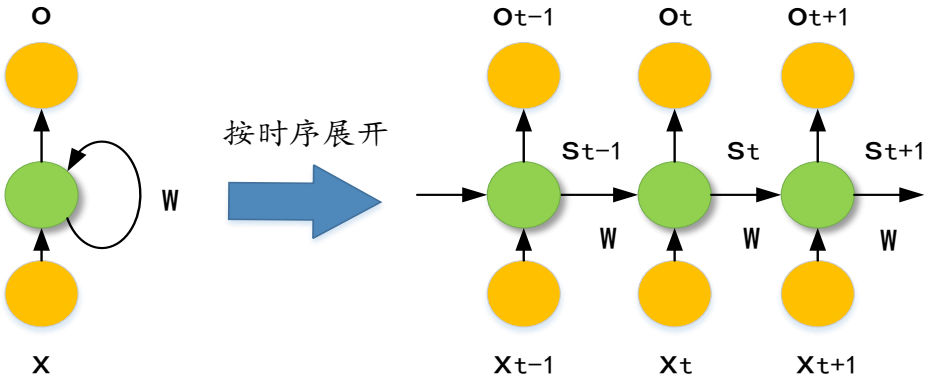

图 2.6 循环神经网络

Fig. 2.6 Recurrent neural networks

最简单的循环神经网络单元有一个权重 W，输入为输入向量 X 和前一个时刻的隐层状态向量 S（即上一刻的输出）以及输出当前时刻的状态 O（即当前时刻的隐层状态）。

其更新方式为每一个时间步都要根据输入 X、上一时刻输出 S 以及隐层权重 W 来计算当前时刻的输出，将当前时刻的输出作为下一时刻输入的隐层状态，直至序列中所有的输入依次计算完毕则循环结束，其结构图如图 2.6 所示。

上图中，左半部分表示了隐藏层中数据传递的方式，且隐藏层有权重矩阵 W，而右侧的图表示将左侧的图按时序展开之后的效果，可以清楚的了解到每一个时序中的输入和输出的值。具体使用公式来表示，则如公式 2.8 所示。

$$s_t = f(U \cdot x_t + W \cdot s_{t-1}) \tag{2.8}$$

上式中，U 和 W 是可训练的权重矩阵，f 是激活函数（通常是 sigmoid 或 tanh）。上式表示在时刻 t，当前的隐层状态s_t是由当前的输入x_t乘上对应的权重矩阵 U 加上前一时刻的隐层状态乘上对应的权重矩阵 W，再经过一个激活函数可得。

虽然简单的循环神经网络解决了序列问题，但是却不能很好的处理较长的序列，一个主要的原因就是简单的循环神经网络容易发生梯度爆炸或梯度消失。为了解决这个问题，科研学者提出了长短时记忆网络[38]（Long short term memory Network 或 LSTM）。长短时记忆网络主要引入了一个记忆向量和门控机制来保存每一个时间步有意义的信息，并从头传递至尾。记忆向量保存着序列从开始到结尾的所有有意义的信息，而网络单元中的门控机制则控制着哪些信息为有意义的信息并加入到记忆向量中，哪些信息为没有意义的信息，从而不加到记忆向量中。这些门控机制包括输入门(input gate)、遗忘门(forget gate)和输出门(output gate)。

长短时记忆网络的结构相对复杂。在后续的研究过程中有人针对这个问题提出了 LSTM 的变种，其中最具代表性的就是门限递归单元[38]（Gated recurrent unit 或 GRU），它将遗忘门和输入门结合输入到单个更新门（update gate）中，同样还将记忆向量和隐藏状态合并，并做出其他一些变化。所得的模型比标准的 LSTM 模型单元简单，且效果与 LSTM 相当，故 GRU 变得越来越流行。

基于循环神经网络的机器翻译模型通常使用 LSTM 或者 GRU 作为网络单元，简单的基于循环神经网络的机器翻译结构可参考 Cho 等人提出的框架图。通常在编码端有一个或两个双向的循环神经网络，而解码端只有一个单向的循环神经网络。在解码时，编码端读入源语并计算，而解码端则从<sos>标识符开始循环计算，直到求出<eos>标识符为止。其中，解码端输出当前词的公式如公式 2.9 所示。其中W_s和b_s分别表示输出层的权重以及偏置，是可以训练的参数。

$$p(y_t|y_{<t}, x) = \text{softmax}(W_s s_t + b_s) \tag{2.9}$$

2.2.4 基于自注意力机制的神经机器翻译

注意力机制最早在图像领域中使用，而随着 Bahdanau 等人将注意力机制引入的机器翻译领域以来，一些研究者开始研究将注意力机制作为独立可训练的神经网络。而最近，谷歌公司提出一种完全基于注意力机制的神经机器翻译模型，其结构如图 2.7。

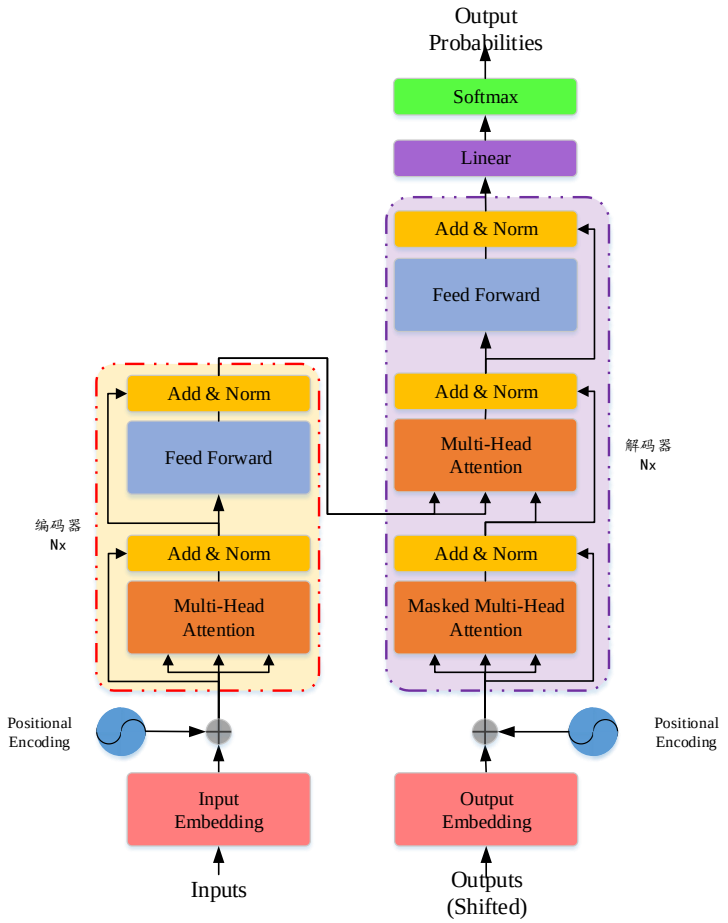

图 2.7 基于自注意力的神经机器翻译模型结构

Fig. 2.7 Structure of attention base neural machine translation

基于循环神经网络的神经机器翻译有训练速度慢的问题，而后来的基于卷积神经网络的机器翻译模型虽然在训练速度上较基于循环神经网络的模型快很多，但是性能上并不比它高出很多。而基于自注意力的神经机器翻译不管在训练速度、收敛速度还是在译文性能上都要优于其他两种神经机器翻译结构，其模型由块结构堆叠而成，编码端和解码端都包含 6 个块，块内有 2 或 3 个子层，每个子层之间又使用残差连接、dropout 以及层归一化操作。其中编码端的块包含两个子层，分别为多头注意力（Multi-head attention）层和前馈神经网络层（FFN）；而解码端的块包含三个子层，即一个多头注意力层、一个带掩码的注意力层和一个前馈神经网络层。其中带掩码的多头自注意力是在多头注意

力层上加了一个掩码，将输入的矩阵上三角置为$-\infty$，这样在使用 softmax 进行归一化时，将不应该关注的部分置零。因为在训练时为了保持训练和解码过程一致，故需要将解码端训练过程像循环神经网络一样，在解码端第 *i* 个时间步时就需要将其余的 *i+1* 到序列长度的位置进行遮盖。为了在训练过程中达到这种效果，就需要通过将矩阵上三角使用负无穷大的值替换，使得在做归一化时其概率为 0。其中注意力层使用自注意力机制，其公式如公式 2.10。

$$Attention(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{softmax}(\frac{\mathrm{Q}\mathrm{K}^{\mathrm{T}}}{\sqrt{d_k}}) \cdot \mathrm{V} \tag{2.10}$$

上式中，Q，K，V 分别表示着自注意力层的输入，在编码端 Q=K=V，而在解码端中有两种情况，分别为 Q=K=V 以及 Q≠K=V。而$\mathrm{Q} \cdot \mathrm{K}^{\mathrm{T}}$则计算了输入向量 Q 和 K 两个位置的相关性，$d_k$表示向量 Q 或 K 的维度，使用$\sqrt{d_k}$做分母可以将点积之后的结果缩放，使其值在合理的实数范围之内。而 softmax 函数则将带有相关性的矩阵进行归一化，以此来得到对齐概率，可以得到序列中每一个位置与其他位置的相关性。输出的该值就是 Q 对 K 的注意程度，再将此结果与 V 做乘积可以得到经过对齐之后的相关性矩阵。从整个计算注意力机制的过程可以看到，这个过程没有像循环神经网络一样固有的时序，而是可以并行计算，这大大加快了训练的速度，同时将信息之间的传递距离复杂度减少的 O(1)。经过自注意力层计算之后的结果再经过残差连接、dropout 以及层归一化之后进入前馈神经网络层。相较于自注意力层，前馈神经网络层相对简单，是通过两个前馈神经网络叠加而成，第一个前馈神经网络将输入向量的维度放大至原先的 4 倍，并使用 ReLU 激活函数进行激活，第二个前馈神经网络层又将增大后的向量维度缩小至输入时相同。

在上文提到的多头注意力机制，就是将输入向量 Q，K，V 分解成多个相同维度的部分，使不同部分之间做注意力计算。每一个部分称之为一个头（head），在每个对应头之间做完注意力计算之后将结果进行拼接输出最后拼接的向量，再做后处理得到该层最后的结果。

基于自注意力的神经机器翻译整个模型结构中，注意力机制被以三种方式使用，分别为：1）编码器的自注意力机制，此注意力机制针对源语言序列进行建模，对源语言序列进行信息抽取，并通过多个层最后得到表示源语言语义信息的上下文向量，其中 Q，K，V 保持一直，均表示源语言端输入的词向量信息；2）解码端自注意力，此注意力机制与源语言端注意力类似，只不过为了保证训练和解码过程一直，加上了带掩码的注意力机制。通过掩码的方式，解码端的自注意力很好的模拟了解码过程中每次输出一个词的过程。其输入的 Q，K，V 保持一直，均为目标序列的词向量信息；3）编码端-解码

端注意力，这部分注意力与基于循环神经网络以及基于卷积神经网络的神经机器翻译中的注意力机制一样，是为了保证目标端的结果对源语言对齐，并根据相关程度来进行输出当前的词。这里，K 和 V 表示源语言端输出的上下文向量，而 Q 则表示解码端自注意力的输出向量。

2.3 神经机器翻译模型训练与解码方法介绍

在开始训练神经网络的时候首先需要对网络参数，即权重矩阵进行随机初始化。显然，初始化之后的网络并不会有很好的效果，故我们需要通过训练进行参数的更新，使其对特定的任务有好的泛化性能。神经机器翻译模型所使用的函数优化方法是基于梯度的优化方法，最简单的有随机梯度下降方法（Stochastic gradient descent 或 SGD），通过该方法，我们可以更新神经网络的权重，使其损失最小。

通常当数据量足够大的时候，神经网络可以很好的完成数据拟合的任务。但是当数据量稀少，无法从少量数据拟合出较好的网络时，则系统就会出现很糟糕的性能表现。面对这个问题，也有很多学者从学习的方法上研究，使模型能够更好的在少量数据，并有大量其他领域或未标注的单语数据基础上提高模型性能，这里主要细讲使用单语语料的模型训练方法、使用参数迁移的训练方法以及了解一下多任务学习方法。

2.3.1 使用单语语料训练方法

通常情况下标注的平行双语数据所能获取的数据量非常少，而且标注数据的成本往往非常大，而未标注的单语数据的获取成本非常小，数据量往往也会非常大。面对这种情况，学者们就开始考虑，怎样将单语语料运用于只有少量平行双语数据的翻译任务上。2016 年 Sennrich 等人提出两种使用目标端单语语料来提高神经机器翻译模型的方法[39]。

其中第一个方法比较简单，将大量目标端单语语料与使用“NULL”标识符的空原语组合成平行双语数据。后将平行双语语料与带有空表示符号的平行双语语料合并为一个数据集。在网络训练过程中，如果模型遇到源语言端为空源语言，则会将编码端的参数进行固定，只更新解码端的参数。这表示同一个数据集中，根据源语言网络模型参数学习的方法不同，是一种机器翻译模型和语言模型交替训练的过程。另一种方法则是将目标端单语语料进行反向翻译，其流程如图 2.8 所示。所谓反向翻译是指将目标端语言作为源语言，源语言端作为目标语言进行模型训练，在训练好模型之后使用系统对大量目标端单语语料进行翻译解码。这样可以生成源语言端为译文输出，目标端为原始单语语料的双语数据，即只有源语言端的句子是伪数据，而目标端的句子是高质量数据。

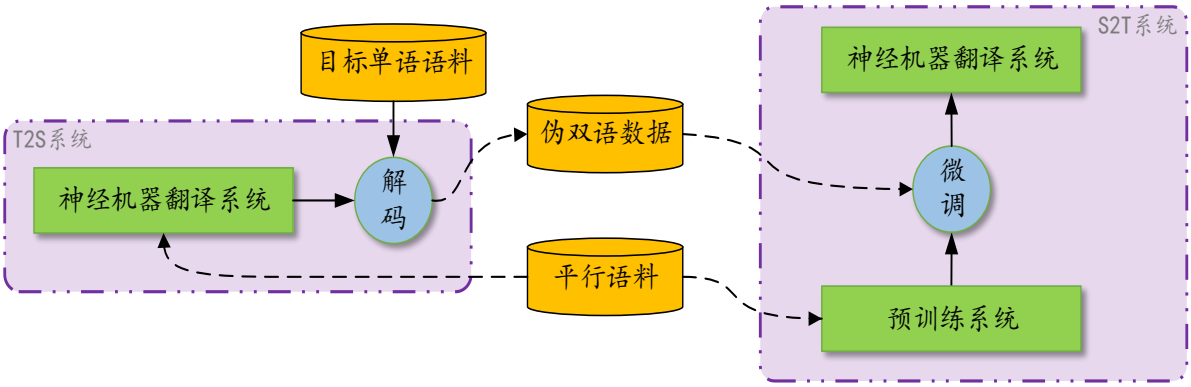

图 2.8 反向翻译训练流程

Fig. 2.8 Process of back-translation

2.3.2 参数迁移训练方法

在 2016 年，Zoph 等人提出一种迁移学习方法[40]。他们提出“编码端-解码端”的神经机器翻译框架对于有大量训练数据的语言对的任务或场景中有很好的性能表现，但是对于一些只要少量语言对的任务中却并不能得到很好的效果，甚至可能比传统的统计机器翻译模型性能还要差。故，使用迁移学习的方法，将在大量平行双语语料上训练的模型参数迁移至只有少量双语平行训练任务的模型上。

迁移学习有很多种方法（如归纳迁移学习、直推式迁移学习、无监督迁移学习等），而 Zoph 等人使用的迁移学习为参数知识迁移学习，是以不同域的模型共享了一些参数或者是贡献了一些先验分布为前提的。比如，论文中提出，使用德语-英语大量平行语料训练德语-英语机器翻译系统，而只有少量数据量的西班牙语-英语的语料则将一部分德语-英语翻译模型的参数读取，并作为自身网络权重训练的起点，即不适用随机初始化的权重矩阵。这样的方法可以使得模型训练之前就有一定的先验分布，是有一定的翻译能力。在这基础上继续使用少量平行语料进行训练则比单纯只使用少量平行语料训练的模型性能要高很多。论文中还提出，如果预训练时所使用的语言对与微调时所使用的语言对在使用上或者语义、语法规则上相似，则会进一步提高迁移的效果，有助于模型性能的提升。

2.3.3 多任务训练方法

所谓多任务学习是指，某些网络的某些层或者某些权重参数共享，并将这个网络应用于多个相关的任务中的一种学习方法。多任务学习中，通常有多个目标针对不同的任务。例如 Rei 等人在 2017 年提出将语言模型和词性标注任务结合在一起[41]，使用同一个循环神经网络模型结构，在输出层有两种不同的任务——语言模型和词性标注。由于

两个任务目标相接近，故共享的模型参数可以很好的学习到相关的知识，并可应用于两个任务中的任意一个任务。

在机器翻译中的稀缺资源领域，也有这种多任务的方式进行学习。其主要方法是共享编码器或者共享解码器的方法。有多个编码器一个解码器的结构，我们可以称之为多对一的多任务结构，有一个编码器多个解码器的结构称之为一对多的多任务结构，以及多个编码器和多个解码器的结构称之为多对多的多任务结构。这些方法是希望其中一个任务能够有大量的数据，而另一个任务是稀缺资源的翻译任务。通过将两个任务的权重共享，可以达到有大量数据的任务帮助翻译任务学习。

2.4 本章小结

本章主要介绍了关于机器翻译相关的技术内容，分别从机器翻译的流程、语料的预处理、神经机器翻译的框架、不同神经网络的机器翻译模型以及一些训练方法的相关技术。

首先，介绍了机器翻译流程以及预处理方法。在机器翻译流程方面，通常我们首先进行语料的预处理，之后进行模型的训练，在解码的时候先将模型读进网络里，后开始解码。在语料预处理部分，介绍了最基本的语料预处理方式，即将句子分成一个个词，所谓词表示比短语更低颗粒度的单位。

