分类号 密级

UDC

学 位 论 文

面向自注意力机制的神经机器翻译系统

解码加速方法研究

|  |  |  |  |
| --- | --- | --- | --- |
| 作者姓名： | 李垠桥 | | |
| 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： | 工学 |
| 学科专业名称： | 计算机软件与理论 | | |
| 论文提交日期： | 2018年6月 | 论文答辩日期： | 2018年6月 |
| 学位授予日期： |  | 答辩委员会席： | 赵宏 |
| 评阅人： | 朱靖波教授 战学刚副教授 | | |

东 北 大 学

2018年6月

##### A Thesis in Computer Software and Theory

**Research of Decoding Acceleration Method for**

**Neural Machine Translation Based on Self-attention Mechanism**

By Li Yinqiao

Supervisor: Associate Professor Xiao Tong

**Northeastern University**

**June 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，不包含其他人己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己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

随着相关技术的发展，机器翻译系统的性能得到了长足的进步，由基于统计的方法发展到了目前备受瞩目的神经机器翻译。神经机器翻译系统主要基于编码解码框架对翻译问题进行建模，在编码器和解码器中分别对源语和目标语的语义信息进行抽象，最终通过二者之间的注意力机制进行语言片段间的对应，实现翻译的目的。

虽然神经机器翻译系统具有较强的翻译能力，但由于神经网络在运算的过程中包含着大量的矩阵运算，使得在翻译的过程中解码器的时间消耗也相对较大。为降低翻译系统在实际应用及部署中的时间代价，本文将针对目前性能最优的基于自注意力的神经机器翻译系统解码过程进行结构上的优化，使其在不影响或较小影响性能的前提下有效提升解码的速度。

本文从降低翻译模型中注意力操作的时间消耗角度出发，主要提出了两种加速解码的方法，二者分别从提高注意力操作的执行效率以及减少注意力操作执行次数的角度对问题进行求解。文中提出的基于coarse-to-fine的注意力机制有效减少了计算注意力权重时的词向量冗余，取消了该过程中不必要的计算。另一种方法为基于共享权重的注意力机制，这种方式基于相邻层之间注意力权重的相似性，将下层中得到的权重直接复用到与其临近的上层中来，使得权重计算的次数有所减少，达到加速解码的目的。

以上两种方法分别在180万的中英方向LDC数据上进行实验，使用NIST标准测试集对其性能进行检验。基于coarse-to-fine的方法在不损失性能的前提下对注意力部分的速度提升约10%左右；在基于共享权重的注意力方法中，以BLEU值下降0.3左右为代价，注意力部分的速度可提升约40%。

**关键词：**机器翻译；神经网络；解码加速；自注意力机制

Abstract

With the development of related technology, the performance of machine translation has made great progress. It went from statistical machine translation to neural machine translation which has caught much attention from researchers now. The neural machine translation system mainly models the translation problem by encoder-decoder architecture and abstracts the semantic information of the source language and the target language in the encoder and decoder respectively. Finally, the system corresponds to fragments between the source language and the target language by attention mechanism and get the translation from the system.

Although the neural machine translation system has a strong translation ability, the time consumption of the decoder in the process of translation is relatively large because of the large number of matrix operations in the process of operation. In order to reduce the time cost of the practical application and deployment of the translation system, this paper will optimize the decoding process of the neural machine translation system based on self-attention, which has the best performance at present. So that the decoding speed can be improved without affecting or less affecting the performance.

From the point of reducing time consumption of attention mechanism in translation model, this thesis mainly proposes two methods to speed up decoding process. These two methods solve the problem from the angle of improving the efficiency of attention operation and reducing the number of attention operation separately. The attention mechanism based on coarse-to-fine proposed in this paper can effectively reduce the redundancy in word embedding during calculate attention weights and eliminate unnecessary computation in the process. Another method is the attention mechanism based on sharing weights, which is based on the similarity of attention weights between adjacent layers. Directly multiplexes the weights from the lower layer to the upper layer which is adjacent to, so that the times of weight calculation are reduced to achieve the purpose of accelerating decoding.

The above two methods are tested on 1.8 million Chinese-English directional LDC data, and their performance is tested on NIST standard test set. In the method based on coarse-to-fine, the speed of the attention part can be increased by about 10% without loss of performance. In the attention method based on sharing weight, the speed of the attention part can be increased by about 40% at the cost of decreasing the BLEU value by about 0.3.

**Key words:** machine translation; neural network; decoding acceleration; self-attention

目 录

[独创性声明 I](#_Toc517365295)

[摘 要 II](#_Toc517365296)

[Abstract III](#_Toc517365297)

[第1章 绪 论 1](#_Toc517365298)

[1.1 研究背景 1](#_Toc517365299)

[1.2 研究内容 2](#_Toc517365300)

[1.3 论文组织结构 3](#_Toc517365301)

[第2章 相关技术概述 5](#_Toc517365302)

[2.1 神经机器翻译框架介绍 5](#_Toc517365303)

[2.1.1编码解码框架 6](#_Toc517365304)

[2.1.2 注意力机制 8](#_Toc517365305)

[2.2 神经机器翻译模型结构介绍 11](#_Toc517365306)

[2.2.1基于循环神经网络的机器翻译 11](#_Toc517365307)

[2.2.2 基于卷积神经网络的神经机器翻译 14](#_Toc517365308)

[2.2.3 基于自注意力机制的神经机器翻译 15](#_Toc517365309)

[2.3 本章小结 18](#_Toc517365310)

[第3章 基于coarse-to-fine的注意力机制 21](#_Toc517365311)

[3.1 研究动机 21](#_Toc517365312)

[3.2 方法描述 24](#_Toc517365313)

[3.2.1基于coarse-to-fine的注意力机制 24](#_Toc517365314)

[3.2.2考虑信息增长的coarse-to-fine的注意力机制 28](#_Toc517365315)

[3.3 实验 29](#_Toc517365316)

[3.3.1 实验设置 29](#_Toc517365317)

[3.3.2 实验结果 31](#_Toc517365318)

[3.3.3 结果分析 34](#_Toc517365319)

[3.4 本章小节 38](#_Toc517365320)

[第4章 基于共享权重的注意力机制 41](#_Toc517365321)

[4.1 研究动机 41](#_Toc517365322)

[4.2 方法描述 41](#_Toc517365323)

[4.2.1 面向自注意力部分的权重共享机制 41](#_Toc517365324)

[4.2.2 面向编码解码注意力部分的权重共享机制 46](#_Toc517365325)

[4.3 实验 50](#_Toc517365326)

[4.3.1 实验设置 50](#_Toc517365327)

[4.3.2 实验结果与分析 52](#_Toc517365328)

[4.4 本章小结 54](#_Toc517365329)

[第5章 总结与展望 55](#_Toc517365330)

[5.1 工作总结 55](#_Toc517365331)

[5.2 创新点分析 56](#_Toc517365332)

[5.3 未来工作 56](#_Toc517365333)

[5.3.1 基于局部信息的注意力机制 57](#_Toc517365334)

[5.3.2 其他注意力权重共享的方式 57](#_Toc517365335)

[参考文献 59](#_Toc517365336)

[致谢 63](#_Toc517365337)

[硕士期间参加的科研项目 65](#_Toc517365338)

[硕士期间取得的学术成果 67](#_Toc517365339)

第1章 绪 论

1.1 研究背景

机器翻译（Machine Translation或MT）是采用电子计算机进行自然语言之间翻译的一门试验性学科。通俗来讲，就是利用计算机将一门自然语言（源语言）转换成另一种自然语言（目标语言）的过程。长期以来机器翻译一直被看作是解决翻译问题的终极技术手段之一，应用需求非常旺盛。比如，中国政府把包括机器翻译技术在内的自然语言理解研究列入到国家中长期科技发展规划纲要中；美国国防部所主导的“2049”计划和BOLT项目都将机器翻译作为现代信息技术中的制高点进行重点攻坚；据报道，谷歌翻译每天为全球两亿多个用户提供服务，每天翻译次数达十亿次，每天翻译处理的文字数量，相当于一百万册图书，超过了全世界的专业翻译人员一年能够翻译的文字规模。这些都体现了机器翻译的巨大价值和技术应用前景。

纵观机器翻译的发展历程，我们可以将机器翻译的方法分为两种。一种是基于规则的机器翻译[1-3]（Rule-Based Machine Translation或RBMT），另一种是基于语料库的机器翻译[4-7]（Corpus-Based Machine Translation或CBMT）。具体来说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）和目前受到广泛关注的神经机器翻译[19-22]（Neural Machine Translation或NMT）。早期，人们主要使用规则来进行机器翻译。但是随着研究的深入，基于规则的方法也逐渐暴露出了问题，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、语种扩充困难等问题。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。当时，IBM和AT&T等机构提出了统计机器翻译的思想。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。统计机器翻译系统的开发仅仅依赖于双语和单语数据以及人工定义的翻译特征，系统的鲁棒性和扩展性大大增强，在很多翻译任务中表现出明显的优势。但是统计机器翻译仍然需要依赖大量的特征工程，而且这种方法假设翻译过程具有一种隐含结构，这些都限制了模型的表示能力。进一步，研究者又提出了基于深度学习的神经机器翻译方法，简称神经机器翻译。这种方法直接将机器翻译问题用神经网络进行建模，模型学习在端到端方式下完成，整个过程不需要人工特征的设计。