第二节介绍了神经机器翻译模型的框架——“编码器-解码器”框架，该框架是目前神经机器翻译模型普遍使用的框架。编码器负责处理源语言端的序列，并将其抽象为语义层面的向量，而解码器则负责根据编码器输出的上下文向量以及目标端序列负责预测目标端词语。在该框架中通过加入注意力机制在很大程度上提升了神经机器翻译的翻译性能，使其超过传统的机器翻译模型。同时简单介绍两种常用的神经机器翻译网络模型以及详细介绍基于自注意力的网络模型。

第三节介绍了三种模型训练方法，这三种方法都可以运用于训练稀缺资源的神经机器翻译模型。其中反向翻译生成伪数据的方法实际用途比较广泛，常用于机器翻译评测中。而迁移学习的方法是比较高效且相对简单，低成本的稀缺资源的训练方式，同样对模型性能的提升有很大的帮助。第三种多任务学习的方法通常偏向理论研究层面，因为多任务学习的过程以及调试等还需要其他大量的经验性的知识。

第 3 章 蒙语语料预处理方法

3.1 研究动机

基于神经网络的机器翻译系统在性能上相较于之前的统计机器翻译模型有了很大的提升，尤其近几年随着深度学习越来越受欢迎，有更多的科研人员和学者开始研究神经机器翻译，提出很多有意义的创新，同时神经机器翻译模型的性能也有着更进一步的提升。虽然在性能上有了很大的提升，但是一般情况下需要大量的、有标注的数据进行模型的训练才能使其达到线上可用的效果。这就对于一些并不常见的语言对或者一些垂直领域的机器翻译系统性能并不能达到很好的效果。训练机器翻译系统需要大量的平行双语语料，而双语平行语料是属于带有标注的数据。通常情况下带有标注的数据是少量的，且制造和标注的成本往往很大。目前在市场上也有专门做数据的公司，为其他一些有数据需求的公司提供数据。这些数据有可能是数据公司本身多年的积累、也有可能短时间内能为客户的需求标注大量的数据。通常情况下购买数据的成本是非常大的，这就导致了很多平行数据数量很少，但是却有较高的需求。

我们搭建的基于神经机器翻译的蒙-汉机器翻译系统所使用的数据只有 19 万平行句对。而想要达到一个较好的、可用的机器翻译系统这些数据是远远不够的。为了在有限的平行语料的基础上减少数据稀缺问题，我们通过对比不同颗粒度的分词方法可以知道，颗粒度越细会减少一些词稀缺的问题，但随着数据颗粒度的细分化，导致数据句子本身的语义越加不明显，从而导致模型训练的困难以及翻译性能的下降。为了既能保留很大程度上的语言句子中语义信息，同时减少词稀缺的问题，结合蒙语文词的词根+词缀的形式知识，本课题中提出了一种高频率词+词缀 BPE 两种分词形式结合的数据预处理方法。

3.2 方法描述

3.2.1 蒙语单语语料数据分析

蒙语文属于一种黏着性语言，通常是由词根后面加若干个词缀组成。根据词缀的不同以及添加的个数不同，词本身的含义或者时态、词性等会发生变化，即每个词的构成和其语法意义的表示都依赖于不同词缀的连接，属于一种词形较丰富的语种[46-47]。通常，只有正确的切分词根、词缀等才能揭示其词类属性和语法关系。如图 3.1 所示为蒙语文

中通过连接不同的词缀来表达不同的词性的例子。

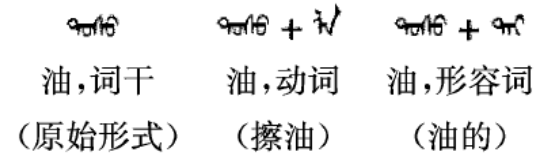

图 3.1 蒙语文词根和构词的附加成分
Fig. 3.1 The roots and affixes in Mongolian

通过上述例子我们可以看到词根“TVsO”是油的意思，是名词词性，在词根后缀接后缀“lA”变成“TVsOlA”就成为动词词性，有擦油的意思。而缀接不同的后缀“TAI”变成“TVsOTAI”，则意思变为有油的，词性变为形容词。其中，动词“TVsOlA”如果再追加“bA”变成“TVsOlAbA”或追加“nA”变成“TVsOlAnA”，则该词就变成擦油这个动词的过去形式或者将来时。从例子中我们可以知道，词根后追加不同的词缀以及添加多个词缀会改变词的词性、时态等信息。这种特性容易使数据中的一些词变成稀疏词，使数据的词有多样性，从而使数据进一步稀缺。

鉴于此，有科研工作者提出通过提取句子中的词根作为蒙语文预处理的结果。图 3.2 表示将蒙语文句子经过提取词根之后分词的结果。

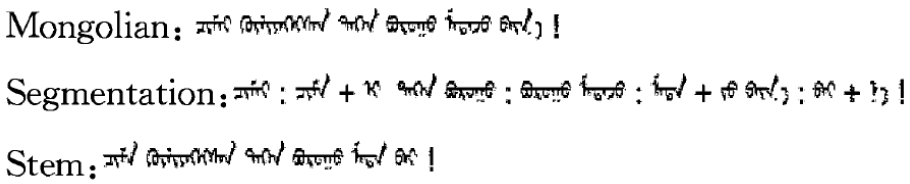

图 3.2 对蒙语文句子提取词根
Fig. 3.2 Extract the root of Mongolian words

第一行表示原始蒙语文句子，第二行则是将原始句子中某些词进行词干和词缀分离，第三行表示只留下词根部分作为最后的蒙语文预处理结果。但是简单通过词根提取的方式对蒙语文进行分词处理容易导致信息的丢失。因为虽然只留下词根会一定程度上缓解数据稀缺的问题，但是丢失词缀会使原始词的意义、时态、词性等信息丢掉，从而导致模型训练时无法捕捉这些信息，这样势必会影响模型的翻译质量。

表 3.1 高频词占比信息
Table 3.1 Ratio information of high-frequency words

阈值	50	150	200	250	300
总词表大小	19458	18700	18215	17900	17689
高频词大小	3778	2908	2375	2015	1765
高频词占比	19.42	15.55	13.04	11.26	9.98

在数据缺少以及词根资源缺少的情况下，本课题使用高频词与词缀 BPE 混合的分词方法对蒙语文进行预处理。长度为3~16的高频词在整个词表中的占比信息如表3.1所示。

3.2.2 高频词与词缀 BPE 混合分词方法

蒙语文不像英语一样，在词根后面加后缀有可能会改变词根字母（例如，-self 后加 s 时将 f 变为 v 再加 es 最后变成-selves 等），而是后加词缀时是直接拼接的（例如上例中“TVsO”后面加词缀“lA”时是直接拼写成“TVsO lA”），所以我们不需要考虑词缀拼接的规则。而由于缺少蒙语文词根资源，我们使用高频词代替词根的方法。我们首先将高频词特定长度（长度主要是 3~16 之间的高频词）之内的词抽取出来，通过上一节表格可知，约有 20%左右的其他词可以被这些高频词分割。在分词时，我们首先将这些高频词保护起来不让其被 BPE 方式分词，后再将剩下的词缀以及未能被高频词分割的低频词进行 BPE 分子词。这么做主要是因为词缀往往有不同种，一样会导致数据稀缺的问题，而通过 BPE 分子词，可以对非高频词以及词缀部分进行分子词。

总结来说，为了使蒙语句子不丢失语义的情况下，同时为了兼顾数据稀缺的问题，我们采用高频词加上词缀、低频词 BPE 的方式对蒙语文进行预处理。对于预料中汉语部分，我们简单使用现有的汉语分词工具加上 BPE 分子词，这样做同样是为了缓解数据稀缺问题。

3.3 实验

3.3.1 实验设置

我们首先使用不同的分词方法对数据进行预处理，并通过不同实验进行对比。之后我们再使用高频词+词缀 BPE 混合预处理的方法对数据进行预处理。使用的评价方法是 BLEU 自动评价方法，BLEU 值越高表示性能越好（是一种将输出译文与标准译文进行对比的方法，两者越相似 BLEU 值越高），测试 BLEU 值使用的脚本是 mteval-13.perl 脚本，该脚本是一种开源的测 BLEU 值的脚本之一。

由于蒙语-汉语平行双语语料缺少，我们仅使用 CWMT 机器翻译评测官方提供的数据（其中包括的数据集有 ICT-MC-corpus-CWMT2017，IMU-corpus-CWMT2017，IIM-CWMT2015，IMU-CWMT2013，IMU-CWMT2015，IMU-dev-mnzh-CWMT2017，IMU-dev-mnzh-CWMT2018），该数据集大约有 26 万句对，其中中文语料约有 323 万中文词，最长句长为 132 个词，蒙语文语料约有 494 万字，最长句长为 264 个字。采用的

语料主要提供方有中国科学院计算机技术研究所、内蒙古大学、中国科学院合肥智能机械研究所，其详细信息如下表 3.2 所示。

表 3.2 蒙汉平行语料信息

Table 3.2 Information of Mongolian-Chinese corpus

数据集	领域	数据量	提供单位	备注
ICT-MC-corpus-CWMT2017	新闻	30007	中科院计算所	
IIM-CWMT2015	新闻	1682	中科院合肥机械所	
IMU-corpus-CWMT2017	政府文件	100001	内蒙古大学	
IMU-CWMT2013	口语、新闻	104790	内蒙古大学	
IMU-CWMT2015	口语、新闻	24978	内蒙古大学	
IMU-dev-mnzh-CWMT2017	口语、新闻	1000	内蒙古大学	4 个参考译文
IMU-dev-mnzh-CWMT2018	口语	1001	内蒙古大学	测试集

使用的测试集为 IMU-dev-mnzh-CWMT2018 数据集，其中包含有 1001 句源语言，有 4 个参考译文，详细信息参考表 3.2。在预处理数据时，使用的 BPE 合并次数统一为 16000 次，而使用不同的预处理方法时所生成的词表大小均不一样，具体详细信息可参考表 3.3。

表 3.3 不同预处理方法所产生的词表大小

Table 3.3 The size of vocabulary produced by different preprocessing methods

预处理	Word	BPE	WdPiece	Wd1+BPE	Wd2+BPE	Wd3+BPE	Wd4+BPE	Wd5+BPE
源词表	30000	16308	16518	19458	18700	18215	17900	17689
目标词表	25000	23654	23654	23654	23654	23654	23654	23654

表格中 WdPiece 表示 wordpiece 分词方法，而 Wd1~5+BPE 是高频词加词缀预处理方法。其中 1~5 分别表示取高频词时使用出现频率分别高于 100 次、150 次、200 次、250 次和 300 次。

在系统实现方面，模型训练时我们使用在谷歌公司开源的 Tensor2Tensor 开源工具包的基础上进行修改之后的系统。Tensor2Tensor 是 2017 年谷歌公司开源的一个深度学习模型的开源库，其中不仅仅包含着机器翻译，而且还包含着图像识别、语音识别、语言模型、文本摘要等多种任务的同时还包括了循环神经网络结构、卷积神经网络结构以及自注意力网络结构。而我们使用的基于自注意力的神经机器翻译是其中一个重要的组成部分。Tensor2Tensor 开源库是同样由谷歌公司开发的深度学习框架 Tensorflow 进行搭建的。目前该开源库一直处于更新状态，我们在做实验时使用的版本是 Tensor2Tensor-1.6.3，故本文中使用的结构在此版本基础上进行改进。

开源的基于自注意力机制的神经机器翻译系统 Tensor2Tensor 针对不同的实验需求和平台情况，预先封装好若干组不同的参数集合。在本实验中我们使用的模型参数集固

定为 transformer_base_single_gpu 集，参数集中具体的参数值如下表 3.4 所示。

表 3.4 transformer_base_single_gpu 参数集参数值信息
Table 3.4 Information of transformer_base_single_gpu param-set

参数	值
隐层大小	512
batch 大小	8192
Initializer	uniform_unit_scaling
学习率	0.1
warmup 步数	16000
层数	6
标签平滑	0.1
过滤器大小	2048
头数	8
前馈层激活函数	ReLU
残差网络[42]dropout[43]	0.1
Attention dropout	0.0
ReLU dropout	0.0
SGD 算法	Adam
Adam_epsilon	1e-9
Adam_beta1	0.9
Adam_beta2	0.98
最大句长	256
更新次数	80000

该参数集主要是在搭建基准系统且使用单个显卡的时候使用的预先定义好的参数集，其中 batch 大小设置为 8192，即每一次更新时所读取的源语言端或目标语言端最大的词的个数。Warmup 为 16000 表示在更新次数达 16000 次以前学习率线性增长，超过 16000 次更新之后学习率指数下降，以确保模型能在保证收敛速度的情况下能够很好的收敛。更新次数设置为 80000 次，在 batch 为 8192 的情况下能保证模型训练 40 轮以上。

表 3.5 服务器配置
Table 3.5 Server configuration

	组件	配置
服务器	中央处理器	Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz
	内存	64GB
	图形处理单元	NVIDIA TITANX
	显存	12GB
	硬盘	3.5TB

我们所使用的实验设备是使用带有图形处理器的服务器（GPU 服务器，适用于计算深度学习）对神经机器翻译系统进行训练和解码以及调试。图形处理器通常比普通的

CPU 在矩阵运算方面有很大的优势，这样我们训练速度和解码速度都有速度上的保障。所使用的服务器主要是使用实验室提供的显卡服务器，所使用的具体设备信息如表 3.5 所示。

另外我们在实际中使用的模型采用了 checkpoint ensemble 的方法，集成了 10 个最近保存的模型参数，对这些模型参数进行平均，这样做的目的是提高实验的准确性，减少单一模型性能上的偶然因素。这样做很大程度上可以使得实验结果更稳定，得到的曲线相对来说更平滑，在后续的实验过程中我们围绕模型集成的结果进行分析。

在解码时，我们使用的解码参数如表 3.6 所示。

表 3.6 解码参数配置

Table 3.6 Decoding parameter configuration

参数	值
batch 大小	32
长度惩罚	1.2
beam 大小	4

3.3.2 实验结果

3.3.2.1 蒙语文预处理对比实验

本小结主要对比不同预处理方法对于基于自注意力的神经机器翻译系统的性能影响。主要的预处理方法有基于词的预处理、基于子词 BPE 预处理、基于 Wordpiece 的预处理方式，其中基于词分词工具使用自己开发的分词工具，BPE 分子词工具使用开源的 Sennrich 的版本，Wordpiece 工具使用 Tensor2Tensor[44]里分词片的工具进行二次开发后的工具。首先，我们对测试集进行了去重处理，即将 1001 句测试集中源语言句子与训练集中源语言句子有重复的进行去除，保证测试集中的所有句子在训练时都没有见到过，提高测试的可信度，经过去重之后剩下 658 句。

表 3.7 不同蒙语文预处理时系统 BLEU 值得分

Table 3.7 BLEU score when applied different preprocessing

预处理方法	测试集（BLEU-5）
Word	28.09
BPE	32.18
Wordpiece	31.62

其次，将 658 句带有四个参考译文改变为 658*4 句单参考译文形式，即将 658 句源语言复制四份，每一份对应于不同的参考译文。故最后总共有 2632 句源语言句子并带有一个参考译文。在测试 BLEU[45]值时由于使用基于字的计算方式，故使用 BLEU-5（基

于词的计算 BLEU 方式通常都使用 BLEU-4 作为最后得分，通常情况下 BLEU-4 会得到比 BLEU-5 更高的得分）。不同蒙语文预处理方法对翻译系统的影响结果如表格 3.7 所示。

3.3.2.2 高频词+词缀 BPE 预处理实验

使用高频词+BPE 分词的方式进行预处理时的结果如表格 3.8 所示，其中高频词的选取方式为根据词频统计，词频分别在 100 以上、150 以上、200 以上、250 以上以及 300 以上时得到不同的预处理结果，也得到不同的系统性能。

表 3.8 高频词+BPE 的方式进行预处理时 BLEU 值得分

Table 3.8 BLEU score of when mixing high frequency words and BPE

系统	词频阈值	测试集（BLEU-5）
Baseline	-	32.18
Word1_BPE	100	31.10
Word2_BPE	150	31.54
Word3_BPE	200	32.11
Word4_BPE	250	32.61
Word5_BPE	300	32.45

3.3.3 结果分析

3.3.3.2 蒙语文预处理对比实验

蒙语文预处理实验主要是想验证基于子词的预处理方法能够很好的缓解数据稀缺的问题。通过对比我们可以看出基于词的预处理方法要比字节对编码的预处理方法低 4.09 个 BLEU-5 分，比基于词片的预处理方法低 3.53 个 BLEU-5 分。基于词的分词方法在源语言端的原始词表约有 9 万 3 千多个词。考虑到模型大小等问题，我们取频率最高的 30000 个词作为源语言词典。所以约有 6 万多个词被认为是稀缺词，并使用 UNK 标识符标识。这就带来了 OOV 等问题，同时数据稀缺性得不到较好的解决。而基于子词的方法处理源语言端则词表约有 1 万 6 千个词左右，这表明很多词都是高频词。或者理解为源语言端的 494 万个字由这 1 万 6 千个词所组成的不同句子。这就在很大程度上缓解了数据稀缺、数据多样性的问题。

基于子词的预处理方法我们使用了两种：字节对编码分词方法和词片分词方法。这两种方法原理和动机上是类似的，但由于不同的实现方式有着不同的效果。在蒙语-汉语翻译系统上字节对编码的方式比词片的方式要高 0.56 个 BLEU-5 值。但是，这与语言对有比较大的关系。在 Vaswani 等人提出的基于自注意力的神经机器翻译模型的论文中在英语-德语的任务上使用词片的方式得到了最好的效果。

3.3.3.2 高频词+BPE 预处理实验

从表格中我们可以看出，抽取不同频率的高频词有不一样的预处理效果，进而导致模型在性能上的差距。如图 3.3 所示，是词频阈值分别为 100、150、200、250 和 300 时不同预处理方法对模型性能的影响，从图中我们可以看出，在词频阈值 100 和 250 时模型性能显著的上升，但词频阈值再增大时模型性能略有下降。

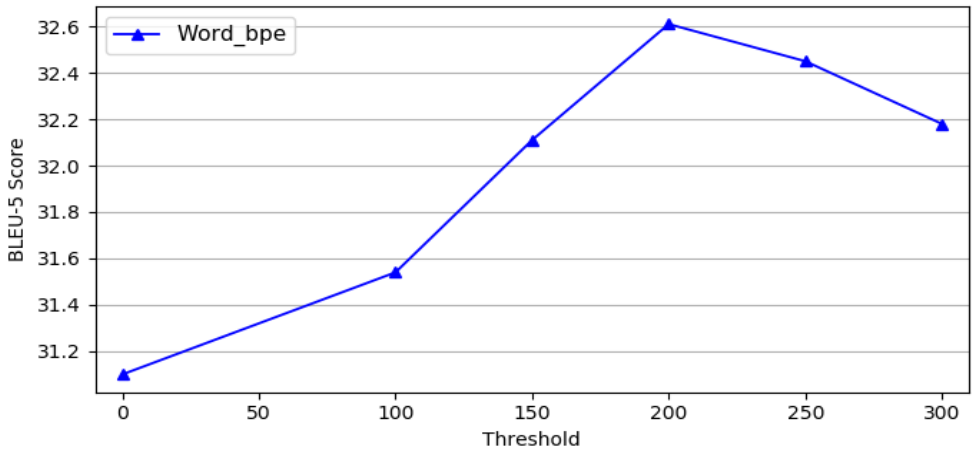

图 3.3 模型性能与词频阈值的关系

Fig. 3.3 The relation between threshold and performance of system

出现这种情况我们认为是较高的词频阈值会使得更多的高频词无法被分子词，数据多样性增加，数据稀缺程度增加，进而影响模型性能。

3.4 本章小结

本章节主要包括了三部分内容，分别是研究蒙语预处理方法的动机、预处理方法的具体描述以及实验相关设置、结果和结果分析等。

首先，我们通过描述可知，语料的预处理对模型的性能有着重要的影响。不仅仅是自然语言处理方面，包括图像领域等其他领域都需要对原始数据进行预处理。在自然语言处理领域中，分词部分是重要的预处理步骤之一，好的分词方法可以减少数据稀缺的问题以及多样性，使模型更容易训练并得到更好的性能。为了兼顾数据的语义能够保留的同时能缓解多样性问题，我们使用高频词和 BPE 混合的方式对数据进行预处理。

其次，为了解决上述的问题，我们使用了简单高效的方法。首先我们抽取一部分高频词，将其当作词典中的词，在分子词的过程中不将其分词。后对低频词以及将高频词与词缀分词之后对后缀进行字节对编码的方式分子词。这样对于常见的词我们保留整个词，使得句子中的高频词得以保留其语义信息，而对于低频词或词缀为了防止出现 UNK 使用分子词的方法对其进行分词。

最后，我们将我们的想法进行验证。首先我们做了一组对比实验，以确保分子词的

方式能更好的解决数据稀缺和数据多样性的问题。其次，我们通过改变不同的词频阈值得到不同的数据预处理结果。通过对比我们发现，在阈值较大的情况下模型性能有略微提升，表明如果高频词太多，反而容易出现数据多样性的问题，导致性能下降。故在合理的阈值时，模型能得到更好的效果。

本章主要是研究在数据量一定且数据较少的情况下数据预处理能为模型性能能够带来的影响，下一章则在模型训练层面对系统性能影响的研究。

第 4 章 基于迭代生成伪数据的模型训练方法

4.1 研究动机

神经机器翻译在性能上相比传统统计机器翻译提高很多。但有一个很严重的问题就是，当数据较稀缺的时候，神经机器翻译一般情况下却比统计机器翻译性能要差。而且，当模型很大，而训练数据较少的时候会很容易产生过拟合，影响模型性能。我们本课题中使用的神经机器翻译是基于自注意力的神经机器翻译，在能得到很好的性能的情况下的模型参数使得模型比较大。比如，基于循环神经网络的神经机器翻译模型系统在一层网络层且隐藏层大小为 128 的情况下也能得到不错的一个性能表现，而基于自注意力的神经机器翻译模型通常情况下需要更多层数且隐藏层大小往往设置的比较大，才能有很好的效果。在我们的实验中使用的参数集为 transformer_base_single_gpu 参数集，当原语词表和目标语言词表大小分别为 16308 和 23654 时，在这个参数集下模型大小如表 4.1 所示。

通过下述表格我们可以知道，模型总共有 76675072 个浮点数，模型大小约为 292.49MiB，所以该模型比较大、比较重，而我们所使用的训练数据只有 26 万句对，很容易出现过拟合的现象。通常我们可以使用 dropout，正则化等方法降低模型对训练数据的过拟合情况。但是，当数据量很少时这种方法往往达不到预期的效果，数据量过小时是没有办法通过改变超参的方法去提升性能。

再有一个方面就是虽然标注的平行双语语料通常很难获取，且获取成本较大，但是未标注的单语语料却可以低成本、大数据量的获取。而使用大量单语数据来提高数据稀缺情况下模型性能也是研究的一个热点之一。在考虑到大模型训练数据少且有大量单语语料的情况下，我们这章节使用单语语料来生成伪数据的方法来提高模型性能。由于模型一旦训练好之后，模型所能表达的能力就已固定，故我们首先考虑在模型训练层面使用单语语料来训练模型。我们使用单语语料来生成伪数据，并使用伪数据筛选的方式提高伪数据的质量，使其尽可能接近高质量数据的表达方式，通过增加数据量来增加平行数据的量。为此，我们提出一种筛选伪数据的方法来筛选伪数据，同时使用微调训练的方法提高模型的性能。通过实验发现只靠增大数据量不能进一步提高模型性能时，我们通过提高伪数据质量的方法来提高最终模型性能，因此在微调训练的实验基础上进一步通过迭代生成伪数据的方法生成质量更高的伪数据进行模型的训练。

表 4.1 默认参数集下模型大小

Table 4.1 Model size of default param-set

参数名	参数形状	浮点数个数
Encoder_self_attention_q × 6	(512, 512)	1572864
Encoder_self_attention_k × 6	(512, 512)	1572864
Encoder_self_attention_v × 6	(512, 512)	1572864
Encoder_self_attention_output × 6	(512, 512)	1572864
Encoder_self_attention_layernorm_scale × 6	(512,)	3072
Encoder_self_attention_layernorm_bias × 6	(512,)	3072
Encoder_ffn_conv1_kernel × 6	(512, 2048)	6291456
Encoder_ffn_conv1_bias × 6	(2048,)	12288
Encoder_ffn_conv2_kernel × 6	(2048, 512)	6291456
Encoder_ffn_conv2_bias × 6	(512,)	3072
Encoder_ffn_layernorm_scale × 6	(512,)	3072
Encoder_ffn_layernorm_bias × 6	(512,)	3072
Encoder_layernorm_scale	(512,)	512
Encoder_layernorm_bias	(512,)	512
Decoder_self_attention_q × 6	(512, 512)	1572864
Decoder_self_attention_k × 6	(512, 512)	1572864
Decoder_self_attention_v × 6	(512, 512)	1572864
Decoder_self_attention_output × 6	(512, 512)	1572864
Decoder_self_attention_layernorm_scale × 6	(512,)	3072
Decoder_self_attention_layernorm_bias × 6	(512,)	3072
Decoder_encdec_attention_q × 6	(512, 512)	1572864
Decoder_encdec_attention_k × 6	(512, 512)	1572864
Decoder_encdec_attention_v × 6	(512, 512)	1572864
Decoder_encdec_attention_output × 6	(512, 512)	1572864
Decoder_encdec_attention_layernorm_scale × 6	(512,)	3072
Decoder_encdec_attention_layernorm_bias × 6	(512,)	3072
Decoder_ffn_conv1_kernel × 6	(512, 2048)	6291456
Decoder_ffn_conv1_bias × 6	(2048,)	12288
Decoder_ffn_conv2_kernel × 6	(2048, 512)	6291456
Decoder_ffn_conv2_bias × 6	(512,)	3072
Decoder_ffn_layernorm_scale × 6	(512,)	3072
Decoder_ffn_layernorm_bias × 6	(512,)	3072
Decoder_layernorm_scale	(512,)	512
Decoder_layernorm_bias	(512,)	512
Input_emb	(16308, 512)	8349696
Target_emb	(23654, 512)	12110848
Softmax	(23654, 512)	12110848
Total	-	76675072

4.2 方法描述

4.2.1 伪数据筛选方法

虽然我们可以低成本且很方便的生成大量的伪数据，但是伪数据质量往往不是很高，甚至偶尔会出现有些句子未能翻译而出现空行、或者源语言句子太长导致翻译的译文只有一部分等。通常情况下，这种类型的伪数据不但不会有助于模型的训练，反而会降低模型的性能。故我们提出一种筛选伪数据的方法，将伪数据中质量比较好的伪数据选取出来。筛选方法主要包括对单语语料的筛选，对生成的伪平行语料筛选。

第一步，我们将待翻译的单语语料进行过滤。由于数字、日期、时间、人名、组织机构名等实体通常需要使用泛化的方法来解决其多样性的问题，故单语语料中，我们首先使用规则将这些实体进行过滤，即我们生成的伪数据中不包含这些实体。上述中泛化通常表示将这些实体识别之后使用统一的符号进行替换，将其归为一类。例如，我们将所有的时间都泛化成标识符“$TIME”、而将数字泛化成标识符“$NUMBER”，这样我们就可以将实体的多样性去除，便于模型的训练。第二步，我们使用 n 元语言模型进行过滤。通常情况下我们有使用大量高质量数据训练好的、传统的 n 元语言模型，使用该训练好的语言模型我们可以对单语语料进行打分，判断每一个句子的顺畅性，将不顺畅的句子丢弃。经过两轮筛选后我们得到质量比较高的单语语料约有 400 万，第三步就是将这 400 万单语语料进行翻译。

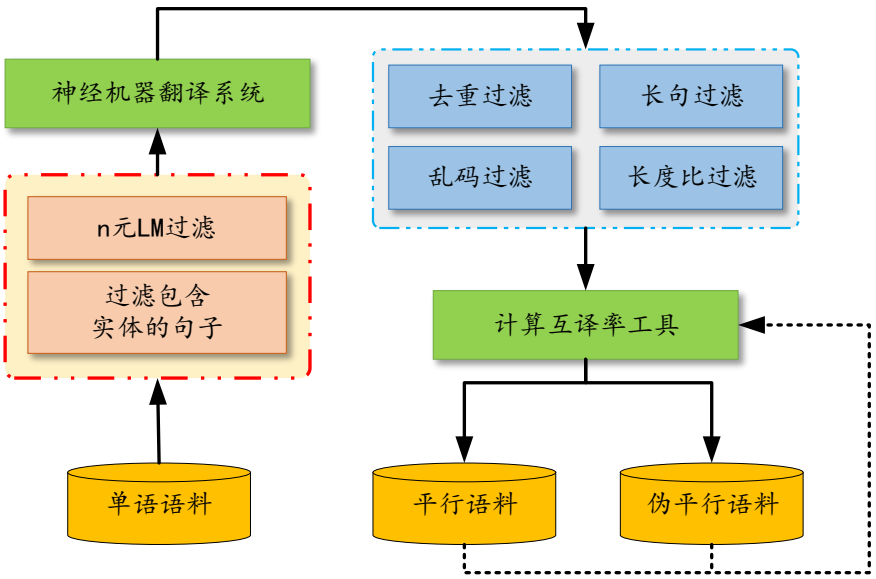

图 4.1 伪平行语料筛选流程

Fig. 4.1 Filtering process of pseudo data

最后一步的筛选过程则是对伪平行语料进行筛选。通常筛选双语数据时，我们都会有长度比过滤、乱码过滤、长度过滤以及重复句对过滤等过程。这些方法同样被运用于过滤伪数据的过程中。除此之外，我们还使用互译率的方法进行双语平行数据的过滤。在了解互译率之前，我们需要了解词对齐，所谓词对齐就是在双语文本中的互译关系的词之间建立对应关系，对应关系有可能有一对多、多对一、多对多或者不被对齐等情况。通常我们可以根据双语平行语料通过无监督的方式进行学习，并对于双语语料生成对应的对齐文件。通过现有的工具我们可以得到双语语料正向对齐文件以及反向对齐文件，进而使用正反向对齐文件可以获取此双语语料的互译率——即根据词对齐的信息得到源语言句子被翻译成目标语言句子的概率（反之亦然，可以得到目标语言句子被翻译成源语言句子的概率）。最后根据互译率值筛选分数相对较高的句子对。但是由于训练词对齐的过程是无监督过程，是根据语料的规律学到的，所以直接将大量伪数据与高质量数据混合在一起训练，则结果会偏向伪数据的对齐关系，使伪数据的对齐关系作为正确的对齐关系。为了解决这个问题，我们使用迭代的方法，每次混合伪数据的量不超过已经融合的数据的百分之三十。例如，第一步我们将 20 万高质量数据与 6 万伪数据进行融合，并使用互译率的方法进行筛选，生成 24 万平行双语数据；我们再将上一步所获得的 24 万平行双语数据与 7 万伪数据融合，使用互译率方法进行筛选生成 30 万平行数据，同样的道理再融合 9 万伪数据，生成 37 万平行双语数据。以此类推，我们可以通过多次迭代生成越来越多的伪双语数据的同时尽量保持伪数据的质量相对高。具体流程如图 4.1 所示。

4.2.2 基于微调的训练方法

通过对第三章的分析可知，蒙语文-中文平行数据相对缺少。但是我们可以很容易且低成本的得到大量的中文单语语料，通过大量单语数据提高资源稀缺时的模型性能有很多种方法。其中，Sennrich 等人提出的反向翻译方法不仅方法简单而且对模型性能的提高有很大的帮助。目前国内国外很多评测中大部分团队都会使用反向翻译的方法提高模型性能。比如，在英语到中文的翻译任务中，虽然有大量的双语语料（有千万级的数据量）但是通常一样会使用大量的单语语料进行微调训练。即首先使用大量的双语平行语料进行模型的训练，后使用反向翻译方法生成百万甚至千万级的伪双语语料，并用这个语料进行进一步训练已训练好的模型。微调训练时为了防止过度学习伪数据的内容，通常把学习率降低。通过这样的方法一般能进一步提高 1 个 BLEU-4 分左右。

这样的方法虽然可以用于英语到中文或中文到英语的、有大量双语平行训练语料的

翻译任务中，但是对于只有少量训练数据的任务中却并不太适用。因为少量的训练数据并不能让模型有很强的表示能力，在这种情况下再使用大量的伪数据去训练模型，势必将模型的表示能力影响，并降低其在测试集上的性能表现。为了利用大量单语语料制造伪数据，同时又希望模型的性能有所提高，我们提出一种使用伪数据训练初始模型，并只使用少量的高质量数据进行模型训练的方法，方法的整个流程如图 4.2 所示。

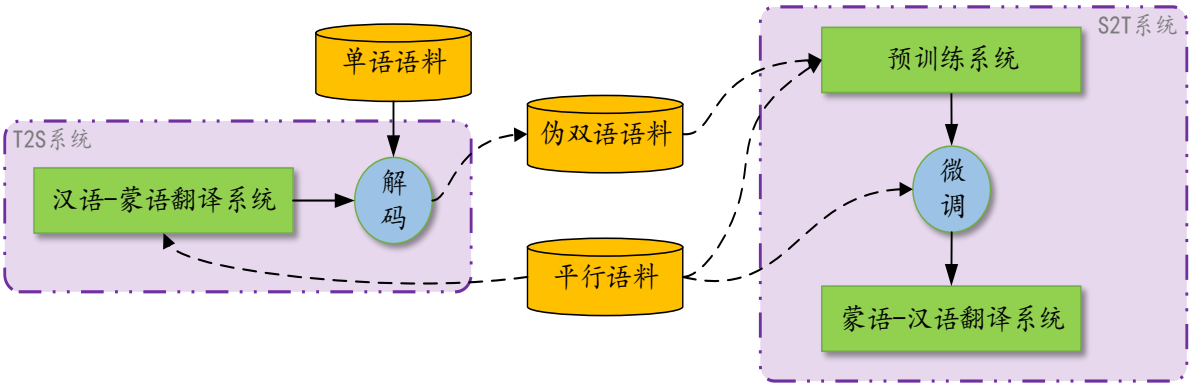

图 4.2 基于微调训练的方法流程

Fig. 4.2 Process of fine-tune training method

通过上图，我们可以看到，首先利用少量的单语语料训练反向翻译的系统——汉语-蒙语翻译系统。这个系统主要用来生成伪双语语料，即系统训练好之后我们将使用大量单语中文语料进行反向翻译。其次，利用反向翻译生成的大量伪双语数据加上少量的高质量平行语料训练我们初始的蒙语-汉语系统，此系统主要是预训练的系统，通常情况下直接使用大量的伪双语语料训练系统时在测试集上不会得到太好的性能。这个我们可以看图 4.3 所示的理解，即有大量的伪数据集 A 集以及少量的数据集 B 集，而我们的校验集或者测试集是与 B 集相似的 B2 集。则使用 B 集训练系统，在 B2 集上测试其性能一般能得到较好的效果，而使用 A 集训练，虽然有大量数据，但由于与 B2 集不相似，故不一定能得到较好的性能。为了使用 A 集训练系统同时希望在 B2 集上有好的性能表现，所以最后一步我们只使用高质量的平行双语语料进行模型的微调。