相比传统的基于统计的机器翻译方法，神经机器翻译系统凭借着较高的译文质量吸引了众多研究人员使用该方法对机器翻译任务进行研究[23]，但由于神经网络本身的特点，其内部存在着大量的矩阵运算，因此其在使用的过程中会更加耗时。该问题在实用化的机器翻译系统中尤为突出，因为它们一般都会对于响应的时间有更严苛的要求，因此神经机器翻译系统的解码速度也成为翻译系统能否实用化的关键，是否能够在现有神经机器翻译系统的基础上对其速度进行优化成为了一个极其重要的课题。

1.2 研究内容

基于自注意力（Self-attention）机制[24]的神经机器翻译系统凭借着更近的信息传递距离在众多同类系统中获得了广泛关注，该方法将不同位置词汇之间的信息传输距离拉近为1，使得模型可以更加充分地表示序列中不同位置词汇之间的复杂关系。其核心思想在于通过对源语或目标语句中任意位置词汇之间的关联程度进行考量，获得词汇之间的相关性，进而将其作为不同词汇或片段信息整合过程中的权重，最终得到源语或目标语中的语义信息表述。

由于注意力机制需要在两个句子片段之间计算词汇的相关程度，其中涉及到大量的矩阵运算，因此会占用较多的解码时间。与此同时，在基于自注意力机制的神经机器翻译系统中，由于其计算过程中会频繁地在句子内部以及句子之间进行注意力对齐的操作，因此更加导致这种机器翻译的方法在实际使用中解码速度很难满足实时响应的需求。面对该问题，本文将从两个角度对基于自注意力机制的神经机器翻译系统解码速度进行优化，旨在以较小性能损失为代价，大幅度提升翻译系统的解码速度，达到性能和速度上的平衡。

1. 基于coarse-to-fine的自注意力机制：由于在传统方法中注意力机制的计算存在冗余，因此我们通过先将计算注意力权重时的输入向量映射到较低的维度后，再进行对齐操作，一定程度上降低了每次注意力对齐操作的计算量，减少了该操作的时间消耗，最终达到加速翻译系统解码速度的目的；
2. 基于共享权重的自注意力翻译系统：在基于自注意力机制的翻译系统中，每层都会进行注意力对齐的操作，但实际上由于不同层之间信息存在着语义上的关联，并非天差地别，因此我们可以通过在层和层之间对注意力权重进行共享，使得可以从计算次数的角度很大程度上减少注意力操作的耗时，同时并不会对系统性能产生较大程度的影响。

以上两种方法分别从降低注意力操作的耗时以及操作次数的角度出发，对基于自注意力机制的机器翻译系统进行加速，最终使得翻译系统在性能变化不大的前提下有效减少解码过程中的时间消耗。

1.3 论文组织结构

本文主要研究了面向自注意力机制的神经机器翻译系统解码加速的两种方法，它们针对注意力机制分别从提高运算速度以及减少运算次数的角度对问题进行求解，提出基于coarse-to-fine的自注意力机制和基于共享权重的自注意力翻译系统两种方法，最终达到加速机器翻译系统解码速度的目的。

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使用到的相关技术，主要围绕神经机器翻译中广泛使用的相关技术进行介绍。其中，详细介绍了近几年神经机器翻译中所使用的不同技术，包括序列到序列的模型[25]、注意力机制、基于循环神经网络（RNN）的神经机器翻译[26-28]、基于卷积神经网络（CNN）的神经机器翻译[29-32]、基于自注意力机制（Self-attention）的神经机器翻译[33-34]等。

第3章主要介绍了基于coarse-to-fine的注意力方法在基于自注意力机制的机器翻译系统中的使用，同时详细介绍其研究动机、实验情况和该方法存在的问题。

第4章主要围绕如何在现有的基于自注意力机制的神经机器翻译框架中采用权重共享的方式减少注意力操作的次数，从翻译模型结构的角度介绍了该方法的合理性，同时通过实验对其效果进行了验证。

第5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 神经机器翻译框架介绍

上世纪九十年代初IBM和AT&T等机构提出了统计机器翻译（statistical machine translation）的思想。与传统方法不同，统计机器翻译把翻译问题看作是搜索翻译概率最大的译文的问题，即

(2.1)

其中*s*和*t*分别表示源语言和目标语言句子，表示把*s*翻译成*t*的概率，表示从所有译文中搜索最佳译文的过程。以上这个过程也蕴含着建模（如何定义）、模型训练（计算所需的模型参数）、解码（过程）三个基本问题。在过去的二十年中，科研人员对上述问题进行了大量的探索。比如，在翻译建模方面，研究者从最开始的产生式模型逐渐进化到判别式模型，从开始的基于词的模型逐渐演变为基于短语的模型，甚至基于句法的模型。特别是在2005年，谷歌公司正式把机器翻译作为互联网在线服务推出，普通用户可以免费享受技术发展所带来的红利，这些都大大促进了机器翻译技术研发及产品化的步伐。由于统计机器翻译具有系统健壮性高、对人工依赖少等特点，时至今日它仍然是许多实用系统所使用的主要框架之一。

随着机器翻译研究的深入及机器学习等相关学科的进步，人们逐渐发现统计机器翻译存在着许多难以回避的问题，比如，翻译过程依赖隐藏结构的假设、翻译特征的定义需要人工设计、特征工程耗时耗力而且往往不具有普适意义。对于这些问题，人们又尝试了一种全新的思路——神经机器翻译。所谓神经机器翻译是使用神经元网络直接对翻译问题建模，这个过程并不假设翻译具有隐藏结构，同时也不依赖人工定义的特征，整个翻译模型都可以在一种端到端的模式下训练完成，而翻译解码也就变成对神经网络进行前向计算或推断的过程。

虽然神经网络已经在很多任务中得到应用，但是2013年之前业界在机器翻译方面并没有取得实质性的进展。主要原因在于：1）并没有一种非常有效的框架来处理翻译这种文字序列到文字序列的转化问题；2）深度神经网络的学习并不十分有效。深度网络学习问题在近些年不断取得进展，而神经机器翻译的框架选择问题在“编码解码”结构被提出之后得到缓解。

2.1.1编码解码框架

神经网络(Neural Network)是一种非常强大的机器学习模型，起初它在图像识别等领域上取得了优异的性能，但由于缺乏合理的框架，因此当时只能应用在输入输出维度固定的问题之中来。但就现实情况而言，这是一个很大的限制，因为对于许多重要的问题而言，我们很难预先对其维度进行一个提前的预判。例如，语音识别和机器翻译都是序列的转化问题的典型代表，它们需要在两个序列之间建立起一个合理的映射。因此，如何使用神经网络进行序列的转换是一个十分值得研究的问题。

面向序列转换问题，Kyunghyun Cho等人最早提出编码解码的框架。这种方法将序列到序列转换的问题建模为两个阶段，在编码阶段对输入序列进行编码，提取其中所包含的序列信息，之后在解码阶段对编码后的向量进行解析，最终得到输出的序列。具体来说在这种方法中，编码阶段利用循环单元（RNN、GRU、LSTM等）结构，在每个时间步向编码器中输入一个序列片段所对应的实数向量，从输入序列中的第一个位置开始依次输入，直到到达输入句子中的最后一个片段结束为止，这时在最后一个时间步中我们将得到整个输入序列的语义向量（上下文向量）。在每个时间步中隐层状态的更新方式如公式(2.2)所示。

(2.2)

由于循环单元本身的特点，它会记录每个时间步所输入的序列片段信息，因此我们在最后一个时刻得到的语义向量可以看作是包含了整个句子中的全部信息，我们可以将其作为输入序列的一个语义表示，也可以理解为所谓的句向量。

在解码阶段，模型根据编码所得到的上下文向量进行解码，每个时间步进行一次序列片段的生成，最终得到整个预测的序列结果。该过程中隐层状态的计算公式如公式(2.3)所示。

(2.3)

进一步来说，我们可以根据该隐层状态计算出的概率如式(2.4)所示。

(2.4)

公式中的函数和函数分别代表循环单元的操作（如GRU）和归一化的操作（如*softmax*），在解码的过程中，我们利用从输入序列中提取出的上下文向量，逐渐地从其中抽取信息作为输出序列的依据。具体来说，首先向解码器中输入一个起始标志，之后解码器就可以根据、和*c*预测出下一时刻的序列片段的概率分布，最后我们再选择概率最高的序列片段作为预测结果，同理，我们可以根据、和*c*预测出的概率分布，这样不断迭代直至预测到句尾标志，指示结束了目标序列的预测过程。