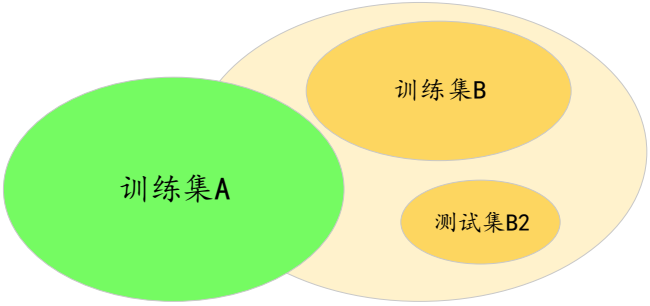

图 4.3 训练集示意图

Fig. 4.3 Diagram of training set

4.2.3 基于迭代生成伪数据的训练方法

上一节所述方法经过验证在实际任务中是非常有效的。因此，我们通过继续增加伪数据的量，希望模型能够在更大的训练数据集上能够有更好的训练，再使用高质量数据进行微调训练。因为虽然是伪数据但是数据量大时能学到的能力更多、表示能力也应该更强大。但是，实际的情况是当我们将伪数据的量从一百万增加，一直增加到两百万时实际的性能并没有明显的增加，反而甚至略有下降。从上述情况可推断，当伪数据量大的时候，并不一定能对最终模型的性能有正向的影响。这有可能是因为伪数据的量大，导致过滤筛选时质量分布不均匀，从而导致虽然数据量大，但是数据集本身的规律越来越模糊，使模型的训练和学习越来越困难。

也就是说，如果想继续使用单语语料生成伪数据的方法进一步提高模型，那么不应该从量入手，而是将伪数据的质提高。为了提高伪数据的质量的提高，我们提出使用一种迭代生成伪数据的方法来提高伪数据的质，具体流程如图 4.4 所示。

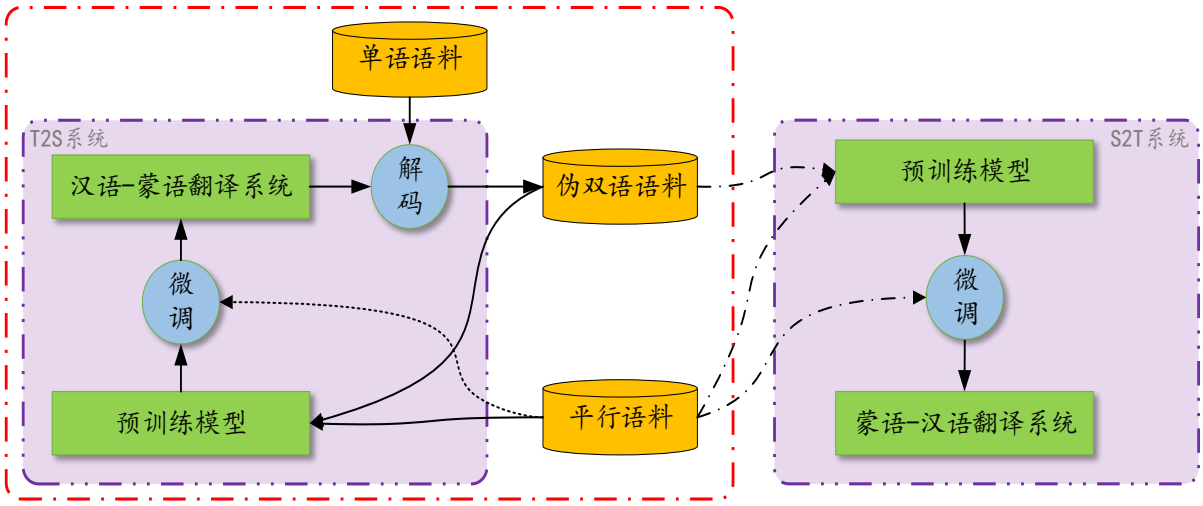

图 4.4 迭代生成伪数据的训练方法流程图

Fig. 4.4 Process of training method for iteratively generating pseudo data

从图中流程我们可以看出，我们将上一节使用的方法应用于汉语-蒙语的系统中，这样使得汉语-蒙语系统比只用高质量数据训练得到的系统有更好的效果。将得到更好效果的汉语-蒙语系统又应用于生成伪数据，利用新生成的伪数据再将汉语-蒙语系统训练、更新，使其有更强的表示能力，直到汉语-蒙语系统性能不能再上升为止，上图中红色虚线框内的流程可以重复多次。这样迭代的方法生成的汉语-蒙语系统有比单纯使用一次迭代的系统有着更高的性能，反向翻译的结果应该更好。而使用该系统生成的伪数据则达到了我们所讨论的提高伪数据质的目的。

4.3 实验

4.3.1 实验设置

我们在源语言端和目标语言端都是用 BPE 作为预处理方式，使用的评价方法是 BLEU 自动评价方法，测试 BLEU 值使用的脚本是 mteval-13.perl 脚本，该脚本是一种开源的测 BLEU 值的脚本之一。

使用的蒙汉、汉蒙平行语料是 CWMT 机器翻译评测官方提供的数据，该数据大约有 26 万句对，其中中文语料约有 323 万中文词，最长句长为 132 个词，蒙语文语料约有 494 万字，最长句长为 264 个字。采用的语料主要提供方有中国科学院计算机技术研究所、内蒙古大学、中国科学院合肥智能机械研究所，其详细信息如下表 4.2 所示。

表 4.2 蒙汉平行语料及中文单语语料信息

Table 4.2 Information of Mongolian-Chinese corpus and Chinese corpus

数据集	领域	数据量	提供单位	备注
ICT-MC-corpus-CWMT2017	新闻	30007	中科院计算所	
IIM-CWMT2015	新闻	1682	中科院合肥机械所	
IMU-corpus-CWMT2017	政府文件	100001	内蒙古大学	
IMU-CWMT2013	口语、新闻	104790	内蒙古大学	
IMU-CWMT2015	口语、新闻	24978	内蒙古大学	
IMU-dev-mnzh-CWMT2017	口语、新闻	1000	内蒙古大学	4 个参考译文
IMU-dev-mnzh-CWMT2018	口语	1001	内蒙古大学	测试集
XMU	新闻	7423867	-	中文单语语料

使用的测试集为 IMU-dev-mnzh-CWMT2018 数据集，其中包含有 1001 句源语言，有 4 个参考译文。其中 XMU 语料是中文单语语料，是新华网关于新闻领域的单语数据，约有 700 多万句，在实验中，我们最多使用了 250 万单语语料进行反向翻译。在预处理数据时，在源语言端和目标语言端使用的 BPE 合并次数统一为 16000 次。

在系统实现方面，模型训练时我们使用在谷歌公司开源的 Tensor2Tensor 开源工具包的基础上进行修改之后的系统。Tensor2Tensor 开源库是由同样由谷歌公司开发的深度学习框架 Tensorflow 进行搭建的。我们在做实验室使用的版本是 Tensor2Tensor-1.6.3，故本文中使用的结构在此版本基础上进行改进。

开源的基于自注意力机制的神经机器翻译系统 Tensor2Tensor 针对不同的实验需求和平台情况，预先封装好若干组不同的参数集合。在本实验中反向翻译系统以及蒙汉翻译系统我们使用的模型参数集都固定为 transformer_base_single_gpu 集。参数集中具体的参数值如下表 4.3 所示。

表 4.3 transformer_base_single_gpu 参数集参数值信息
Table 4.3 Information of transformer_base_single_gpu param-set

参数	值
隐层大小	512
batch 大小	8192
Initializer	uniform_unit_scaling
学习率	0.1
warmup 步数	16000
层数	6
标签平滑	0.1
过滤器大小	2048
头数	8
前馈层激活函数	ReLU
残差网络 dropout	0.1
Attention dropout	0.0
ReLU dropout	0.0
SGD 算法	Adam
Adam_epsilon	1e-9
Adam_beta1	0.9
Adam_beta2	0.98
最大句长	256
更新次数	80000
微调次数	20000

该参数集主要是在搭建基准系统且使用单个显卡的时候使用的预先定义好的参数集，其中 batch 大小设置为 8 倍的 1024，表示每一次更新时所读取的源语言端或目标语言端最大的词的个数。也就是说，每一次更新时所读取的词数是固定的，而句子个数不固定。Warmup 为 16000 表示在更新次数达 16000 次以前学习率线性增长，超过 16000 次更新之后学习率指数下降，以保证模型能在保证收敛速度的情况下能够很好的收敛。更新次数设置为 80000 次，在 batch 为 8192 的情况下能保证模型训练 40 轮以上。

另外我们在实际中使用的模型采用了 checkpoint ensemble 的方法，集成了 10 个最近保存的模型参数，对这些模型参数进行平均，这样做的目的是提高实验的准确性，减少单一模型性能上的偶然因素。这样做可以很大程度上可以使得实验结果更稳定，得到的曲线相对来说更平滑，在后续的实验过程中我们围绕模型集成的结果进行分析。

反向翻译系统训练的参数与正向翻译系统的训练参数保持一致，因为通常情况下数据的翻译方向不与训练参数强相关。但是解码过程中长度惩罚因子 alpha 则与正向解码时不同，正向解码时我们将 alpha 设置为 1.2，在反向翻译过程中我们将这个值设置为 0.5，这个值是我们在校验集上通过调参调出来的。

4.3.2 实验结果

从上述章节中我们知道，我们使用 20 万的数据训练了目标语到原语的机器翻译系统，后使用大量单语语料 XMU 进行反向翻译生成伪数据。实验中我们反向翻译了大约两百五十万单语语料。这些单语语料是将七百多万的单语语料经过筛选之后剩下的四百六十多万的单语语料中随机筛选出来的，筛选方法有语言模型，规则等过程。后我们将两百五十万多万单语语料通过不同批次进行反向解码，生成对应数量的单语语料。由于数据质量或者某些不确定因素生成的伪数据中有些句子未能翻译出来，即翻译结果为空，我们首先将这部分数据进行清除。然后使用常规的数据过滤方法（长度比过滤、长句子过滤、乱码过滤）进行进一步的过滤。

表 4.4 过滤伪数据信息

Table 4.4 Information of filtering pseudo data

迭代序号	增加数据量	最后数据量
d_0（Original）	-	192141
d_1（+d_0*31%）	60000	241254
d_2（+d_1*29%）	70000	301243
d_3（+d_2*30%）	90000	377325
d_4（+d_3*32%）	120000	475042
d_5（+d_4*32%）	150000	605351
d_6（+d_5*30%）	180000	779156
d_7（+d_6*30%）	230000	984191
d_8（+d_7*25%）	250000	1206335
d_9（+d_8*30%）	360000	1521254
d_10（+d_9*30%）	460000	1943211
d_11（+d_10*30%）	580000	2416978

为了保证数据相对有较高的质量，我们在上述过滤的基础上又使用互译率进行迭代过滤。具体使用的方法可以参考本章节中数据过滤过程的解释，本节中将其结果展示如表 4.4 所示。表格中，我们将迭代序号标记为“d_n”，其中 n 表示第几次迭代，而括号中的“+d_n*m%”表示我们当前的数据增加的量，d_n-1 是上一个迭代剩下的数据量加上其百分之 m 的伪数据量；例如 d_2=d_1+d_1*29%，即在 24 万数据量的基础上再加 24 万乘以 0.29 的伪数据（即 7 万伪双语数据），经过互译率的过滤方式过滤最终生成 30 万的数据。经过如此反复迭代过滤伪数据，最终我们生成了大约 240 万句对的双语数据，并在后续的实验中使用了大约 220 万的数据进行实验。

首先，我们使用生成的伪数据进行预训练，然后用 20 万高质量数据微调。在使用伪数据预训练时，我们首先使用 100 万的伪数据与高质量数据融合进行模型的训练，其

结果是对模型的性能提升有很大的帮助，故我们又将伪数据的量慢慢增大至 210 万，共 230 万左右的数据进行模型的预训练。使用伪数据与高质量数据预训练后使用高质量数据微调的实验结果如表 4.5 所示。

表 4.5 使用伪数据和微调训练时模型的性能结果

Table 4.5 Results for using pseudo data and fine-tuning

系统	数据量	测试集	
		BLEU-4	BLEU-5
Baseline	19w	36.67	31.68
Baseline+100w	19w+101w	39.97	34.73
Baseline+150w	19w+143w	39.99	34.76
Baseline+200w	19w+210w	40.05	34.81

通过上述实验结果，我们发现增大伪数据的量所能带来的性能提升并不明显。比较合理的解释是，随着伪数据的增加，在使用互译率筛选的过程中筛选标准越来越差（因为伪数据的比重越来越多）则越来越多的质量差的伪数据可以被留下来，通过几轮迭代之后大量的质量较差的伪数据保留下来。

表 4.6 使用伪数据训练的汉-蒙翻译系统性能

Table 4.6 The performance of C-M system when using pseudo data

系统	数据	BLEU-4	
		8k-steps	100k-steps
CM_Baseline	19w(Original)	27.22	27.07
CM_Iter_1	120w(Decode by CM_Baseline)	21.18	29.09
CM_Iter_2	120w(Decode by CM_Iter_1)	22.90	29.21
CM_Iter_3	120w(Decode by CM_Iter_2)	23.39	29.16
CM_Iter_4	120w(Decode by CM_Iter_3)	23.87	28.16

鉴于我们无法通过增大伪数据的量进一步提升模型的性能，我们想到通过提高伪数据的质来使得模型的性能进一步提高，为此，我们提出通过迭代生成伪数据的方法生成伪数据的方法，此方法中有两个系统，分别是源语言到目标语言的翻译系统以及目标语言到源语言的翻译系统。其中目标语言到源语言的翻译系统负责生成伪数据，并且通过迭代使得生成的伪数据的质量越来越高，我们通过在测试集上的性能表现来确定模型通过迭代性能越来越好，并假定模型性能的提升在生成伪数据时生成的质量也越来越高。两个系统性能表现如表 4.6 和表 4.7 所示，其中汉蒙系统使用 multi_bleu.perl 脚本测 BLEU 值。表格中，“CM_”开头的系统表示汉蒙系统，“Iter_n”表示迭代 n 次。“MC_”开头的系统表示蒙汉系统，迭代表示与汉蒙系统类似。第 n 次迭代的系统是由第 n-1 次迭代的汉蒙系统所生成伪数据进行训练的。

表 4.7 使用伪数据训练的蒙-汉系统性能

Table 4.7 The performance of M-C system when using pseudo data

系统	数据	BLEU-4		BLEU-5	
		8k-steps	100k-steps	8k-steps	100k-steps
MC_Baseline	19w(Original)	37.24	36.67	32.18	31.68
MC_Iter_1	120w(Decode by CM_Baseline)	32.90	39.88	27.40	34.62
MC_Iter_2	120w(Decode by CM_Iter_1)	33.50	39.97	27.84	34.73
MC_Iter_3	120w(Decode by CM_Iter_2)	32.97	39.88	27.36	34.55
MC_Iter_4	120w(Decode by CM_Iter_3)	33.23	39.65	27.63	34.35

4.3.3 结果分析

本节我们将从上一节所得出的实验结果进行分析总结。使用少量数据训练反向系统并将大量单语语料翻译后生成了约 250 万伪数据，我们分别使用其中的 100 万、150 万和 200 万伪数据与高质量双语数据融合并预训练模型，通过上节表格可以看出直接使用混合训练数据时模型性能会比基线低很多（例如在汉蒙和蒙汉系统中我们使用混合数据更新模型 8 万次时模型性能分别为 21.18 的 BLEU-4 分和 27.40 的 BLEU-5 分，相比基线 27.22 和 32.18 分别低了 6.04 个 BLEU-4 分和 4.96 分），这表明我们只使用大量伪数据和高质量数据混合的数据进行模型的训练是没有效果的，虽然数据量增大了很多但是数据的质降低了很多，导致模型最终的模型性能急剧下降。而经过使用高质量双语数据进行微调训练之后模型的性能却比基线要分别高出 2.02 个 BLEU-4 分和 2.94 个 BLEU-5 分。当我们使用 150 万和 200 万伪数据进行同样的训练方法时却发现模型性能的提升并不大，在蒙汉系统中最高只比使用 100 万伪数据的情况要高出 0.08 个 BLEU-5 分，基本属于统一水平。从这个实验当中我们总结出两个结论，即第一，我们使用的伪数据和高质量数据融合的数据预训练模型，后使用高质量数据进行微调的训练方法是有效的，且对模型性能的提升由非常大的帮助；第二，我们一味地增加伪数据的量是无法进一步增加模型的性能，如果希望使用伪数据来增加模型性能应该从数据的质做一些研究。

从上一个结论中我们进一步想到将提高伪数据的质是否对模型性能有帮助这个点，并将上述方法运用于反向翻译系统——汉蒙系统的训练上，使其迭代训练越来越好的翻译系统，同时翻译出越来越好的伪数据。从图 4.5 可以看出，两个模型不使用高质量数据微调，只使用伪数据和高质量数据的融合数据训练时，随着迭代次数的增加，汉蒙系统的性能越来越好，表明通过迭代伪数据的质量越来越好；而蒙汉系统的性能浮动并不大，有个比较合理的解释是由于源语言数据是伪数据，而一点点伪数据的质的提高可能并不对模型性能的提升有较大的帮助。

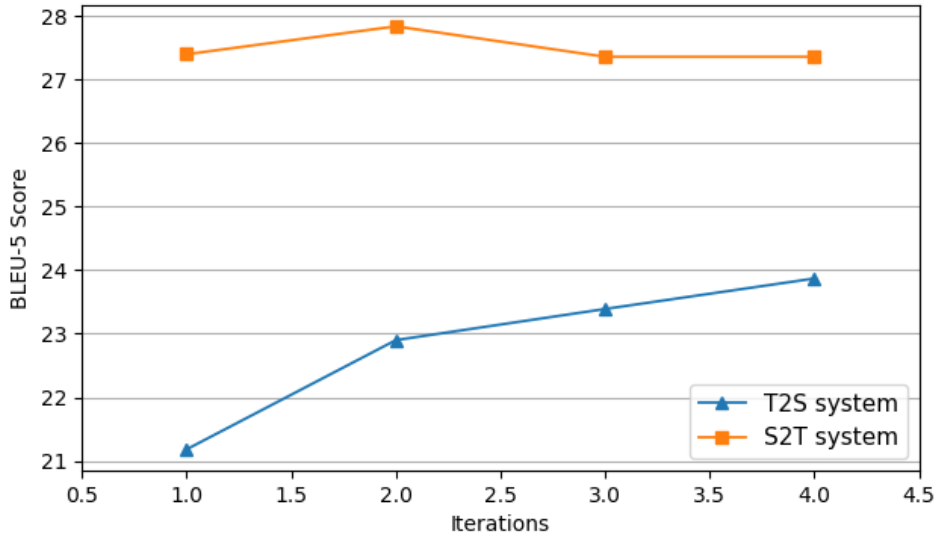

图 4.5 随着迭代两个系统在 80k 次更新时性能的变化

Fig. 4.5 The performance of two systems when update 80k times

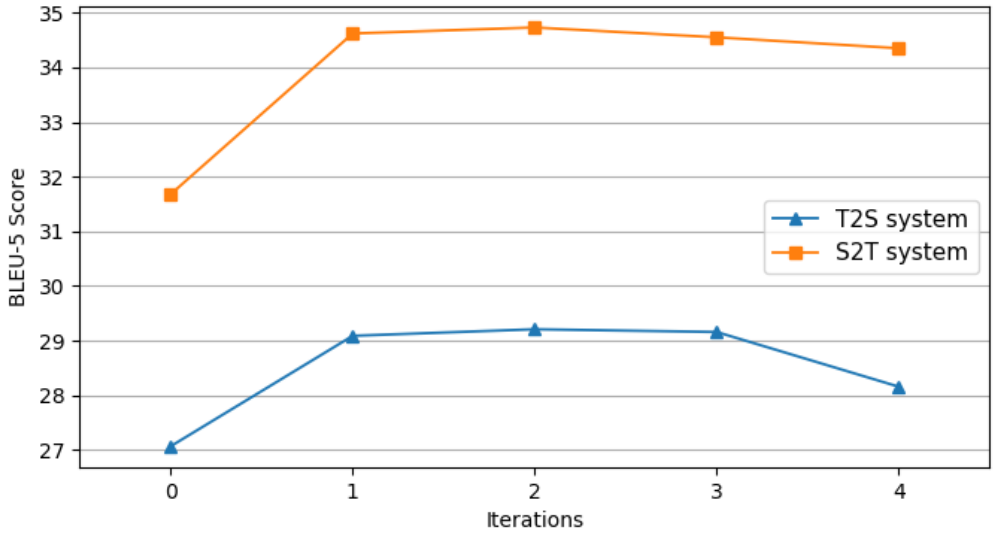

图 4.6 随着迭代两个系统在 100k 次更新时性能的变化

Fig. 4.6 The performance of two systems when updates 100k times

在预训练之后使用高质量数据做微调训练的结果随迭代次数的变化图如图 4.6 所示，我们发现迭代一次的时候模型性能有较大的提升，但是随着迭代次数的增加性能提升并不明显，反而开始下降。其中，蒙汉系统使用基线系统和第一次迭代系统所生成的伪数据时性能增长较多，使用基线系统生成的伪数据时相比基线系统有明显的增长，但是使用第一次迭代的系统时，虽然性能有所增长，但是增幅非常小。而此后随着系统的迭代性能并没有明显的增长，反而略微下降。这一实验现象表明，模型性能并不会随着伪数据质量的提高而总是增加，至少使用预训练与微调训练的方式无法增加模型性能。

在图 4.7 中，我们可以清楚的看到，虽然 80k 次更新时模型性能一直稳步增加，但是微调训练时起点的模型性能越高并不一定微调训练之后的模型性能越好，这一点也可以表明只增加伪数据的质虽然前期有所效果，但是并不是质量越高，对微调之后的模型系统性能越好。

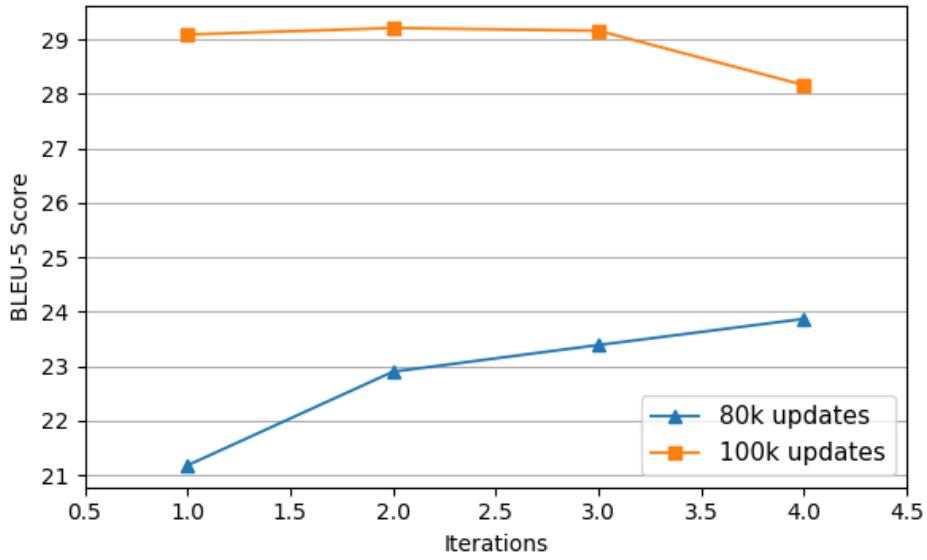

图 4.7 反向系统在 80k 和 100k 时系统性能的变化图

Fig. 4.7 The performance of back-translation system when updates 80k and 100k times

4.4 本章小结

本章主要介绍了使用伪数据训练模型方法的动机、方法的描述以及实验和实验相关的分析。

首先，我们介绍了使用该方法的动机——即高质量平行语料较少的情况下，通过反向翻译的方法增大平行数据量，再通过合理的训练方法可以提高模型性能。之后我们再从伪数据的选取方法、高质量数据和伪数据结合的训练方法上进行介绍。由于伪数据是使用少量高质量数据训练的翻译系统进行反向翻译的，通常情况下质量比较低，故我们需要使用一种筛选方法从大量低质量的伪数据中选取质量相对高的数据。其次，利用筛选过的低质量伪数据与高质量双语数据进行融合，并通过预训练以及微调的方法进行模型的训练，以提高模型性能的目的。通过实验我们发现，增大伪数据的量不能再提高模型性能时，我们通过提高伪数据的质来进一步提高模型性能，为此，我们提出迭代生成伪数据的方法生成伪数据，并使模型得到进一步的提高。最后，我们通过实验验证并分析其结果。

第 5 章 Decoder 与 LM 融合解码方法

5.1 研究动机

在第 4 章，我们使用单语语料生成伪数据，在模型训练阶段做了一些研究，最后的实验结果也表明，在只有少量双语平行数据的情况下使用单语语料反向翻译生成伪数据的方法对模型性能提升有很大的帮助。在这一章，我们将单语语料运用于模型解码阶段，利用单语语料的另一个有利于提升神经机器翻译模型的点。

本章利用单语语料的方法有两个重要的原因：首先，因为少量的双语数据不能够很好的训练一个好的解码器，使解码器不能够很充分的训练出一个强的表达能力，并且神经机器翻译系统的性能主要体现在输出的译文上，而神经机器翻译的解码器则与译文的输出有着直接的联系，所以在少量平行双语数据的情况下神经机器翻译的效果得不到一个很好的效果。其次，大量单语语料虽然是未标注的，但是其中却包含着很多丰富、可用的信息可以供神经机器翻译使用。而想使用未标注的大量单语语料则离不开语言模型的使用。通过前面的介绍，我们知道神经机器翻译使用“编码器-解码器”框架，且编码器或者解码器则是由神经语言模型构成的，这使得神经语言模型与神经机器翻译的融合有了可能。

从上述两个原因：1）数据稀缺导致模型性能不够好，未标注的单语语料隐含的丰富的、可利用的信息可被挖掘；2）神经机器翻译和神经语言模型可融合性，我们希望使用语言模型辅助神经机器翻译，在神经机器翻译进行解码时使用神经语言模型提供的丰富信息。

5.2 方法描述

5.2.1 Decoder 与 LM 融合方法

基于神经网络的神经机器翻译模型当平行数据的数据量较少时通常不能得到很好的效果，在前面的章节，我们使用单语数据在训练阶段通过加入伪数据的方法提升模型的性能，本节的方法主要是将单语语料运用于解码过程中。我们使用的方法主要是将使用无标注的单语语料训练的语言模型与神经机器翻译解码器端进行融合。为了达到更好的融合的效果，我们使用基于自注意力的神经语言模型，该模型与神经机器翻译的解码器类似，区别在于不再有与编码器端输出的上下文向量进行对齐的操作。

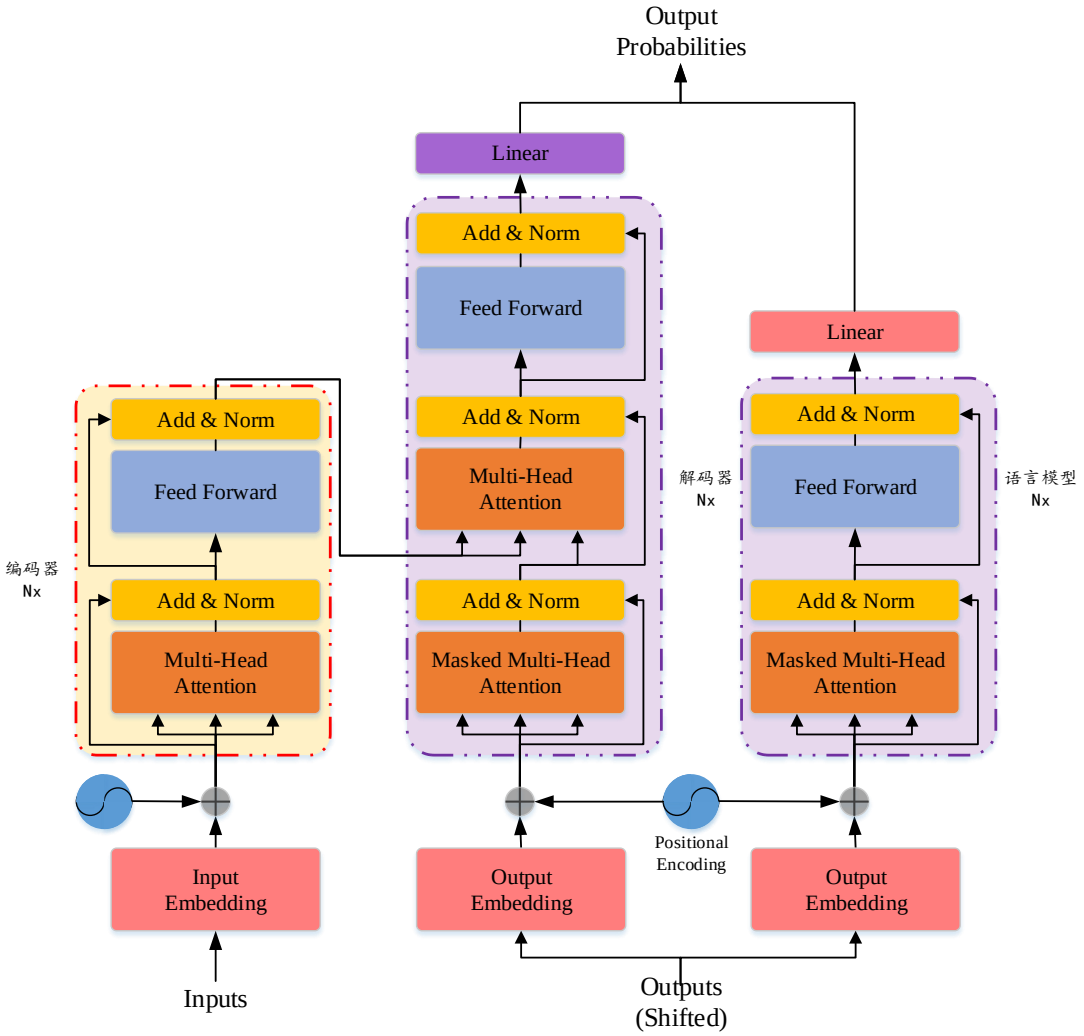

图 5.1Decoder 与 LM 融合模型结构

Fig. 5.1 The structure of fusion of Decoder and LM

首先，我们使用平行双语语料训练我们的神经机器翻译模型，以及使用大量单语语料训练神经语言模型。训练模型时，为了能够使模型能够融合，目标端的词表使用同一个词表，即要确保神经机器翻译的目标端和神经语言模型的词表、词向量以及每个词所对应的编号是严格相同的。在解码过程中，源语言首先经过神经机器翻译的编码端，经过编码端输出该句子表示语义信息的上下文向量，后此上下文向量会被神经机器翻译的解码端读入，而不影响神经语言模型的计算。编码器计算完之后神经机器翻译的解码端以及语言模型同时计算第一个输入词<PAD>的输出结果，即下一个词的分布，此时由于<PAD>词向量的值为全 0，所以神经语言模型输出的分布是平均分布，不具有任何决策的能力，而神经机器翻译的解码端由于与编码器输出的上下文向量进行对齐计算，将神经机器翻译模型的分布与神经语言模型的分布结合最后的结果取决于神经机器翻译模型的输出。而得到第一个词之后，其余的输出结果取决于神经机器翻译模型与神经语言

模型共同的分布。在该方法中，神经机器翻译模型与神经语言模型单独进行计算，每个模型分别给出自己所得的分布结果，并最后将结果结合。结合时，我们通过系数来确定每个模型的参与程度，通常情况下系数的和为一，在时间步$t = k$时得到输出值y_t公式如公式 5.1 所示，式中β是超参数，可以根据模型在检验集上的得分调整。

$$\log\big(p(y_t = k)\big) = (1-\beta)\cdot\log\big(p_{TM}(y_t = k)\big) + \beta\cdot\log(p_{LM}(y_t = k)) \quad (5.1)$$

模型结构图可参考图 5.1，从图中我们知道神经语言模型由于不需要与神经机器翻译的编码端做信息对齐计算，故网络中每一块的子层只有两个，与编码端一样先进行自对齐计算后经过前馈神经网络计算。

5.2.2 基于权重共享的 Decoder 与 LM 融合方法

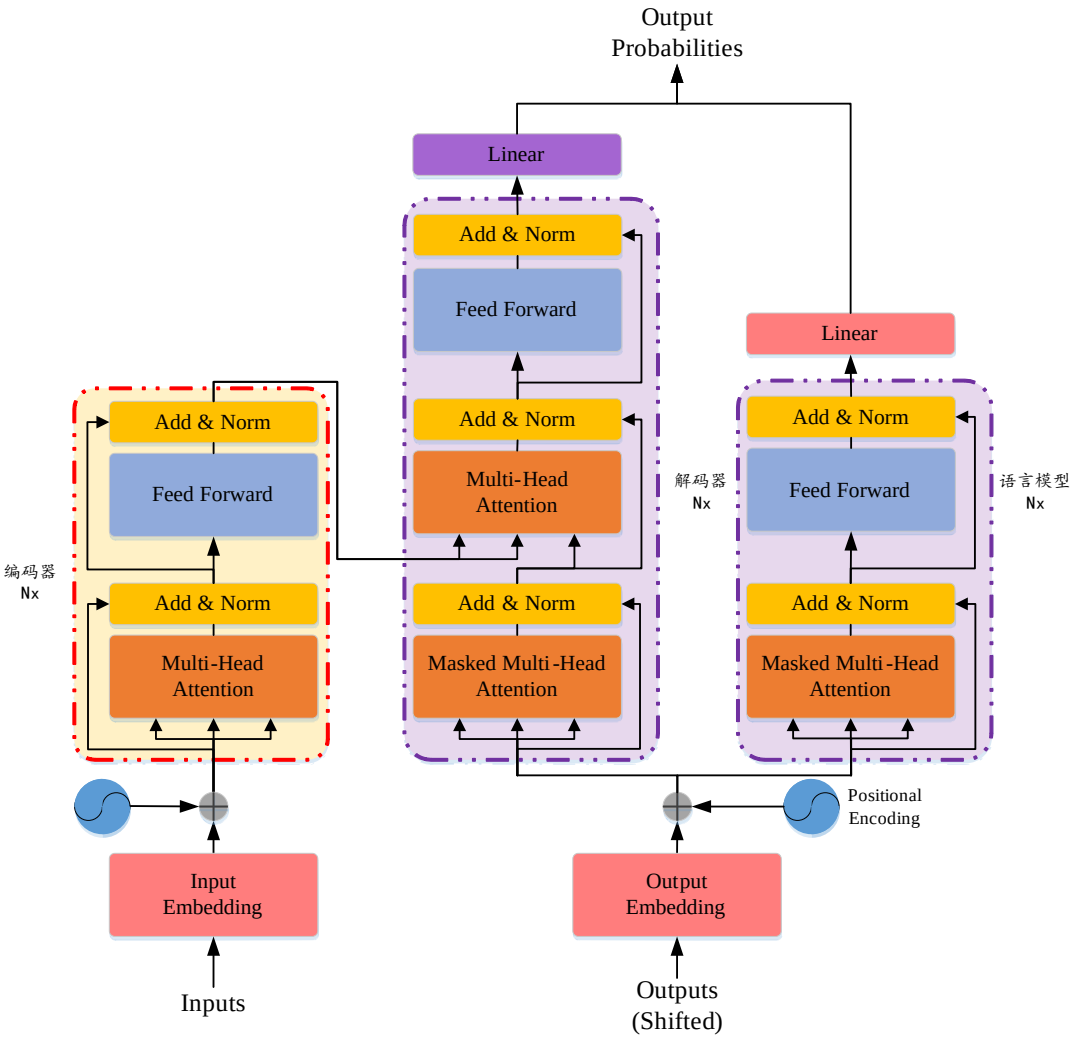

图 5.2 基于共享权重的 Decoder 与 LM 融合模型结构

Fig. 5.2 The structure of weight-shared based fusion of Decoder and LM

在上述方法中，我们将神经机器翻译与神经语言模型进行结合，主要是希望利用单语语料的信息，训练出好的语言模型，使其辅助神经机器翻译模型的解码端输出更优的

译文结果。通常情况下这是可取的，但是这种方法我们无法将神经语言模型的系数调得过大，因为神经语言模型虽然是使用目标端的大量单语语料训练，但是由于对源语言端不敏感，导致神经语言模型的分布与神经机器翻译的分布有较大的不同。如果这时我们将神经语言模型输出结果的系数过大，容易导致译文是流畅的目标端语言，但是不一定是源语言对应的译文，进而导致翻译系统的整个译文结果比较差。