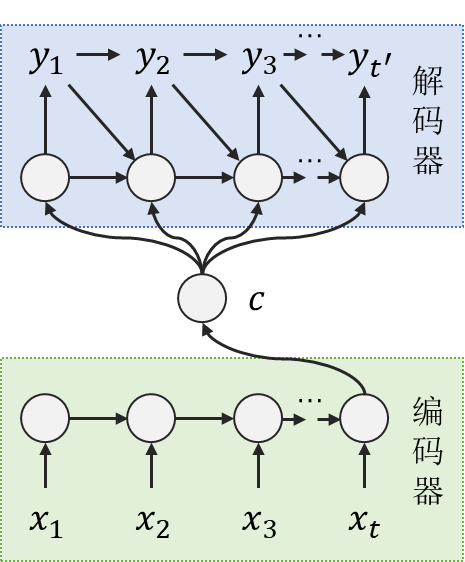


图2.1 编码器解码器框架

Fig. 2.1 Encoder-Decoder framework

在Kyunghyun Cho等人的论文中，上下文向量在解码的每个时间步中都会参与计算，但是该结构在Ilya Sutskever等人的论文中得到了改变，编码端得到的上下文向量仅仅会应用在解码端的第一个时间步中，而非每个时刻向量都会参与序列的生成。Kyunghyun Cho等人的论文中解码端所预测出的第*t*时刻的输出可用公式(2.5)表示。

(2.5)

而Ilya Sutskever等人的论文中所描述的输出则如式(2.6)所示。

(2.6)

在简化的编码解码结构中，我们可以看到编码端的上下文向量仅仅作用于解码端的第一个时刻，这样做使得整个编解码框架在使用的过程中计算量更小，同时模型结构更简单。整个过程的结构如下图所示。

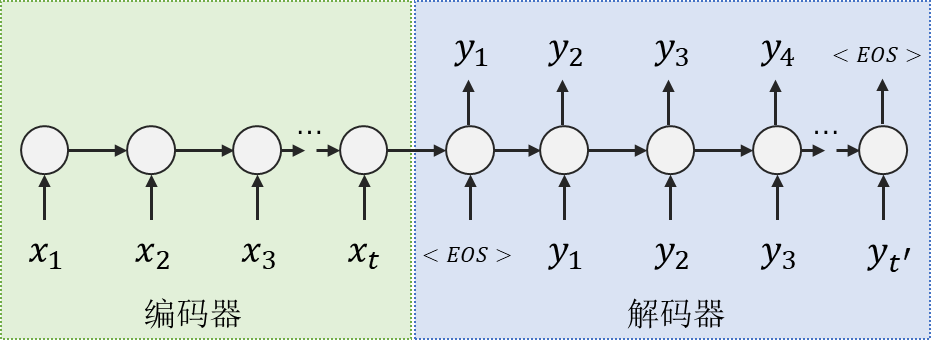


图2.2 简化的编码器解码器框架

Fig. 2.2 Simplified encoder-decoder framework

整体来说，所谓编码解码结构就是把序列到序列的转化问题定义为两阶段的建模问题。首先，将含有个单词的输入词序列进行编码，编码得到的结果是一个实数向量，它表示了整个输入序列在到第个单词为止的信息；第二个阶段利用编码得到的向量进行解码，依次生成输出序列，可以被描述为公式(2.7)的形式。

(2.7)

其中描述了目标语第个单词生成的概率。由于编码器已经将表示为，因此的条件部分仅仅与和相关.由于可以用神经元网络进行计算，这样也就使用了一个网络完成输入序列到输出序列的转化，而且网络可以通过相对成熟的反向传播方法进行训练。更为重要的是，这种模型在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。

由于编码解码框架完全把机器翻译问题转换为输入序列到输出序列的网络计算问题，不依赖人工设计的特征，因此可以更好的捕捉不同语言之间的复杂对应关系。可以说基于编码解码框架的神经机器翻译已经成为了相关研究机构和企业的标配。在这个框架之上，研究人员也进行了大量的改进、升级。相比三到五年前的水平，当今的机器翻译品质已经得到飞跃式的发展。

2.1.2 注意力机制

在神经机器翻译中引入注意机制的方法使得模型可以根据当前待翻译的词汇从源语端选择语义最相关的部分进行信息提取，利用提取到的信息指导当前目标语词汇的预测过程，这种方法可以更有针对性地对当前词汇进行翻译，因此相对之前未使用注意力机制[37]的神经机器翻译系统有着更优的性能。

在传统的未使用注意力机制的神经机器翻译系统中，根据编码解码结构，在编码器部分会将输入的源语句子编码为一个固定长度的上下文向量，然后在解码端再根据这个向量进行译文的生成。但是由于在翻译的时候句子长度可能相对较长，因此其隐层状态很难有效兼顾从句首到句尾中间的每一个词，最终导致翻译系统对于句长较长的句子翻译效果较差。

图2.3为Minh-Thang Luong等人在论文中提出的将注意力机制应用到神经机器翻译的模型结构中来。在模型训练及解码过程中，解码器根据之前时刻的译文状态，通过某些计算方法得到当前时刻待预测的译文和源语端序列片段的相关程度，然后以该值作为源语序列中每个片段信息的权重进行加权求和，最终得到输入序列的信息向量。在不使用注意力机制的传统模型中，从源语端得到的输入向量信息包括源语句中的全部信息，而在引入了注意力机制之后，该向量会着重包含和当前所预测词汇最相关的源语信息，这样可以使得预测的内容更有针对性，类似于在统计机器翻译中的词对齐模块的功能。

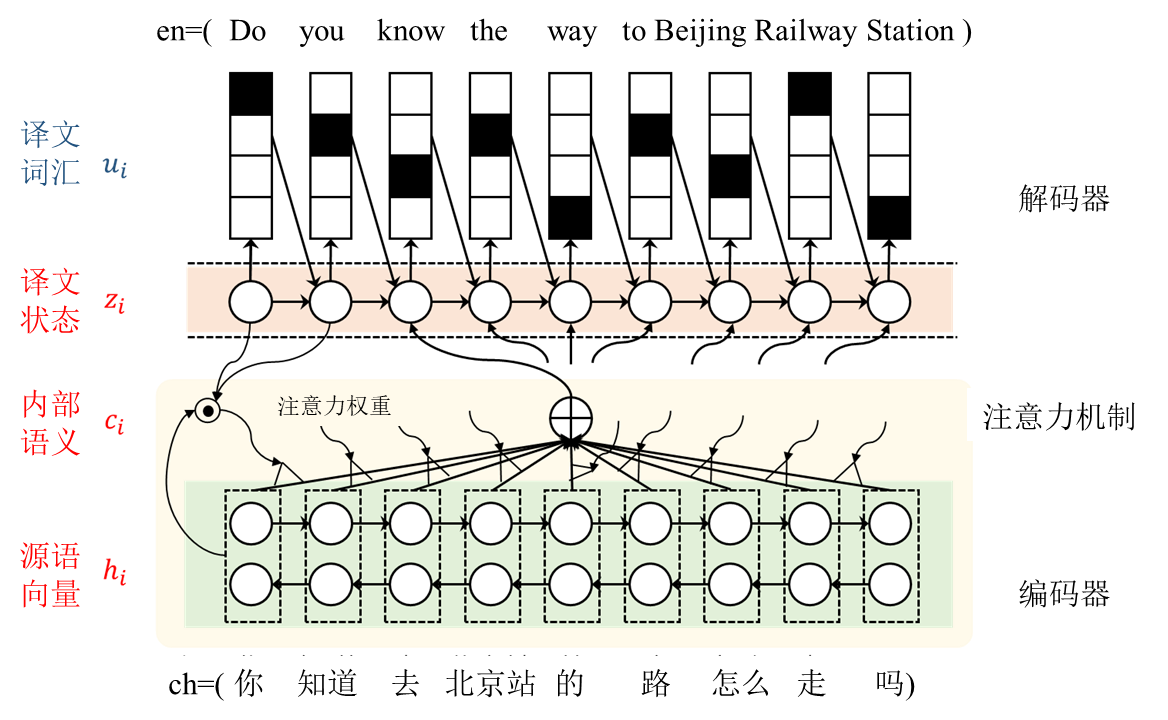


图2.3 引入注意力机制的神经机器翻译模型

Fig. 2.3 Neural machine translation model with attention mechanism

对于注意力机制的具体实现有很多种方法，整体可以分为两大类，一类是面向所有源语词汇片段的全局注意力机制（global attention），另一类是面向部分片段的局部注意力机制（local attention），两种方法主要区别在于对源语端的哪些片段进行注意力权重的计算。

全局注意力机制将通过当前待预测的词汇信息对源语端的全部词汇进行注意力对齐操作，其权重通过公式(2.8)进行计算得到。

(2.8)