为了缓解这个问题，我们想到一种将权重共享的方法，希望尽量使得两个模型的分布相似。在该方法中，我们将神经机器翻译模型训练好的词向量和 softmax 权重矩阵保存起来，在训练神经语言模型时其词向量则使用神经机器翻译模型保存权重矩阵。由于神经语言模型使用共享参数——即神经语言模型的词向量权重矩阵以及 softmax 权重矩阵使用同一个参数矩阵，故读取神经机器翻译所保存的矩阵参数时可以使用不同的参数矩阵。在神经语言模型训练过程中，神经机器翻译训练好的参数矩阵则不参与训练，只更新神经语言模型其他部分的网络层参数矩阵。这样做的目的主要是希望神经语言模型的其他可训练参数能够根据神经机器翻译模型的参数矩阵进行调整，以达到神经语言模型向神经机器翻译模型的输出分布靠拢。基于共享权重的解码端与神经语言模型融合的模型结构如图 5.2 所示。

5.3 实验

5.3.1 实验设置

在此项实验中，我们在源语言端和目标语言端都用 BPE 分词方法作为预处理方式，在检测系统性能方面，我们使用 BLEU-5 作为评价标准，使用的自动评价脚本是 mteval-v1.3a.perl，这个脚本是公开的脚本。

表 5.1 蒙汉平行语料及中文单语语料信息

Table 5.1 Information of Mongolian-Chinese corpus and Chinese corpus

数据集	领域	数据量	提供单位	备注
ICT-MC-corpus-CWMT2017	新闻	30007	中科院计算所	
IIM-CWMT2015	新闻	1682	中科院合肥机械所	
IMU-corpus-CWMT2017	政府文件	100001	内蒙古大学	
IMU-CWMT2013	口语、新闻	104790	内蒙古大学	
IMU-CWMT2015	口语、新闻	24978	内蒙古大学	
IMU-dev-mnzh-CWMT2017	口语、新闻	1000	内蒙古大学	4 个参考译文
IMU-dev-mnzh-CWMT2018	口语	1001	内蒙古大学	测试集
XMU	新闻	7423867	-	中文单语语料

使用的蒙汉平行语料是 CWMT 机器翻译评测官方提供的数据，该数据大约有 26 万

句对，其中中文语料约有 323 万中文词，最长句长为 132 个词，蒙语文语料约有 494 万字，最长句长为 264 个字。采用的语料主要提供方有中国科学院计算机技术研究所、内蒙古大学、中国科学院合肥智能机械研究所，其详细信息如表 5.1 所示。

使用的测试集为 IMU-dev-mnzh-CWMT2018 数据集，其中包含有 1001 句源语言，有 4 个参考译文。其中 XMU 语料是中文单语语料，是新华网关于新闻领域的单语数据，约有 700 多万句，在实验中，我们最多使用了 200 万单语语料进行反向翻译。在预处理数据时，在源语言端和目标语言端使用的 BPE 合并次数统一为 16000 次。

表 5.2 transformer_base_single_gpu 参数集参数值信息

Table 5.2 Information of transformer_base_single_gpu param-set

参数	值
隐层大小	512
batch 大小	8192
Initializer	uniform_unit_scaling
学习率	0.1
warmup 步数	16000
层数	6
标签平滑	0.1
过滤器大小	2048
头数	8
前馈层激活函数	ReLU
残差网络 dropout	0.1
Attention dropout	0.0
ReLU dropout	0.0
SGD 算法	Adam
Adam_epsilon	1e-9
Adam_beta1	0.9
Adam_beta2	0.98
最大句长	256
更新次数	80000
微调次数	20000

在系统实现方面，机器翻译模型训练时我们使用在谷歌公司开源的 Tensor2Tensor 开源工具包的基础上进行修改之后的系统。Tensor2Tensor 开源库是由同样由谷歌公司开发的深度学习框架 Tensorflow 进行搭建的。我们在做实验室使用的版本是 Tensor2Tensor-1.6.3，故本文中使用的结构在此版本基础上进行改进。

开源的基于自注意力机制的神经机器翻译系统 Tensor2Tensor 针对不同的实验需求和平台情况，预先封装好若干组不同的参数集合。在本实验中我们使用的模型参数集固定为 transformer_base_single_gpu 集。参数集中具体的参数值如表 5.2 所示。该参数集主

要是在搭建基准系统且使用单个显卡的时候使用的预先定义好的参数集，其中 batch 大小设置为 8 倍的 1024，表示每一次更新时所读取的源语言端或目标语言端最大的词的个数。也就是说，每一次更新时所读取的词数是固定的，而句子个数不固定。Warmup 为 16000 表示在更新次数达 16000 次以前学习率线性增长，超过 16000 次更新之后学习率指数下降，以保证模型能在保证收敛速度的情况下能够很好的收敛。更新次数设置为 80000 次，在 batch 为 8192 的情况下能保证模型训练 40 轮（即整个训练集可以从头到尾扫过 40 次）以上。

在语言模型网络结构实现方面我们使用 Tensor2Tensor 开源库中基于自注意力的语言模型模块。为了保持神经语言模型与神经机器翻译模型的网络结构足够相似我们语言模型使用的参数集同样使用 transformer_base 参数集，其具体信息如表 5.3 所示。

表 5.3 神经语言模型参数集

Table 5.3 Hyper-parameters of language model

参数	值
隐层大小	512
batch 大小	4096
Initializer	uniform_unit_scaling
学习率	0.1
warmup 步数	16000
层数	6
标签平滑	0.1
过滤器大小	2048
头数	8
前馈层激活函数	ReLU
残差网络 dropout	0.1
Attention dropout	0.0
ReLU dropout	0.0
SGD 算法	Adam
Adam_epsilon	1e-9
Adam_beta1	0.9
Adam_beta2	0.98
最大句长	256
更新次数	30000

神经语言模型参数设置上与神经机器翻译相似，但是网络结构上只有两个子层，同时由于语言模型行对容易训练，故在实验中，我们只更新 3 万次。

另外我们在实际中使用的模型采用了 checkpoint ensemble 的方法，集成了 10 个最近保存的模型参数，对这些模型参数进行平均，这样做的目的是提高实验的准确性，减少单一模型性能上的偶然因素。这样做可以很大程度上可以使得实验结果更稳定，得到

的曲线相对来说更平滑，在后续的实验过程中我们围绕模型集成的结果进行分析。

5.3.2 实验结果

本节主要展示神经机器翻译解码端与神经语言模型融合的实验结果，我们首先将神经机器翻译模型与神经语言模型直接融合，权重不共享时神经语言模型系数对于模型性能的影响。

基于自注意力的神经语言模型在标准集 Penn Tree Bank（或 PTB）训练集上的困惑度（Perplexity 或 PPL）为 45.32，在 PTB 校验集上的困惑度为 65.25，在高质量双语语料目标端单语训练集上的困惑度为 78.46，在训练集上的困惑度为 23.61，经验证神经语言模型的实验设置是有效的，并达到基线标准。由于神经机器翻译模型以及神经语言模型是不同的任务，且为了不让神经语言模型参与太多决策，故我们将神经语言模型的系数从 0.025 调至 0.3。将超参数神经语言模型的系数β设置的比较小一点，具体实验结果如下表 5.4 所示。

表 5.4 语言模型系数不同情况下的 BLEU 得分

Table 5.4 BLEU scores when different beta value

系统	系数β	测试集	
		BLEU-4	BLEU-5
Baseline		36.67	31.68
Baseline + LM(β = 0.3)	0.3	36.83	32.03
Baseline + LM(β = 0.25)	0.25	37.45	32.54
Baseline + LM(β = 0.20)	0.2	37.59	32.63
Baseline + LM(β = 0.175)	0.175	37.43	32.49
Baseline + LM(β = 0.15)	0.15	37.41	32.43
Baseline + LM(β = 0.125)	0.125	37.41	32.41
Baseline + LM(β = 0.1)	0.1	37.45	32.44
Baseline + LM(β = 0.075)	0.075	37.25	32.55
Baseline + LM(β = 0.05)	0.05	36.91	31.89
Baseline + LM(β = 0.025)	0.025	36.68	31.72

我们将神经语言模型系数最高调至 0.3，也就是说在最终的决策中，神经语言模型的有不到三分之一的决策能力，由于主要任务是机器翻译任务，所以我们不将系数调至过大。使用双语语料目标端单语数据是希望语言模型与神经机器翻译模型能学习相同领域的内容，但是数据量相对少。故我们使用 XMU 大量单语语料训练语言模型，希望能够得到更好的语言模型性能，为了缓解领域的问题，我们将高质量双语数据目标端单语语料并入大量单语语料，并训练语言模型。使用该训练集训练的神经语言模型在同一个校验集上的困惑度为 80.39，在训练集上的困惑度为 47.28，可以达到基线标准，其实验

结果如表 5.5 所示。

表 5.5 使用大量语料训练语言模型时融合系统 BLEU 得分
Table 5.5 BLEU scores when using language model trained by big data

系统	系数β	测试集	
		BLEU-4	BLEU-5
Baseline		36.67	31.68
Baseline + LM(β = 0.3)	0.3	37.14	32.24
Baseline + LM(β = 0.25)	0.25	37.47	32.51
Baseline + LM(β = 0.20)	0.2	37.43	32.49
Baseline + LM(β = 0.175)	0.175	37.52	32.55
Baseline + LM(β = 0.15)	0.15	37.47	32.44
Baseline + LM(β = 0.125)	0.125	37.29	32.28
Baseline + LM(β = 0.1)	0.1	37.26	32.27
Baseline + LM(β = 0.075)	0.075	37.21	32.22
Baseline + LM(β = 0.05)	0.05	37.07	32.10
Baseline + LM(β = 0.025)	0.025	36.77	31.80

除此之外，我们尝试将神经机器翻译模型的词向量共享的实验，但是由于神经语言模型使用的是词向量与输出层权重矩阵使用同一个矩阵参数，导致将神经机器翻译的词向量用于神经语言模型的输出层时性能下降。故我们主要将神经机器翻译的输出层的矩阵共享给神经语言模型。基于权重共享的解码端与语言模型融合的在语言模型参与度不同情况下的 BLEU 值如表 5.6 所示。

表 5.6 基于权重共享的融合方法在语言模型系数不同情况下的 BLEU 得分
Table 5.6 BLEU scores of weight-shared based fusion system when different beta value

系统	系数β	测试集	
		BLEU-4	BLEU-5
Baseline		36.67	31.68
Baseline + LM(β = 0.3)	0.3	36.97	32.06
Baseline + LM(β = 0.25)	0.25	37.53	32.56
Baseline + LM(β = 0.20)	0.2	37.67	32.72
Baseline + LM(β = 0.175)	0.175	37.70	32.75
Baseline + LM(β = 0.15)	0.15	37.40	32.49
Baseline + LM(β = 0.125)	0.125	37.50	32.58
Baseline + LM(β = 0.1)	0.1	37.43	32.49
Baseline + LM(β = 0.075)	0.075	37.43	32.50
Baseline + LM(β = 0.05)	0.05	36.94	31.91
Baseline + LM(β = 0.025)	0.025	36.71	31.75

我们将上述方法运用于使用伪数据训练的神经机器翻译模型上。我们使用伪数据与高质量数据混合的数据训练神经机器翻译模型，同时用这批数据的目标端单语语料训练

神经语言模型，神经机器翻译模型训练方法为第四章中介绍的迭代生成伪数据的方法。训练好神经机器翻译模型后，对比其在不同语言模型系数时的性能表现。我们使用的神经机器翻译模型是用经过两次迭代之后生成的伪数据训练，其模型效果在迭代的方法中取得最好的成绩。具体使用伪数据训练的神经机器翻译模型与语言模型融合的实验结果如下表 5.7 所示。

表 5.7 使用伪数据训练的模型与语言模型融合时系统 BLEU 得分

Table 5.7 BLEU scores of fusion system trained by pseudo data and fine-tuning

系统	系数β	测试集	
		BLEU-4	BLEU-5
Baseline		39.97	34.73
Baseline + LM(β = 0.3)	0.3	37.56	32.28
Baseline + LM(β = 0.25)	0.25	38.21	32.99
Baseline + LM(β = 0.20)	0.2	39.39	34.16
Baseline + LM(β = 0.175)	0.175	39.47	34.25
Baseline + LM(β = 0.15)	0.15	39.59	34.35
Baseline + LM(β = 0.125)	0.125	39.76	34.51
Baseline + LM(β = 0.1)	0.1	40.04	34.79
Baseline + LM(β = 0.075)	0.075	40.04	34.81
Baseline + LM(β = 0.05)	0.05	39.96	34.72
Baseline + LM(β = 0.025)	0.025	40.06	34.83
Baseline + LM(β = 0.01)	0.01	40.13	34.88
Baseline + LM(β = 0.005)	0.005	40.02	34.77

5.3.3 结果分析

通过上述将神经语言模型与解码端融合解码的实验以及实验结果，我们可以得到一些经验性的结论，下面我们分几部分进行讲解。

（1）首先，我们将基线与经过神经语言模型融合的系统结果进行对比，发现在数据量较少时训练的神经机器翻译模型，经过与神经语言模型融合的系统一般情况下能得到比较明显的 BLEU 值的提升，使用权重共享的方法性能提升最多，模型性能提升有 1.04 个 BLEU-5 分值，其在系数值为 0.2 时达到最高分，表示当语言模型参与度是 20% 时对模型的性能提升最有帮助。详细结果如表 5.8 所示，将基线与进行融合的系统相比，此表格种展示的结果均表示当不同模型组合尝试不同的系数时能达到的最高分的情况。

表格 5.8 中，括号里的数字表示用多大的数据量训练模型，例如 Baseline(20w)表示使用高质量的双语数据训练翻译模型，而 Baseline(120w)表示使用伪数据与高质量数据合并的数据进行模型的训练。语言模型则使用双语语料的目标端语料进行训练，所以数据量上与训练神经机器翻译模型的数据保持一致。从表格中我们可以看出，在使用小数

据量训练的机器翻译系统上进行模型融合，会提高 0.85~1.00 的 BLEU-4 分，提高 0.87~1.04 的 BLEU-5 分，而使用大量数据训练的神经机器翻译模型中约能提高 0.16 的 BLEU-4 分以及 0.15 的 BLEU-5 分。

表 5.8 基线与融合系统性能比较

Table 5.8 The comparation of baseline and fusion systems

系统	系数β	测试集	
		BLEU-4	BLEU-5
Baseline(20w)	-	36.67	31.68
Baseline(20w)+LM(20w)	0.2	37.59(+0.92)	32.63(+0.95)
Baseline(20w)+LM(20w)+shared	0.2	37.67(+1.00)	32.72(+1.04)
Baseline(20w)+LM(120w)	0.175	37.52(+0.85)	32.55(+0.87)
Baseline(120w)	-	39.97	34.73
Baseline(120w)+LM(120w)	0.01	40.13(+0.16)	34.88(+0.15)

同时我们可以看出，使用少量双语数据训练翻译模型时这种方法能提高较多的分数，但使用大量数据训练（伪数据和微调的方法训练）时提升并不明显。由于使用大量数据时，我们经过其他的训练方法，使得模型性能提高，进而导致模型的整体基线提高较多，压缩了模型性能进一步提升的空间。而另一方面，在模型基线已经很好的情况下与语言模型融合的方法能进一步提高模型性能，表明这种方法是可取的，是有效的。同时，上述对比表格中，我们发现使用大数据量训练的语言模型，与使用少量双语数据训练的翻译模型的融合并未能得到较好的效果，其中一个较好的解释是领域问题导致了两个模型的差异性增大，也表明，语言模型使用的数据如果与机器翻译模型训练的数据领域相似或者完全使用翻译模型训练的双语数据的目标端数据则会有较好的结果。

（2）我们比较通过设置不同的神经语言模型系数β来控制语言模型的参与度时模型性能的表现。通过将语言模型系数修改不同的值，使其对翻译模型的结果有不同程度的影响。通过柱状图 5.3 我们可以看出，语言模型系数β值较为合理时模型性能会有所上升（柱状图显示不管是哪种模型的融合，在语言模型的参与程度在 17.5%~20%区间时模型性能有显著上升，但是过大或过小性能都有所下降），但是调过大会导致模型性能的下降，这是因为不是翻译任务的语言模型参与比重过大，导致最终译文的输出偏向语言模型的输出，而不是翻译模型的输出，这使得译文虽然能够输出流畅、易读的文本，但是变得对源语言不敏感，会直接影响翻译性能。而调得过小，则容易导致语言模型参与度过小，与不使用模型融合时性能相似（需要指出得是，当语言模型系数值为零时即为没有融合的翻译系统），故性能有些许下降。其中，当最终模型性能最好时，神经语言模型的系数值在 0.175 和 0.2 左右，表示当语言模型有百分之二十的决策权时翻译模型