该式表示源语端第个词汇和目标语端第个词汇之间的关联关系，即注意力的权重，为解码端在预测当前词时的隐层状态，为源语端第个词的状态，我们通过这两个向量预测当前待预测词汇和源语端的第个词汇之间的关联程度，其具体计算公式如公式(2.9)所示。

(2.9)

为使得对齐权重以概率的形式进行表达，因此我们需要对其进行归一化的处理，因此上式中使用*softmax*函数对解码端在预测当前词时的隐层状态和源语端词汇的状态之间的得分进行处理，最终得到它们之间的注意力权重。

对于其打分函数的具体计算方法，在Minh-Thang Luong等人的论文中给出了三种方式，分别为点乘的方式，通用的方式和连接的方式。

基于点乘的公式如公式(2.10)所示。

(2.10)

通用的公式如公式(2.11)所示。

(2.11)

基于连接的方式如公式(2.12)所示。

(2.12)

三种方式均有效利用了解码端在预测当前词时的隐层状态和源语端词汇的状态信息，不同之处在于点乘的方式直接对这两个向量进行点乘的操作作为其相关程度，通用的方法在点乘的基础上增加一个线性变化，提升注意力模型的通用程度（当该变换矩阵为单位矩阵时通用的方法退化成点乘的方法），连接的方式是将两个状态信息连接到一起后再通过一个线性变换得到。

除了全局注意力机制之外，Minh-Thang Luong等人在论文还提出一种局部的注意力机制，这种方法在计算当前待预测的词汇信息和源语端词汇相关程度的时候仅关注源语端的部分词汇，而非和全部序列片段进行计算，这样做一定程度上会减少注意力机制的计算量，提升系统训练和解码的速度。

具体来说，在计算注意力权重的时候，该模型并不是去对源语端的全部词汇进行考量，而是通过一个函数预测源语言序列中哪一部分更可能和当前待预测的词汇更相关，在这个窗口内进行注意力权重的计算，这样做一定程度上相当于缩小了源语端在计算注意力权重时的序列长度。

对于如何确定源语序列中的哪一部分将作为当前解码端预测的源语窗口，我们需要通过一个预测函数来预测解码当前单词时要对齐的源语言端的位置，然后在该位置左右开设一个窗口作为计算注意力权重的区间，对于的预测主要有两种方式：

第一种方式相对简单一些，它的核心思想在于当在对目标语端的第个词汇进行预测的时候，源语端同样位置（第个词汇）很可能是和它最相关的，因此我们直接将设置为，之后对窗口内的词汇片段按照公式(2.13)来计算注意力权重即可。

(2.13)

第二种方式主要通过解码端当前待预测词汇的隐层状态对源语端的窗口位置进行预测，公式如公式(2.14)所示。

(2.14)

式中和为注意力机制模型中的参数，为输入序列的长度，上式中通过对解码端的隐层状态进行一系列线性/非线性变换后预测出当前待预测的词汇在源语中最相关的区域，之后再通过式(2.13)计算出当前待预测的目标语词汇和窗口内的词汇之间的关联程度，最终的词对齐概率结果还要根据窗口内的源语词汇和的距离进行相应的缩放，论文中通过一个高斯分布对其进行建模，公式如公式(2.15)所示。

(2.15)

该方法实际上就是在预测出的窗口内所计算出的相关程度基础上增加了一个高斯分布，认为越接近位置的源语词汇应该有更大的对齐概率。

2.2 神经机器翻译模型结构介绍

在上一小节中介绍了神经机器翻译的基本框架——基于序列到序列的框架，但框架仅仅提供了如何对机器翻译任务进行建模的整体方式，对于其内部的网络结构并没有强制的要求。随着国内外相关研究人员的深入探索，神经机器翻译模型结构的设计也逐渐发生着变化，由最开始主流的基于循环神经网络的翻译结构，逐渐演变出基于卷积网络甚至是自注意力机制的神经机器翻译系统，这些系统针对前人模型结构中所存在的问题进行革新，将神经机器翻译系统的性能逐渐提升到一个新的层次，本小节将对这个发展过程中几种主流的模型结构进行介绍。

2.2.1基于循环神经网络的机器翻译

机器翻译作为自然语言处理的一个重要任务，和情感分析、语言理解等任务一样，需要模型能够有效对变长序列进行处理。在神经网络中，对于变长序列的问题最经典的解决方案之一就是通过循环神经网络对问题进行建模。由于循环神经网络（Recurrent Neural Network, RNN）在神经网络结构中引入了时序的概念，因此对于变长序列的处理有着得天独厚的优势。

一个最简单的循环神经网络由输入层、隐藏层和输出层组成，其结构如图2.4所示，其核心思想在于每个时刻向网络中送入一个输入x，然后通过线性、非线性变换将结果送入隐藏层中，隐藏层对经变换后的上一时刻隐层状态以及输入层的输入进行结合得到当前时刻的隐层状态，一方面将其送往输出层，另一方面送往下一时刻的隐层中继续计算。整个过程如图2.4所示，其左侧部分抽象地表示了隐藏层中数据的传递模式，右侧将该表示按照时序进行展开，得到每一个时间点中模型的输入和输出。

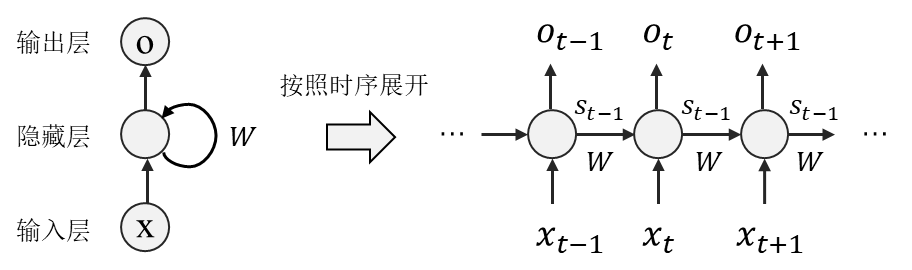


图2.4 循环神经网络结构

Fig. 2.4 Structure of recurrent neural network

具体来说，在*t*时刻，对于隐藏层而言，其输入层给出的输入向量为，上一时刻隐藏层的状态为，那么当前时刻隐层状态可以表示为公式(2.16)。

(2.16)

式中*W*和*U*分别为输入向量、隐层状态的变换矩阵，*b*为隐层中的偏置，为隐层的激活函数（如*tanh*、*sigmoid*等），通过上述公式的计算我们即可得到当前时刻隐层状态的输出了。由于传统的循环神经网络中存在梯度爆炸和梯度消失的问题，因此研究人员提出了长短时记忆网络[38]（LSTM）来缓解该问题。该结构属于循环神经网络的一种，但其内部引入了门控单元，结构相对普通的循环神经网络来说更加复杂。

在目前的神经机器翻译模型中，不论具体的模型结构采用何种方式实现，其整体框架仍然使用编码解码的结构进行建模。在该框架下，我们可以使用多种多样的模型结构对源语、目标语内容进行抽象。以图2.5中源语端句子为例，“你 知道 去 北京站 的 路 怎么 走 吗”，通过循环神经网络的处理该句子将由编码器抽取出一个代表其语义的向量表示，在解码的过程中该向量将被还原为译文端的每个词汇（“Do you know the way to Beijing Railway Station”）。

具体来说，翻译模型的输入为一个one-hot向量，代表输入序列中的每个单词，其维度大小为词表的大小。该向量中只有一位为1，其余均为0，值为1的那一位代表当前向量所代表的词即为词表中当前顺位的词汇。由于这种表示方法随着词表的增大会发生维度爆炸的问题，因此在网络中会将其映射到一个低维空间上对词汇进行表示，得到的向量被称为词向量。此外这种表示方法还使得我们可以通过词向量之间的距离来对词汇相关度进行度量，使得词向量之间产生了相互之间的关联，而非原本方法中完全孤立的点。接下来循环神经网络将针对按序列顺序输入的词向量依次编码，得到一个源语言序列的压缩向量，方式可归结为公式(2.17)所示。

(2.17)

其中，代表*i*时刻的隐层状态，其初始值为全零向量，随着输入序列的增加中的信息量逐渐丰富，最终得到整个句子的压缩表示。

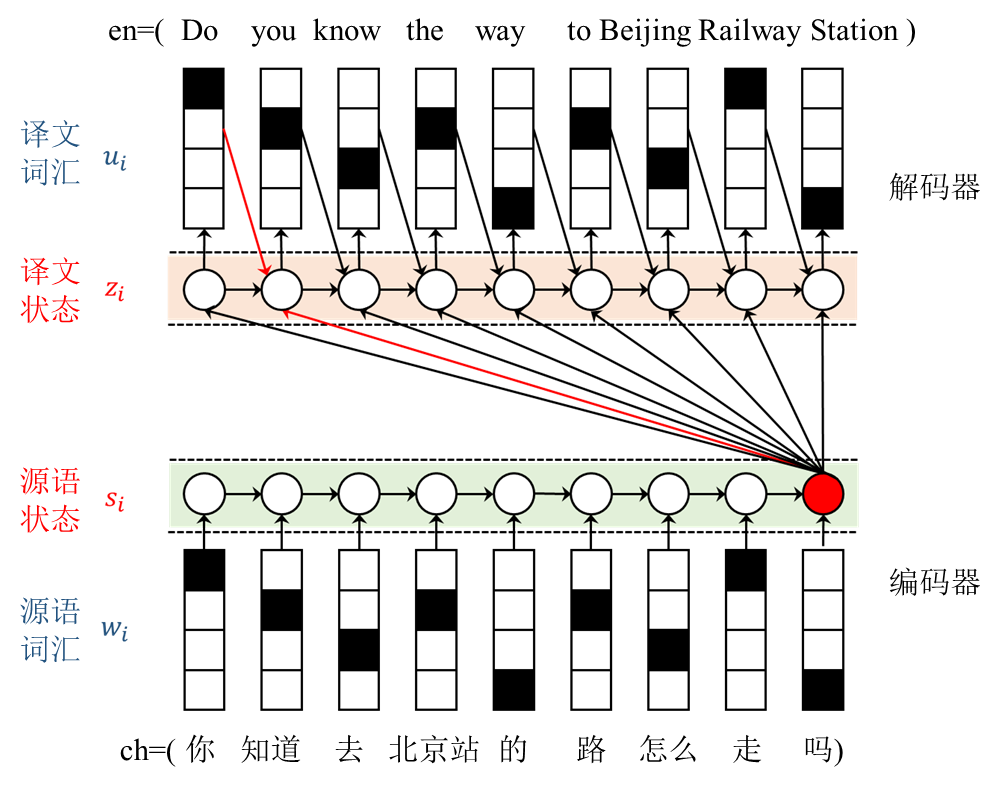


图2.5 基于循环神经网络的神经机器翻译

Fig. 2.5 RNN-based neural machine translation

在解码端，模型将会根据源语端收集到的信息（称为上下文向量*c*）对目标语端的词汇进行预测，依次获得翻译系统解码出的译文。整个过程中的隐层状态可以形式化地描述为公式(2.18)。

(2.18)

其中*c*代表源语端的上下文向量，其中包含了源语端句子的全部信息。表示*t-1*时刻（上一时刻）预测出的单词所包含的词汇信息（输出层预测的单词对应的词向量），为*t-1*时刻（上一时刻）隐藏层的输出状态，最终结果作为当前时刻隐层的状态值。在实际使用中，由于解码过程总是依赖前一时刻的隐层状态及输出信息，因此系统需要假定为一个全零向量，为<EOS>所对应的词向量（<EOS>为句子的起始标志位，用来表示编码过程的结束以及解码过程的开始）。

在输出层中，我们对隐层得到的信息进行相应的转化，得到当前时刻所预测的单词在词表中的概率如公式(2.19)所示。

(2.19)

通过上式我们可以得到当前待预测的单词所对应的概率分布，该式中为当前时刻隐层的输出，我们对其通过和进行线性变换，最后通过*softmax*函数进行归一化，得到每个词汇的预测概率。不断重复上述过程，直至解码出句尾标志<EOS>时停止预测，认为生成的序列已经结束，得到译文的句子。

2.2.2 基于卷积神经网络的神经机器翻译

由Yann LeCun提出的卷积神经网络作为一种擅长提取局部信息的网络结构在图像识别等领域取得了良好的成绩，由于文本处理相关任务往往被看作是整个序列到序列的整体变换，因此在最初并未得到相关研究人员的广泛关注（相对于循环神经网络）。但相关研究人员在使用循环神经网络处理文本相关任务的时候逐渐意识到循环神经网络在某些方面存在其设计上的不足，由于循环单元在处理句子的时候需要去理解句中每一个单词的含义，但是当面对句长较长的句子的时候，循环单元很难有效地对句中每个单词的信息进行提取，常常会出现在读取到最后一个词的时候已经忘了句子开头内容的问题。另外，由于循环神经网络本身结构上的特性，因此句中两个相距较远的单词之间的关联程度会通过很长的方式才能得到连接，这种方式和我们直观的理解稍有不同，一般来说句中任意两个相互关联的单词在神经网络中应该可应通过较短的路径就可达（比如句子开头的词汇可能和句尾的某个词语义上存在很大的依赖关系）。与此同时，由于循环神经网络本身对于输入序列顺序的要求，使得基于该模式的机器翻译系统很难进行大规模的并行运算，并没有很好地将GPU自身对于并行运算的优势有效地发挥出来。这一问题卷积网络由于其自身结构的设计，使得网络能够从局部中捕获句子片段的信息，这就为并行训练和解码提供了很大的便利条件。基于卷积神经网络的机器翻译系统可以并行地关注句子中的不同语义片段，使得能够更加有效地利用GPU的计算能力，提升系统的执行效率。

Facebook在2017年提出一种利用卷积神经网络对机器翻译任务进行建模的方法，一改之前卷积网络在自然语言处理中的劣势。该方法使用多层的卷积操作对序列中的信息进行提取和解析，最终在多项翻译任务中得到了优于循环神经网络的效果，与此同时优于其高度可并行化的结构，使得模型在保证性能的前提下，执行效率相较基于循环神经网络的GNMT系统快9倍左右。在WMT-2014英语-罗马尼亚语的测试任务上相较之前的最佳结果提升1.8个BLEU值[39]，在英语-法语的任务上提升1.5个BLEU值，在英语-德语的任务上也有0.5个点的提升。

基于卷积神经网络的机器翻译系统结构如图2.6所示，在编码端卷积网络分别对每窗口个大小的词汇片段进行卷积，得到该片段的语义信息，之后通过堆叠若干层的卷积操作，使得网络在编码端收集到全部序列的语义信息。在解码端模型利用上述提取到的源语信息，对其进行解析，生成目标语的预测结果。

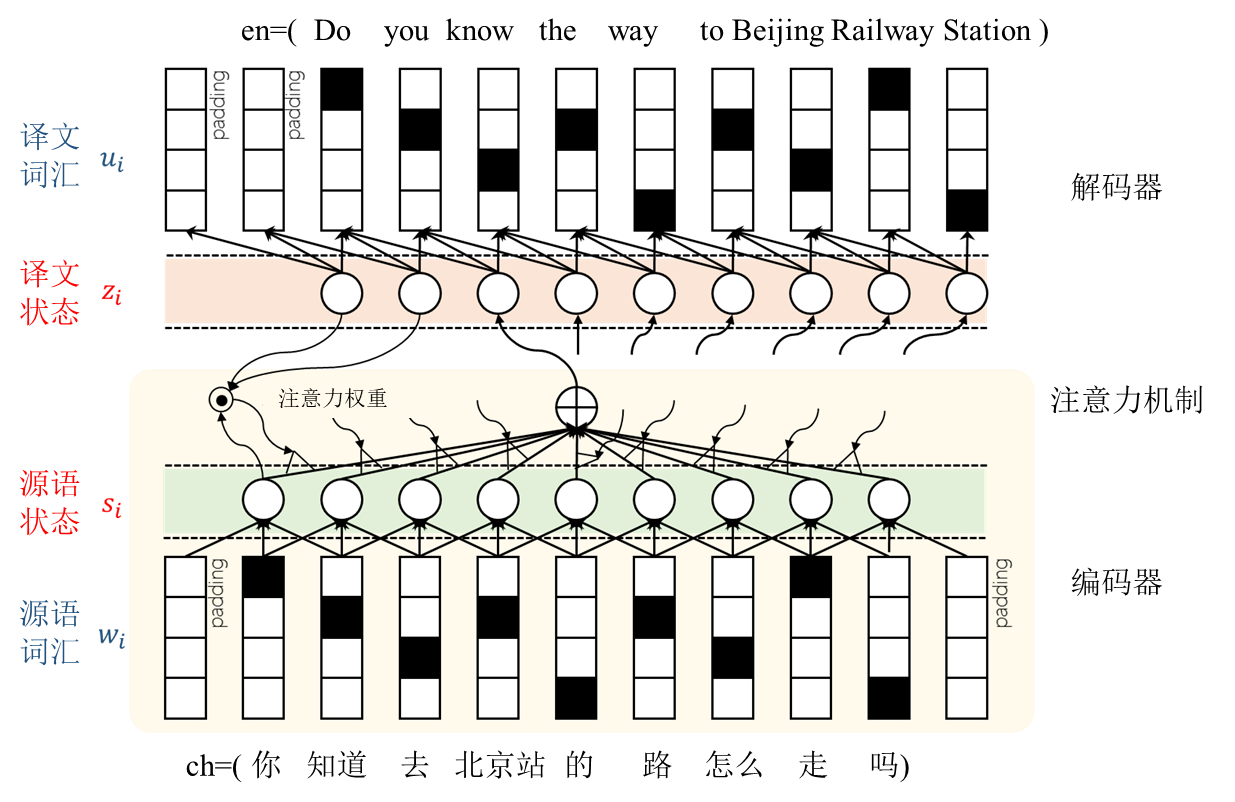


图2.6 基于卷积神经网络的神经机器翻译

Fig. 2.6 CNN-based neural machine translation

类似于卷积操作在机器翻译任务中的应用，卷积模型在其它序列到序列的任务中也可以进行广泛的应用（如摘要提取、写对联等）。由于卷积网络结构上的优势，使用该结构的模型在计算效率上更具优势，能够更加高效地利用GPU设备的并行能力，有效提升模型在训练以及解码中的速度，尤其是对于序列长度较长的句子相对传统的基于循环神经网络的模型更具优势。