与语言模型的融合能得到最好的效果。

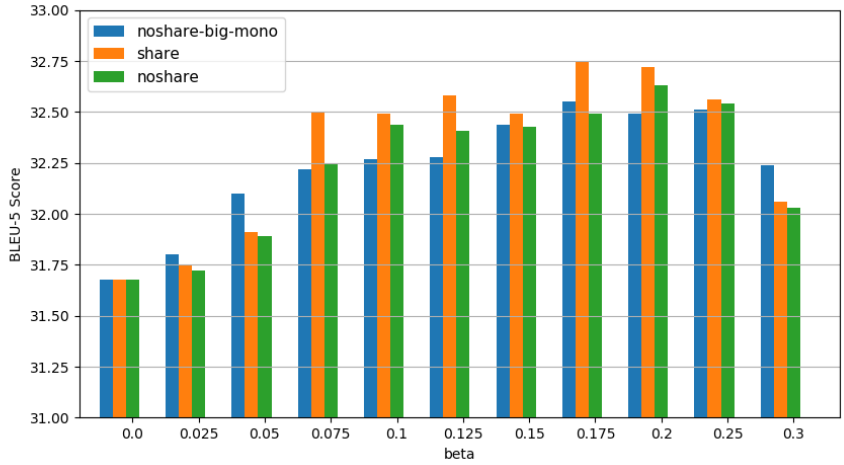

图 5.3 不同融合模型的β值不同情况下的 BLEU 得分情况

Fig. 5.3 BLEU scores of different fusion method when beta changing

同时，通过图 5.4 我们也可以看到，在翻译模型性能较好的情况下（即使用大量伪数据通过迭代和微调训练的模型），语言模型的参与度与其他情况不一样——当β值较小时略有提升，但是随着语言模型系数增大，模型性能有显著下降，甚至要小于基线系统。这表明，当翻译模型的性能足够好时如果语言模型的参与程度较多则会导致模型性能的下降，而且模型性能甚至会小于基线性能，但是如有少量语言模型的参与对最终模型性能的提升有一定的帮助。

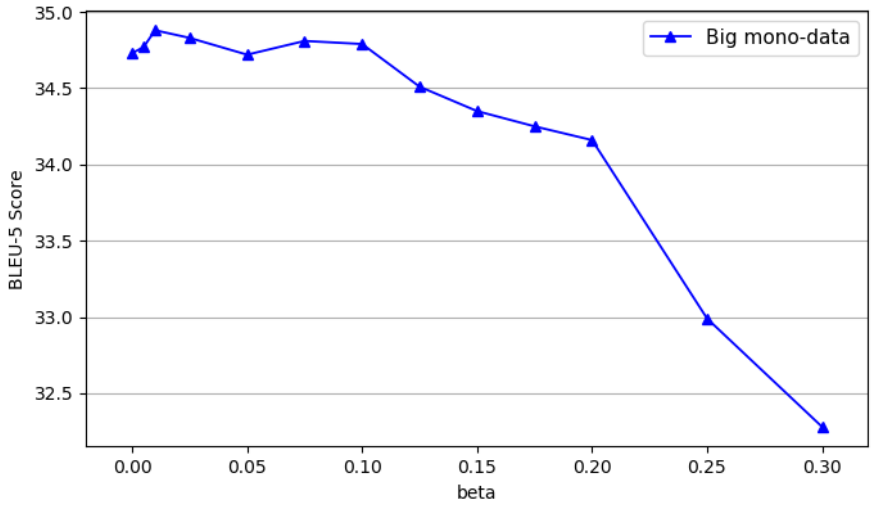

图 5.4 使用大量语料训练翻译模型时融合系统得分情况

Fig. 5.4 BLEU scores when using language model trained by big data

（3）最后，我们将有参数共享和没有参数共享的方法进行对比，用来确定权重共享是否能为系统性能的提升带来帮助，是否会使两个任务的分布相近。我们最初希望通

过权重共享希望使得神经机器翻译模型的分布与神经语言模型的分布尽可能的相似，以此来提升模型性能的提升。但实验结果显示，即图 5.5 所示，虽然将神经语言模型系数的值调到特定值可以提高模型的性能，在语言模型系数分别在 0175、0.125 和 0.075 时比没有权重共享的融合方法高 0.25~0.26 的 BLEU-5 分相差较高，但是大部分情况下模型性能与不用权重共享的方法相似。在语言模型系数不同时，两个方法所得到的性能走势的规律是相似的，都是随着系数的增加性能慢慢增长，之后虽然有一定的下降但很快模型性能又增长，最后性能开始逐渐下降。

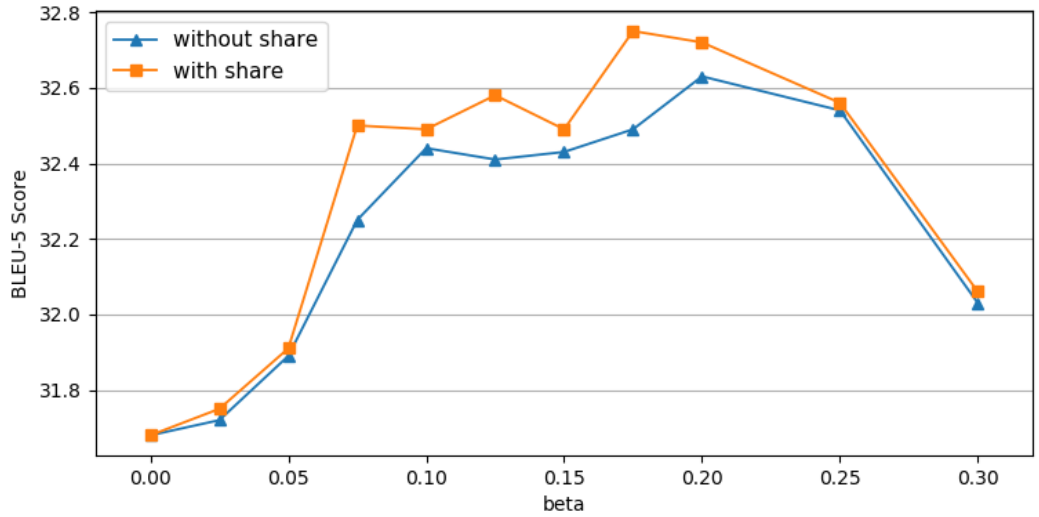

图 5.5 无共享的融合与权重共享的融合对比

Fig. 5.5 The comparation of weight-shared and not shared fusion systems

5.4 本章小结

本章主要介绍了神经机器翻译模型与神经语言模型通过融合进行解码方法的描述、实验及实验结果的分析等内容。

单语语料虽然是未标注的但是却有很多的信息可以供神经机器翻译模型利用，为此，我们使用单语语料训练语言模型，使语言模型的输出信息与神经机器翻译模型的输出信息进行结合，希望通过融合的方法可以使模型在解码过程中有更好的决策。

我们首先简单的将神经机器翻译模型与神经语言模型进行融合，只在输出概率分布时将神经机器翻译模型的概率分布与神经语言模型的分布进行插值操作。这种情况下，虽然可以很好的利用语言模型输出的信息，但是由于两个模型是完全不同的任务，容易导致两个模型所学的信息不一致，进而影响我们主要的模型——即翻译模型的性能。为了解决这个问题，我们又在前一个方法的基础上使用共享权重的方法，将神经机器翻译模型的词向量矩阵或 softmax 参数矩阵进行共享，并在训练神经语言模型的过程中将这

些参数固定不变，更新模型的其他参数，进而使神经语言模型与神经机器翻译模型的输出分布相似。

最后，我们通过实验将上述方法实现并验证。通过实验结果，我们分析得出，在系数一定的情况下，将使用单语语料训练过的语言模型与神经机器翻译模型融合解码有一定的性能提升，如果进一步将权重做一些共享，则会得到进一步的性能提升。

第 6 章 系统展示与集成

6.1 系统总体架构及部署

6.1.1 总体架构

系统总体架构如图 6.1 所示，系统采用 B/S（Browser/Service）模式。前端是简单的 Web 界面，负责获取用户的输入以及将结果输出给用户，用户通过按钮可以向服务器发送翻译的请求，并将输入原句子发送给服务器。服务器得到输入原句子之后会依次经过分词（首先经过基于词的分词，后经过字节对编码分子词工具），并将经过预处理的句子进行词到编号的转换，并送入基于 CUDA/C++实现的解码器中。解码器则根据网络结构进行自回归的方式解码并输出最终的译文。译文结果输出之后则要进行后处理操作，后处理操作主要包括删除多余的空格、英文单词的大小写还原以及标点符号的全角符号转半角符号等过程。

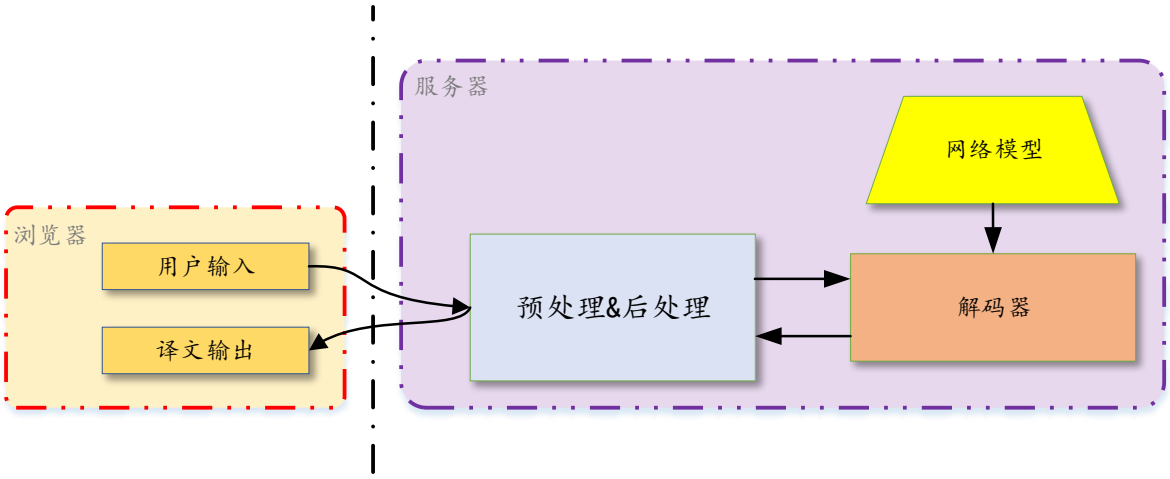

图 6.1 B/S 模式的蒙汉机器翻译系统
Fig. 6.1 The MT system of B/S mode

在实现时，我们前端 Web 界面使用 JSP 开发，功能相对简单（只有两个文本框和一个按钮，其中一个文本框负责获取用的输入，支持编辑，而另一个文本框只负责输出译文结果给用户，用户不可编辑文本）。界面通过 Java 本地接口（Java native interface 或 JNI）与后端神经网络解码器连接。蒙文分词工具、字节对编码分子词工具均由 C/C++编程语言自主开发，并且在运行性能上也都能达到标准速度。最后，神经语言模型的解码器是用 CUDA/C++开发，是根据 Tensor2Tensor 开源库中基于注意力的神经机器翻译

网络结构，以及运算方式同构。解码方法为单句解码，支持 Cache 缓存计算方法。该方法可以提高计算速度，将当前状态之前的信息通过保存在缓存里的方式减少计算注意力信息。即每次计算当前词的信息以及与 Cache 缓存的对齐信息。

6.1.2 系统部署

我们将神经机器翻译系统部署于实验室提供的单个物理 GPU 服务器上，该服务器拥有 4 块 NVIDIA TITANX 显卡，显存大小都为 12G。服务器上运行的系统为 Linux CentOS 7，64 位版本，Linux 内核版本为 3.10.0。运行的显卡驱动版本为 390.48，使用 CUDA-9 以及 CuDNN-7，详细信息如表 6.1 所示。

表 6.1 服务器配置

Table 6.1 Server configuration

硬件	说明
CPU	Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz
内存	64G
GPU	NVIDIA TITAN X
显存	12G
硬盘	3.5T

我们用到的服务器软件版本说明如表 6.2 所示。

表 6.2 服务器软件版本信息

Table 6.2 software versions of the server

软件	说明
系统	CentOS Linux release 7.2 (64-bit)
GCC	4.8.3
显卡驱动	NVIDIA-SMI 390.48
CUDA	8.0
Java	1.8.0_65 (64-bit)

6.2 系统的实现

统一计算架构（Compute Unified Device Architecture 或 CUDA）是由 NVIDIA 公司所推出的一种集成技术，通过这个技术，用户可以利用 NVIDIA 的 GeForce8 以后的 GPU 和较新的 Quadro GPU 进行并行程序运算。深度学习框架之前由于消耗大量的计算运算资源，使得其发展缓慢，在当时的人们看来，其大量的矩阵运算无法通过高效的运算完成。而 CUDA 的出现很大程度上解决了深度学习框架所需要大量的计算资源的需求，加快深度学习框架发展的同时也让人们相信深度学习神经网络的重要意义以及实现的可能。

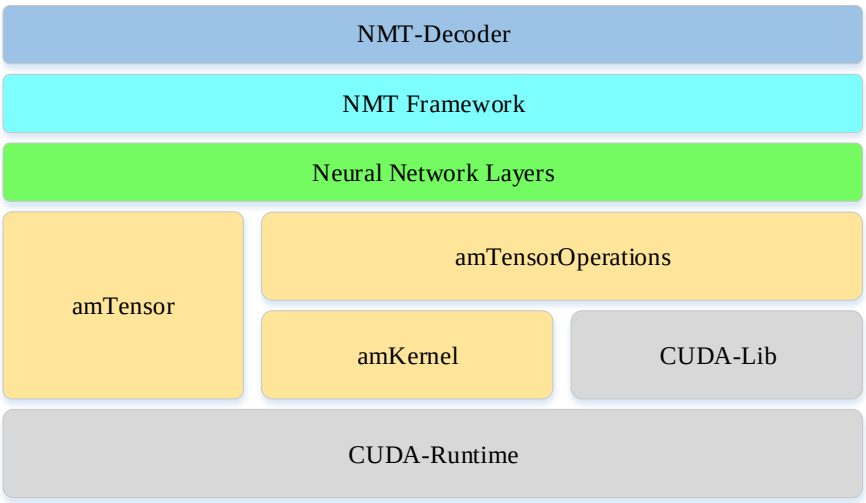

图 6.2 解码器架构图

Fig. 6.2 The architecture diagram of decoder

我们使用 CUDA9 版本编写基于自注意力机制的 C/C++版本解码器，其结构图如图 6.2 所示。首先我们是在 CUDA 官方提供的运行时接口 CUDA-RT（CUDA-RUNTIME API）的基础上进行编写的。CUDA-RT API 与显卡驱动很类似，但是驱动 API 提供了更细粒度的编码方式，尤其是在上下文和模块管理方面。但是，运行时 API 所有内核在初始化期间都会自动加载并在程序运行时保持加载的状态，而驱动内核启动实现起来要复杂得多。总体来讲，对于用户来说，使用运行时 API 相对使用驱动 API 要简单得多。运行时 API 提供了查找并读取设备信息，设置运算显卡，核函数的启动等基本的操作，但是并没有与神经网络直接相关的矩阵运算等相关接口。而在运行时接口基础上，英伟达官方又推出了 CUDA 运算库——基本线性代数子程序（Basic Linear Algebra Subprograms 或 CuBLAS）库，该库将一些矩阵运算进行封装，使用户更加方便学习以及运用。CuBLAS 包含了很多基本的矩阵运算操作，例如两个矩阵相乘，两个向量相乘或者矩阵和向量的相乘等，同时也包括了原子操作等接口。

基于 CUDA 运行时 API 以及 CuBLAS 计算库，我们首先实现了 Tensor、Kernel 以及 TensorOperations 操作。其中 Tensor 保存着神经网络所需要的矩阵的值，实现了深度学习网络中矩阵所能做的最基本的操作——保存和读取矩阵值，保存矩阵的形状、大小以及将矩阵值在内存和显存传输等功能。我们实现的 Tensor 可以最高实现 4 维矩阵张量乘法。Kernel 是程序在显卡上进行运算时调用的函数库，最基本的操作都包含在其中，比如矩阵的分割和合并操作、规约操作、Reshape 操作、Transpose 操作以及 Softmax 操作。这些是基于神经网络结构的机器翻译模型所需要的基本操作。我们将这些操作进一步进行封装，成为 TensorOperations 操作，即我们可以使用 TensorOperations 对张量进行

运算。至此，我们有了最基本的张量以及张量运算的操作。在实现神经机器翻译解码器前，我们又多加了一层网络层，网络层则封装了张量以及张量的运算，包含了层归一化网络层、自注意力网络层、前馈神经网络层、词向量层以及输出层等网络层。其中自注意力层包含前馈计算、归一化计算、归约计算，是基于自注意力的模型结构的核心部分。

6.3 系统前端展示

本系统前端界面包含了一个输入文本框，输出文本框以及翻译按钮。其中输入文本框为可编辑文本框，输入用户想翻译的源语言（蒙语）句子，之后用户通过点击翻译按钮可将输入的文本发送至后台服务器，并激活正在监听的服务程序，服务程序通过 JNI 调用预处理模块、神经网络解码器，将最后的输出结果返回给前端。前端通过输出文本框将神经机器翻译模型输出的结果展示给用户。至此，一次翻译结果流程结束，前端界面图如图 6.3 所示。

图 6.3 蒙汉翻译系统前端展示

Fig. 6.3 The interface of MC translation system

6.4 本章小结

本小节主要介绍了蒙汉神经机器翻译系统的系统架构、部署、系统的实现以及前端界面的展示等内容。系统的总体架构使用 B/S 模式，主要通过 Web 浏览器进行访问和发送请求。实现时前端 Web 页面展示使用 JSP 编程语言开发，功能相对简单（只有一个输入文本框、输出文本框以及一个翻译按钮）。前端通过 Java 本地接口（JNI）与后端神经网络解码器进行连接，后端神经网络解码器使用 CUDA/C++实现，重现了