2.2.3 基于自注意力机制的神经机器翻译

最近一些研究者将自注意力机制应用到神经机器翻译中，完全消除了模型对RNN和CNN等单元的依赖。该模型可以更有效地处理单词之间的关系，不同位置的信息传递距离仅为O(1)，这样可以非常充分地表示序列中不同位置之间的复杂关系，如图2.7所示。

对于自注意力模型，它的编码器由多个相同的栈堆叠而成。对于一个栈来说，它主要由两个子层构成：多头注意力（Multi-Head Attention）和前向连接层。

所谓多头注意力，它由两部分构成：头（head）和自注意力。它首先利用不同的权重矩阵把输入压缩成多个维度较小的输出，然后分别做自注意力，最后把所有结果都级联起来再做一个线性变换得到最终的输出，如图2.8所示。具体过程描述如公式(2.20)所示。

(2.20)

公式(2.20)中的计算方式如公式(2.21)所示。

(2.21)

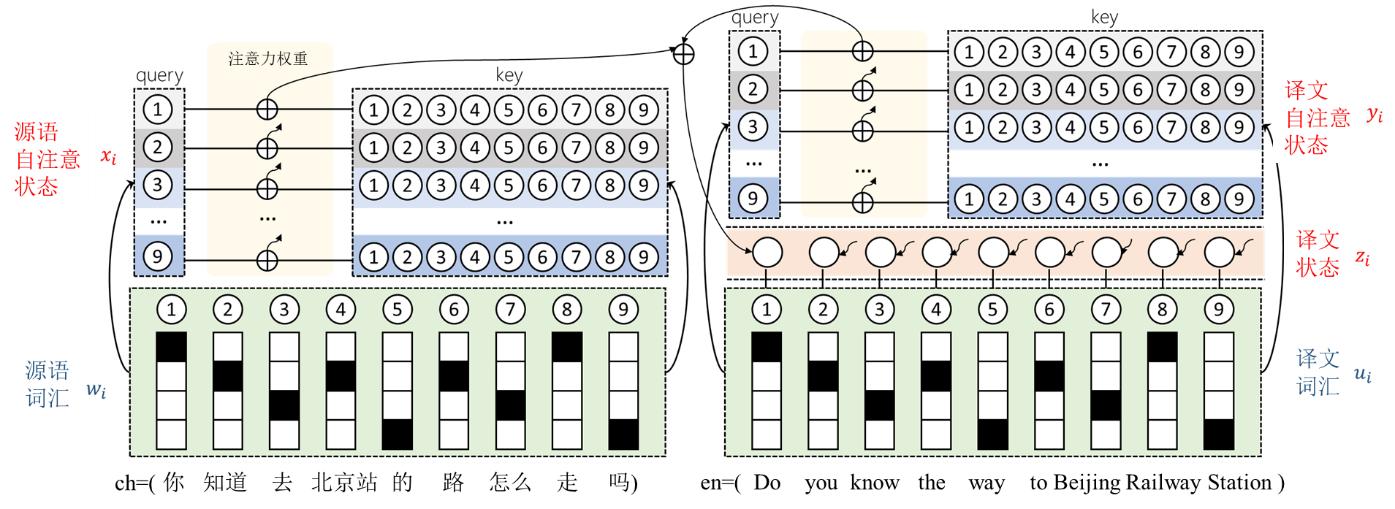


图2.7 基于自注意力机制的神经机器翻译

Fig. 2.7 Self-Attention-based neural machine translation

对于编码器的栈来说，它的多头自注意力输入的*Q*、*K*和*V*都是相同的，为源语言的词嵌入或下层栈的输出。实际上计算了源语言任意两个位置的相关性，是每个头的维度，做分母可以把相关性转化为合理的实数范围。是按源语言位置进行归一化，得到的结果是i与其它任意位置相关性权重，这个结果与*V*再相乘得到的是所有位置向量的加权和。这个过程没有使用任何循环单元或者卷积单元，而且可以并行化。

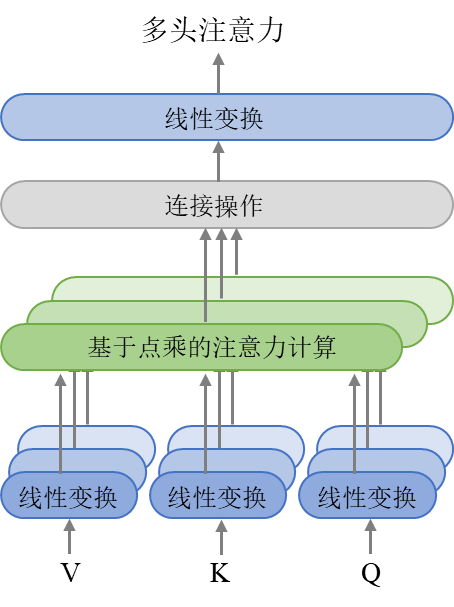


图2.8 多头注意力机制

Fig. 2.8 Multi-Head attention

在多头注意力的输出结果上，我们会加上残差连接[40]，也就是把输入加到输出上，具体计算如公式(2.22)所示。

(2.22)

残差连接通过直接在输入输出之间建立连接，避免了梯度消失的问题，使得复杂网络的优化变得更加容易。这是因为当我们在计算一层的输出到输入的导数的时候，残差连接总会给出梯度1，然后加上层内的导数构成当前层输出到输入的导数。在层内导数很小的情况下，残差连接保证了梯度不会小于1，使得网络底层同样能接收到相当量的梯度来进行更新。

在残差连接后的结果下，我们还会加上层正规化[41]，以加速模型收敛，如公式(2.23)所示。

(2.23)

其中为输出在隐藏层维度的均值，其计算过程如公式(2.24)所示。

(2.24)

其中为输出在隐藏层维度的标准差，其计算过程如公式(2.25)所示。

(2.25)

其中*g*、*b*是参数。层正规化把输出沿着隐藏层维度进行正规化，然后替换上自己学出来的均值和标准差。最后，这个结果送到一个双层，激活函数为ReLU的前馈神经网络里，结果加上残差连接以及层正规化，得到栈的最终结果。

而自注意力模型的解码器则是由多个相同但是与解码器不同的栈堆叠而成。它由三个子层构成：带掩码的多头自注意力，多头自注意力以及前向连接层。

所谓带掩码的多头自注意力，其实就是在多头自注意力的基础上，把的输入表示序列中未来的部分设成，也就是把这部分盖住。这样做是因为在测试的时候模型是自左向右来解码，在生成当前输出的时候模型并不知道将来的输入是什么，但是训练时模型是输入了完整的目标序列，因此做训练当前时刻预测的时候，需要把大于当前时刻的序列部分盖住，使得解码与训练一致。它的输入是目标语言的词嵌入或者是下层栈的输出。

在带掩码的多头自注意力后是正常的多头自注意力。与编码器的多头自注意力不一样，在它的输入中*Q*是带掩码的多头自注意力的输出，*K*、*V*则是编码器顶层栈的输出。它代表着模型把目标语言当前词以及其与前面词的关系作为请求，然后从编码器抽象得到的源语言特征序列或者说记忆当中，提取与下一个目标语言词相关的特征来做预测。

后面的前向连接层与编码器的栈一样，都是一个两层的前馈神经网络。模型最后在解码器最顶层栈的输出基础上，使用单层前馈神经网络和作下一词的概率分布预测。

这种方法中没有使用传统的网络结构（如循环神经网络、卷积神经网络），而是将注意力机制发挥到极致，通过这种方式对源语和目标语信息进行建模和提取。该模型中的注意力机制可以分为三个部分：

(1) 编码器端的自注意力：这部分用来对机器翻译系统中的源语内容进行信息抽取，其中*Q*、*K*、*V*均来自于编码器端上一层的输出；

(2) 解码器端的自注意力：这部分用来对机器翻译系统中已经预测出的译文片段进行信息抽取，其中*Q*、*K*、*V*均来自于解码器端上一层的输出。由于在解码端计算的过程中尚未翻译完全部句子，因此并不能看到全部的译文片段，这里使用遮罩对未翻译出的部分进行屏蔽；

(3) 编码器-解码器间的注意力：这部分用来确定机器翻译系统中待预测的词汇和源语端那些词汇片段最相关，从而可以更准确地对信息进行提取，其中*Q*来自于当前栈中的自注意力模块输出，*K*和*V*来自于编码器的输出。

这一方法的使用使得机器翻译的性能相较之前的基于循环神经网络的结构得到了较大程度的提升，获得了目前而言的最好成绩。和基于卷积网络的神经机器翻译类似，这种方法的使用也并不限于在机器翻译任务中去使用，理论上来说在基于序列到序列的模型上都可以使用该方法来降低序列中相关部分信息传递的距离，更进一步来说自注意力模型也可以拓展到音频、图像等非自然语言处理任务中来。

2.3 本章小结

本章主要介绍围绕机器翻译相关技术进行介绍，内容主要包括两部分：神经机器翻译的基本框架和模型中网络结构的概述。

在神经机器翻译框架的部分，本文主要介绍了现行较为流行的翻译框架——基于注意力机制的编解码框架。在该框架中，翻译系统先对源语端的序列信息进行抽取，将其压缩为一个固定维度的向量，然后在解码端对信息进行展开，得到目标语端所预测出的词汇序列。编码器和解码器之间使用注意力机制进行连接，通过解码端的当前状态对下一时刻待预测单词和源语序列中的片段对应关系得到最相关的上下文向量，保证信息二者信息对应上的高效准确。

在模型中网络结构的设计部分，本文主要介绍了在神经机器翻译中比较知名的三种结构：基于循环神经网络的机器翻译系统、基于卷积神经网络的机器翻译系统、基于自注意力机制的机器翻译系统。后两种模型结构于2017年提出，相较传统的循环神经网络而言，降低了序列中相距较远的相关词汇之间处理的距离，因此对于长句子的处理具有更优的性能。

本章对神经机器翻译的基本方法进行了介绍，在后续的内容中将对其中基于自注意力机制的神经机器翻译方法的解码过程进行加速。

第3章 基于coarse-to-fine的注意力机制

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之前备受瞩目的基于统计的翻译系统而言，在性能上已经得到了十足的进步。但由于神经网络本身涉及大量的矩阵运算，因此在训练以及解码上相对之前的方法会更加耗时。对于这两方面的耗时，在实用系统中往往对于解码的时间消耗更为看重。为使得神经机器翻译系统能够在实际中被应用起来，我们必须要求系统在解码的过程中具有较高的响应速度，否则面对难以接受的延迟，即使翻译系统具有更为优秀的性能，在很多场景下（如在线翻译、翻译机等）用户也很难接受。

对于目前性能最佳的机器翻译模型——基于自注意力的神经机器翻译系统，由于其中存在大量的注意力权重计算，因此导致系统在解码过程中的速度并不能得到保障，很难直接将其应用到实际中来为用户提供稳定的服务。因此如何提升基于自注意力机制的机器翻译模型解码速度成为了一大亟待解决的问题。

基于自注意力的神经机器翻译模型中，由于其结构摒弃了循环神经网络、卷积神经网络等传统的网络结构，因此其结构的组成大部分都是注意力机制的反复使用，其中包括编码端的自注意力结构、解码端的自注意力结构以及编码器解码器之间的注意力结构。整个模型通过堆叠多层的注意力子模块最终得到整体的设计。在实验系统中我们统计了基于自注意力机制的神经机器翻译系统中各个模块的时间消耗，如表3.1所示，展示了在基于自注意力机制的神经机器翻译系统在解码过程中的各部分耗时占比，该测试模型参数如表3.2所示，展示了在基于自注意力机制的神经机器翻译模型解码的过程中，注意力操作（表中的Decoder-Block-SelfAtt和Decoder-Block-EncDecAtt项，二者分别指的是编码器解码器内部的自注意力操作和它们之间的注意力操作）在解码时间消耗上有着较大的比例，占整体时间消耗的63.99%，因此如果能够在现有注意力机制的结构上进行改动，使得在不影响性能的同时让解码速度得到提升将会大大提高模型的可用性。

前文提到，不论是自注意力机制还是普通的注意力机制，其计算均可以归结为公式(3.1)所示。

(3.1)

表3.1 解码过程中各部分操作时间占比

Table 3.1 Percentage of operating time for each part of the decoding process

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作名称 | 次数/次 | 总时间/ms | 平均时间/ms | 占比/% |
| Build-Batch | 104 | 23.05 | 0.22 | 0.04 |
| Encoder | 104 | 904.02 | 8.69 | 1.70 |
| Beam-Search-Prepare | 4620 | 51.87 | 0.01 | 0.10 |
| Decoder-Embedding | 4620 | 179.41 | 0.04 | 0.34 |
| Decoder-Att-QKV | 55440 | 5025.48 | 0.09 | 9.45 |
| Decoder-Att-Split | 55440 | 4734.98 | 0.09 | 8.90 |
| Dot-Pro-Scale | 55440 | 382.10 | 0.01 | 0.72 |
| DotPro-BMM | 110880 | 20180.05 | 0.18 | 37.93 |
| DotPro-Softmax | 55440 | 1180.08 | 0.02 | 2.22 |
| Decoder-Att-DotPro | 55440 | 22058.40 | 0.40 | 41.46 |
| Decoder-Att-Merge | 55440 | 667.08 | 0.01 | 1.25 |
| Decoder-Att-Out | 55440 | 1155.09 | 0.02 | 2.17 |
| Decoder-Block-SelfAtt | 27720 | 17696.92 | 0.64 | 33.26 |
| Decoder-Block-LN | 83160 | 3304.18 | 0.04 | 6.21 |
| Decoder-Block-EncDecAtt | 27720 | 16347.75 | 0.59 | 30.73 |
| Decoder-Block-FNN | 27720 | 2568.77 | 0.09 | 4.83 |
| Decoder-Multi-Block | 4620 | 40103.38 | 8.68 | 75.38 |
| Update-Cache | 4620 | 1719.61 | 0.37 | 3.23 |
| Decoder | 4620 | 42105.59 | 9.11 | 79.14 |
| Softmax | 4620 | 3643.85 | 0.79 | 6.85 |
| TopK-in-update | 4620 | 1890.02 | 0.41 | 3.55 |
| Beam-Search-Candidate | 58594 | 329.44 | 0.01 | 0.62 |
| Beam-Search-Update-Beam | 58594 | 123.03 | 0.00 | 0.23 |
| Beam-Search-Update | 4620 | 2498.14 | 0.54 | 4.70 |
| Model-Update-State | 4620 | 3171.60 | 0.69 | 5.96 |
| Trans-Batch | 104 | 52511.64 | 504.92 | 98.70 |
| Rerank | 1664 | 192.18 | 0.12 | 0.36 |
| Get-Best-Translation | 1664 | 445.70 | 0.27 | 0.84 |
| Beam-Search-Forch-Finish | 4 | 0.01 | 0.00 | 0.00 |

表3.2 解码速度相关的模型参数

Table 3.2 Model parameters related with speed of decoding process

|  |  |
| --- | --- |
| 参数名 | 参数值 |
| hidden-size | 512 |
| filter-size | 2048 |
| num-head | 8 |
| num-hidden-size | 6 |

将公式中的计算拆分开来，我们可以得到四个子运算：

(1) 和的乘法运算：执行query矩阵*Q*和key矩阵*K*的矩阵乘法操作，其中矩阵*Q*的大小为，矩阵的大小为，其中*n*为序列长度（编码序列或解码序列，具体为哪一个要根据该注意力操作属于哪个模块），为模型隐藏层大小，因此该操作的时间复杂度为。

(2) 按位除以的运算：对的结果按位除以，该操作相当于是对第一步中的矩阵结果进行缩放。由于的结果矩阵大小为，因此该操作的时间复杂度为。

(3) *softmax*的运算：这一步骤将经过缩放的乘积结果按行进行归一化，得到它们的概率表示形式。这个过程中*softmax*的计算公式如式(3.2)。

(3.2)

从公式中我们可以看出这部分*softmax*的时间复杂度为。

(4) 概率乘以的运算：这个步骤将上一步中得到的概率作为权重乘到value矩阵*V*上，由于权重概率矩阵和*V*矩阵的大小分别为和，因此这部分操作的时间复杂度为。

通过上述分析我们可以看出影响注意力操作速度的主要参数为和。由于为源语或译文的序列长度，与模型的结构设计无关，因此我们很难通过对模型的修改降低*n*值，进而实现对解码过程的加速。而为模型中隐层的大小，我们可以通过参数设置的方式降低它达到我们解码加速的目的，但通过实验我们可以知道降低了隐层大小对于翻译模型性能的影响相对来说还是较大，因此我们不能简单地对减低这个参数来达到我们加速的目的。这里我们通过对现有网络结构进行修改，在本章提出一种基于coarse-to-fine的方法对注意力机制进行加速，从而使得基于自注意力机制的神经机器翻译系统的解码速度得到提升。