Tensor2Tensor-1.6.3 版本的基于自注意力的神经机器翻译模型，包括其中的 Cache 缓存计算技术等。在翻译译文效果上与 Tensor2Tensor 版本保持一致，速度上目前与开源版本的解码器稍快一点。后介绍了 CUDA 解码器的具体实现架构，在英伟达官方提供的 API 基础上我们逐层实现了张量、张量操作方法，网络层计算，神经机器翻译网络层以及神经机器翻译框架层等计算层，最后实现神经机器翻译的解码器。

整个蒙汉系统的重点部分是后端解码器部分，前端部分功能相对简单，不是本系统的重点。

第 7 章 总结与展望

7.1 工作总结

近些年来随着神经网络技术的发展，硬件设备性能的进一步提高，基于神经网络的神经机器翻译技术在性能上有了很大的进步。神经机器翻译模型使用“编码器-解码器”框架，从基于循环神经网络的模型结构、基于卷积神经网络的模型结构到最新的基于自注意力的神经网络结构，每一次更新进步都会带来模型性能上的巨大的提升。最新的基于自注意力的神经网络结构由于网络本身独有的结构，使得在学习过程中减少长短时记忆的依赖，同时支持了模型在训练过程中并行的执行，这在最终的模型性能的提升和模型训练的成本上的降低都有较大的帮助。我们使用该网络结构完成蒙语-汉语机器翻译系统，由于蒙汉双语数据的稀缺，导致使用基于自注意力结构的网络结构并不能得到性能较好的系统。

我们知道，神经机器翻译模型的训练属于有监督的机器学习，而有监督的机器学习一般都需要大量的带有标注的数据，使得模型学习数据的分布规律，以便完成类似的任务。但是蒙汉双语数据量少，这使得很难得到性能较好的蒙汉翻译系统。同时，我们具有大量的目标端单语语料。对此，我们分别从数据的预处理、模型的训练过程以及模型的解码过程进行了研究和分析，希望分别在不使用单语语料和使用大量单语预料时提高最终蒙汉翻译系统的性能。

首先，我们在数据的预处理阶段做了研究，希望通过数据的预处理，减少数据的稀缺性。因为蒙语是一种黏着性语言，一般的词都是词根后缀接后缀而得出新的词，根据后缀的不同、缀接个数的不同会导致词的词性、时态以及意义上的改变。所以蒙语是一种词形丰富的语言。而词形的丰富会容易导致数据的稀缺，会对神经网络模型的学习带来很大的负面影响。同时我们为了希望尽可能的保持词语义的保留，我们提出高频率词以及子词混合的预处理方式对数据进行预处理，这种方法会首先将高频词保留，包括一个词中包含着的高频词部分，剩下一些低频词以及词缀部分，我们统一使用字节对编码的方法对其进行分子词处理。通过对比实验，我们发现分子词的方法对模型性能的提升有很大得帮助，而保留整个词能够尽可能的保留语义信息。最后的实验结果表明，当保留的高频词的个数保持在一定值时，高频词和分子词的预处理方法对模型的性能有进一步的提升。

其次，我们在模型的训练部分做了研究，因为我们通过实验发现，当训练好的模型

固定时，通过后处理的方法提升性能的空间已经很有限，所以我们需要在模型训练阶段对其进行改进。在实践中，我们首先使用混合数据预训练模型后高质量数据微调训练的方法，发现这种方法对模型性能的提升很有帮助，但我们进一步加大伪数据的数据量时发现单纯增加数据的量是没有办法进一步提高模型的性能，所以我们从伪数据的质量入手，希望通过提高伪数据的质量进而提高模型性能。我们将上述方法运用于反向翻译系统中，即汉蒙系统中，通过迭代将反向翻译系统的性能一步一步提高，使用最好的反向翻译系统生成伪数据供正向翻译系统的训练。实验中，我们发现，通过迭代的训练，只使用伪数据训练系统性能随迭代提升，但是经过微调训练发现其最终模型的性能虽然有所增长，但是提升的富度并不大。因此，我们得出结论使用微调训练时，起点模型的性能对于最终模型的性能提升帮助并不大。

此外，我们还把注意力放在了模型解码的过程中，希望在解码过程中利用单语语料的信息辅助模型的输出。为此，我们提出将基于自注意力的神经语言模型与神经机器翻译的解码器部分进行模型融合，这样在翻译模型进行决策时，语言模型可以辅助翻译模型进行决策。由于两个模型属于不同的任务，我们通过使用一个系数来确定语言模型的参与程度。通过实验我们发现，当语言模型参与度的系数合理时，对于模型的性能提升有非常大的帮助，同时实验结果表明，语言模型系数过大或过小都会导致模型性能的略微下降，甚至低于基线性能。当模型系数过小时与未融合的翻译模型趋于同一水平，需要提出的是，当语言模型系数为零时，就是我们的翻译系统。而语言模型系数的过大则会导致最终的输出与语言模型强相关，而翻译任务的权重就会大大减小，而我们恰恰需要的是翻译任务的结果。故，在语言模型系数合理时，这种方法对于模型性能的提升才很有帮助。

最后，我们通过将第二种方法（即使用单语语料做伪数据在训练阶段进行改进）与第三种方法（即将单语语料运用于训练语言模型，后将语言模型与翻译模型融合的方法）对比，发现将单语语料运用于生成伪数据，并在训练阶段做一些改进比与语言模型融合的方法更有效果。

本文中分别在三个方面提出三种方法，希望提高具有少量标注数据的蒙汉翻译系统的性能，分别为不使用单语语料的情况下对少量双语语料的预处理过程、使用单语语料生成伪数据，并在训练方法上做一些改进以及使用单语语料训练语言模型，并在蒙汉翻译系统解码阶段做与语言模型的模型融合的方法，最终使得翻译模型性能有较显著的提升。

7.2 创新点分析

本课题中最大的创新点在于：第一，在使用伪数据训练模型时提出通过伪数据的质量来提高模型的性能；第二，通过共享权重的方法将翻译模型的解码端与神经语言模型进行融合。首先，使用伪数据进行预训练后使用微调训练的方法对于模型的性能提升有非常大的帮助，而通过增加数据的量的方法不能进一步的增加模型的性能，对此我们将关注点聚焦于伪数据的质上，使得即使使用伪数据时对模型的性能也有较好的帮助。其次，使用语言模型与神经机器翻译解码端融合的方法可以利用单语语料的信息，而通过共享权重，可以使得两个不同的任务的分布尽可能相似，这对最终翻译任务的性能的提升是非常有帮助的。

7.3 未来工作

本文中提出了三种方法分别围绕机器翻译流程中的预处理过程、模型的训练过程以及解码过程做了一些改进。但是在我们做实验的过程当中，我们发现，在使用伪数据的训练过程和在模型解码过程当中还有很多研究可以值得我们继续做。

7.3.1 使用更优的微调训练方法

我们通过迭代的方法将伪数据的质量提高，但最终发现这种提高对于最终模型性能的提升并不是很大，虽然有提升，但是提升幅度比较小。我们相信，更高质量的伪数据对于最终模型性能的提升应该有更大的帮助。为此，我们后续希望在微调训练方法上做一些研究。目前，我们所做的微调方法主要是使用衰减过后的学习率进一步将神经网络中所有的权重都根据高质量数据微调一遍。后续我们可以调整学习率，在伪数据质量较高时降低学习率，使其微调时影响不要过大；此外，我们还可以将一些高层次的网络层进行固定，训练层次较低的网络层。由于高层网络层具有更高的抽象能力，所以其迁移能力也是比较强，而底层的网络层没有很强大的迁移能力，故将底层的网络层重新微调使其更与高质量数据接近，有助于模型微调。

7.3.2 基于权重共享的融合方法

在本次课题中，我们只考虑了将两个模型的词向量或者输出层的权重进行了共享。而我们可以将模型网络层中间的一些层进行共享的方法，可以使得两个模型之间具有更相似的分布。通常基于神经网络的模型随着层数的增加，网络层所处理的信息越来越抽象，越来越无法易于人类进行理解。为此，我们可以考虑将层数较高的网络层进行共享，这样我们可以将底层的一些层具有不同的信息的输入，保证两个任务的差异性，而随着

层数越来越高，两个任务的差异性越来越小，最终可以使得当信息流经过一定层数之后上层的网络层同时处理两个信息流。这样，我们可以更充分的利用单语语料去训练一个与神经机器翻译模型分布相类似的模型，可以充分利用单语语料的信息，有助于模型性能的进一步增长。

参考文献

[1] Dugast L, Senellart J, Koehn P. Statistical post-editing on SYSTRAN's rule-based translation system[C]. Proceedings of the Second Workshop on Statistical Machine Translation. Association for Computational Linguistics, 2007: 220-223.

[2] Makoto Nagao, Machine Translation: How Far Can it Go?[D]. Oxford:Oxford University Press, 1989.

[3] Mayor A, Alegria I, De Ilarraza A D, et al. Matxin, an open-source rule-based machine translation system for Basque[J]. Machine translation, 2011, 25(1): 53.

[4] Brown R D, Hutchinson R, Bennett P N, et al. Reducing boundary friction using translation-fragment overlap[C]. Proceedings of MT Summit IX, 2003:24-31.

[5] Doi T, Yamamoto H, Sumita E. Graph-based retrieval for example-based machine translation using edit-distance[C]. Phuket, Thailand: MT Summit X second workshop on example-based machine translation, 2005:51-58.

[6] Kim J D, Brown R D, Jansen P J, et al. Symmetric Probabilistic Alignment for Example-Based Translation[C]. Proceedings of the tenth workshop of the European Assocation for Machine Translation (EAMT-05), 2005:153-159.

[7] Nagao M. A framework of a mechanical translation between Japanese and English by analogy principle[J]. Artificial and human intelligence, 1984:351-354.

[8] Grefenstette G. The World Wide Web as a resource for example-based machine translation tasks[C]. Proceedings of the ASLIB Conference on Translating and the Computer. 1999, 21.

[9] Somers H. Example-based machine translation[J]. Machine Translation, 1999, 14(2): 113-157.

[10] Sumita E, Iida H. Experiments and prospects of example-based machine translation[C]. Proceedings of the 29th annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1991: 185-192.

[11] Philipp Koehn, Franz Och and Daniel Marcu. Statistical phrase-based translation[C]. In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT:NAACL),

2003, 48-54.

[12]Koehn P, Hoang H, Birch A, et al. Moses: Open source toolkit for statistical machine translation[C]. Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, 2007: 177-180.

[13]刘群. 基于句法的统计机器翻译模型与方法[J]. 中文信息学报, 2011, 25(6):63-71.

[14]肖桐. 树到树统计机器翻译优化学习及解码方法研究[D]. 沈阳:东北大学, 2012.

[15]Chiang D. A hierarchical phrase-based model for statistical machine translation[C]. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005: 263-270.

[16]Brown P F, Pietra V J D, Pietra S A D, et al. The mathematics of statistical machine translation: Parameter estimation[J]. Computational linguistics, 1993, 19(2): 263-311.

[17]Och F J, Ney H. Discriminative training and maximum entropy models for statistical machine translation[C]. Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002: 295-302.

[18]Xiao T, Zhu J, Zhang H, et al. NiuTrans: an open source toolkit for phrase-based and syntax-based machine translation[C]. Proceedings of the ACL 2012 System Demonstrations. Association for Computational Linguistics, 2012: 19-24.

[19]Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[20]Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]. Advances in neural information processing systems. 2014: 3104-3112.

[21]Kalchbrenner N, Blunsom P. Recurrent continuous translation models[C]. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013: 1700-1709.

[22]Wu Y, Schuster M, Chen Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. arXiv preprint arXiv:1609.08144, 2016.

[23]Junczys-Dowmunt M, Dwojak T, Hoang H. Is neural machine translation ready for deployment? a case study on 30 translation directions[J]. arXiv preprint arXiv:1610.01108, 2016.

[24]Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. Advances in Neural

Information Processing Systems. 2017: 6000-6010.

[25] Paulus R, Xiong C, Socher R. A deep reinforced model for abstractive summarization[J]. arXiv preprint arXiv:1705.04304, 2017.

[26] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.

[27] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[28] Zhou J, Cao Y, Wang X, et al. Deep recurrent models with fast-forward connections for neural machine translation[J]. arXiv preprint arXiv:1606.04199, 2016.

[29] Mike Schuster, Kaisuke Nakajima, et al. Japanese and Korean voice search[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2012: 5149-5152.

[30] Bradbury J, Merity S, Xiong C, et al. Quasi-recurrent neural networks[J]. arXiv preprint arXiv:1611.01576, 2016.

[31] Gehring J, Auli M, Grangier D, et al. Convolutional sequence to sequence learning[J]. arXiv preprint arXiv:1705.03122, 2017.

[32] Kalchbrenner N, Espeholt L, Simonyan K, et al. Neural Machine Translation in Linear Time[J]. arXiv preprint arXiv:1610.10099, 2017.

[33] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series[J]. The handbook of brain theory and neural networks, 1995, 3361(10): 1995.

[34] Sennrich R, Haddow B, Birch A. Neural machine translation of rare words with subword units[J]. arXiv preprint arXiv:1508.07909, 2015.

[35] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, Christian Jauvin. A Neural Probabilistic Language Model[C]. Journal of Machine Learning Research 3, 2003, 1137-1155.

[36] Kim Y, Denton C, Hoang L, et al. Structured attention networks[J]. arXiv preprint arXiv:1702.00887, 2017.

[37] Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation[J]. arXiv preprint arXiv:1508.04025, 2015.

[38] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997,

9(8): 1735-1780.

[39]Rico Sennrich, Barry Haddow, Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data[J]. arXiv preprint arXiv:1511.06709, 2016.

[40]Barret Zoph, Deniz Yuret, Jonathan May, Kevin Knight. Transfer Learning for Low-Resource Neural Machine Translation[C]. Proceeding of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, 1568-1575.

[41]Marek Rei. Semi-supervised Multitask Learning for Sequence Labeling[J]. arXiv preprint arXiv:1794.07156, 2017.

[42]He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[43]Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[44].Ashish Vaswani, Samy Bengio, Eugene Brevdo, et al. Tensor2Tensor for Neural Machine Translation[C]. Proceedings of AMTA 2018, 193-199.

[45]Papineni K, Roukos S, Ward T & Zhu, W J. BLEU: a method for automatic evaluation of machine translation[C]. In Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational, 2002, 311-318.

[46]那顺乌日图, 蒙古文词根、词干、词尾的自动切分系统[D]. 呼和浩特:内蒙古大学, 1997.

[47]李金廷, 侯宏旭, 武静, 王洪彬, 樊文婷. 语料预处理对蒙古文-汉文统计机器翻译的影响[J]. 计算机科学, 2017, 44(10):.047-052

致 谢

很高兴能够完成此次毕业设计，在毕业之际，向实验室的所有帮助我、给我照顾的老师、同学们表示衷心的感谢。

首先，非常感谢我的导师肖桐老师对我的耐心指导和帮助，从论文选题、论证、研究到最后的实现和完成，老师对我提供了很多方面的无微不至的帮助。在整个毕业设计过程中，积极关注我，时刻掌握着我的实现进度，并时时督促我，使得整个研究和实现的过程十分顺利。肖桐老师的认真负责的态度，勤奋进取的精神，以及广播的学识对我的研究和生活都产生了很大的影响。肖桐老师可以说是我人生路上十分重要的一个人，老师缜密的思维逻辑，认真的治学态度值得每个人学习，并且他时常告诉我做研究不只是将想法用程序实现，更重要的是要对问题有分析能力，对结果有预期能力，这样才能让自己的工作生活更加有条理，遇事不急不躁，做事不慌不忙。肖桐老师了解哪些课题适合我去研究，会站在我的角度选择一些适合的问题供我参考，让我能够在自己能力范围内通过自己的努力取得一个好的结果。感谢您这两年多的悉心培养和照顾，真的非常感谢您！

然后，我要感谢实验室的负责人朱靖波老师，您教会我的不仅仅是专业相关的知识，在做人做事方面也让我学到了很多。很多时候您对问题的看法和见解都很大程度影响了我对人生的规划，让我能够以不同的视角看待许多事情，再次感谢您。还要感谢自然语言处理实验室，感谢实验室提供的平台。感谢各位学长、学姐、学弟、学妹们在我遇到问题的时候给予的帮助。因为有你们，我才可以在欢快的氛围中生活和学习，谢谢大家。

我还要感谢寝室的三位舍友，最后几个月我总是打扰他们休息，但他们从未有怨言。在学校期间，我们总是互帮互助，在第一学期上课时期我们通常一起学习，一起准备考试。我们分享我们所学的东西，每个人都由衷的希望我们都能够顺利毕业、顺利找到工作，并有美好的未来。

我还应当感谢我的父母，在最困难和迷茫的时候，他们充分尊重、全力支持我的选择。天底下最难得的就是父母的理解和包容，我很幸运生在一个这样开明的家庭中，非常感谢他们对我的爱与付出，这也是我不断前进的动力。

硕士期间参加的科研项目

机器翻译评测：

[1] 全球机器翻译评测 WMT 参加中英、英中翻译项目，获得中英自动评价第四名，人工评价第二名，2018；

[2] 中国机器翻译评测 CWMT 参加中英、英中、蒙汉翻译项目，获得英中第一名、中英第二名、蒙汉第二名，2018；

[3] AI Challenger 全球挑战赛英汉口语翻译竞赛双周赛亚军，2017。

硕士期间取得的学术成果

学术成果：

[1] 以第一作者写技术报告《Machine translation system for CWMT2018 Mongolian to Chinese translation task》

[2] 以第二作者发表技术报告《TencentFmRD Neural Machine Translation for WMT18》