3.2 方法描述

3.2.1基于coarse-to-fine的注意力机制

根据上一小节的分析，我们可以确定注意力操作的时间占比在整个基于自注意力机制的翻译系统中占比较大，而且如果我们想对这部分进行提速的话，只能从序列长度和隐层大小两个角度去寻找方法。本章提出的基于coarse-to-fine的注意力机制主要是从削减隐层大小的角度出发，旨在不损失（或损失较少）性能的前提下对系统中注意力操作进行加速。

前文提到，对于神经机器翻译系统而言，目前的主流方法为采用编码解码框架对翻译系统进行建模，而编码解码框架的核心思想在于将机器翻译的过程看作是对源语的抽象和对目标语的生成过程。基于编码解码框架有着各种各样不同的网络结构，它们之间的差别主要是采用了不同的方式对源语和目标语种的信息进行抽象，比如循环神经网络通过循环单元按照语言序列的顺序对源语和目标语中的语义进行学习。在基于自注意力机制的神经机器翻译系统中，从某种程度上而言，模型结构的可解释性更强了。以编码端为例，在每一层内主要包含自注意力的子层和前馈网络的子层。其中前馈网络的结构主要目的在于增加模型的表达能力，将自注意力子层的信息进行整合后送入下一层。而自注意力机制的操作则可以看作是对源语词向量的富集抽象，该过程如图3.1所示。

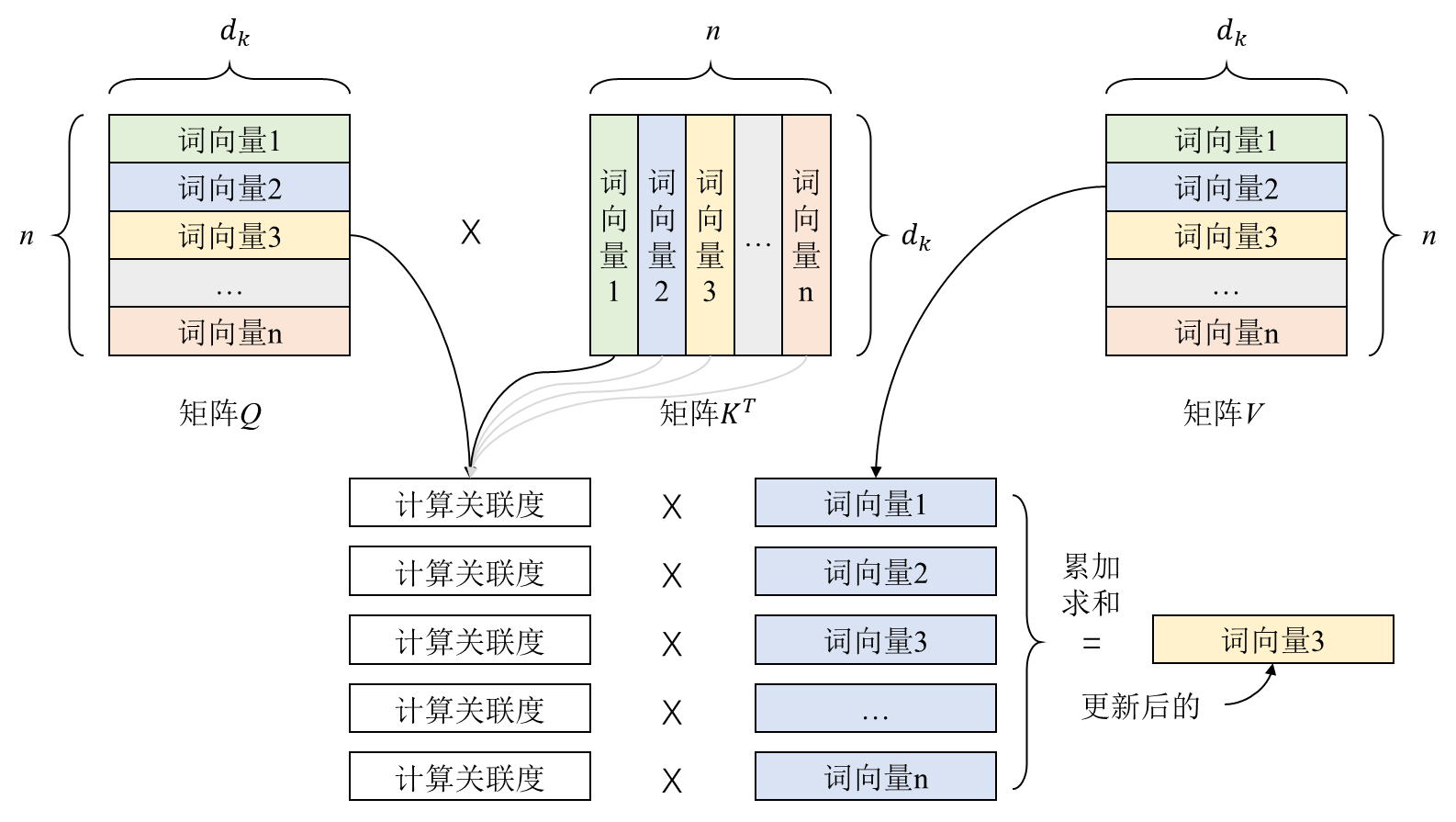


图3.1 自注意力机制的实例

Fig. 3.1 Example of self-attention mechanism

从图3.1中我们可以看出自注意力机制的使用实际上是一种对词向量的浓缩。在注意力机制的计算过程中，首先我们将计算的结果，然后对该结果进行缩放后再通过*softmax*函数进行归一化。我们可以看到，这部分的计算对应着图中关联程度计算的模块。将矩阵乘法的过程拆开来看，我们可以发现实际上相当于是每个词的向量表示在和句中的每个词汇进行运算，最终得到当前词和每个词汇之间的关联程度，该过程中得到的每个词和句中词汇之间的关联关系矩阵*p*可以表示公式(3.3)。

(3.3)

这时我们将每个词的关联程度作为一个概率权重乘到相应的词向量中，就得到了若干经过不同权重调整过的词向量，最后我们将这些词向量进行累加求和得到最终当前词的更新后的词向量，该词向量将传递到下一层中继续运算，将这些向量放在一起可以表示为自注意力操作的输出，公式如式(3.4)所示。

(3.4)

其中等式左边代表未经残差和正规化的自注意力部分输出。

举个例子，比如输入到系统中的源语句子为“你 知道 去 北京站 的 路 怎么 走吗”，其中以“北京站”的词向量为例，我们在第一层的自注意力计算过程中将会计算这个词的向量表示和句中其它词汇之间的关联关系，我们发现得到的注意力权重里“北京站”这个词和“北京站”自身的关联程度最大，概率为0.5，其次为“路”，概率为0.3，再次为“走”，概率为0.1，剩下的词汇和这个词的关联度相对较小。这时我们将这些关联程度作为句中每个词向量的权重对它们进行加权求和，得到更新后的“北京站”的词向量，具体来说就是公式如式(3.5)所示。

(3.5)

我们可以看到，原本句中每个词汇的词向量虽然都处在同一个向量空间中，但彼此之间并没有因为这句话的存在而产生特殊的联系，因此我们说这时候的词向量仍然是相互孤立的。但经过了自注意力机制的更新后，每个词汇的词向量将不仅仅包含自身的语义信息，还融入了本句话中和它最相关的词汇的信息，这里我们将其称为“词向量+”。在上述例子中我们可以看到“北京站”这个词经过了第一层的自注意力运算，其中的信息虽然大部分仍为“北京站”的词向量，但根据不同的关联程度包含了譬如“路”、“走”等词汇的信息。这样更新后，在下一层中，我们得到的向量中每个位置仍代表着当前词汇的词向量，但其中的信息更为丰富了，最终得到的矩阵局部的每个向量上可以看作是源语端每个词汇的“词向量+”表示，整体上来看可以视为是源语信息的浓缩，即为在编码解码框架中所需要的源语端信息。这个过程我们将其称为“词向量的浓缩”。

对于如何对注意力机制进行加速，前文中通过对注意力机制复杂度的分析可以知道如果可以在计算过程中通过某种方式降低的话，注意力机制的执行效率将会有所提升。但由于直接削减将使得矩阵*Q*、*K*、*V*的维度均发生变化，降低了模型对语言的建模能力，因此一般来说翻译系统的性能将会下降得较为明显，因此本文提出一种软性的降低的方式，保持原本的矩阵*Q*、*K*、*V*维度不变，仅在计算词向量之间关联程度的时候对原有词向量进行一定的压缩，将其映射到相对低维的空间中进行注意力权重的计算，具体计算过程如图3.2所示。

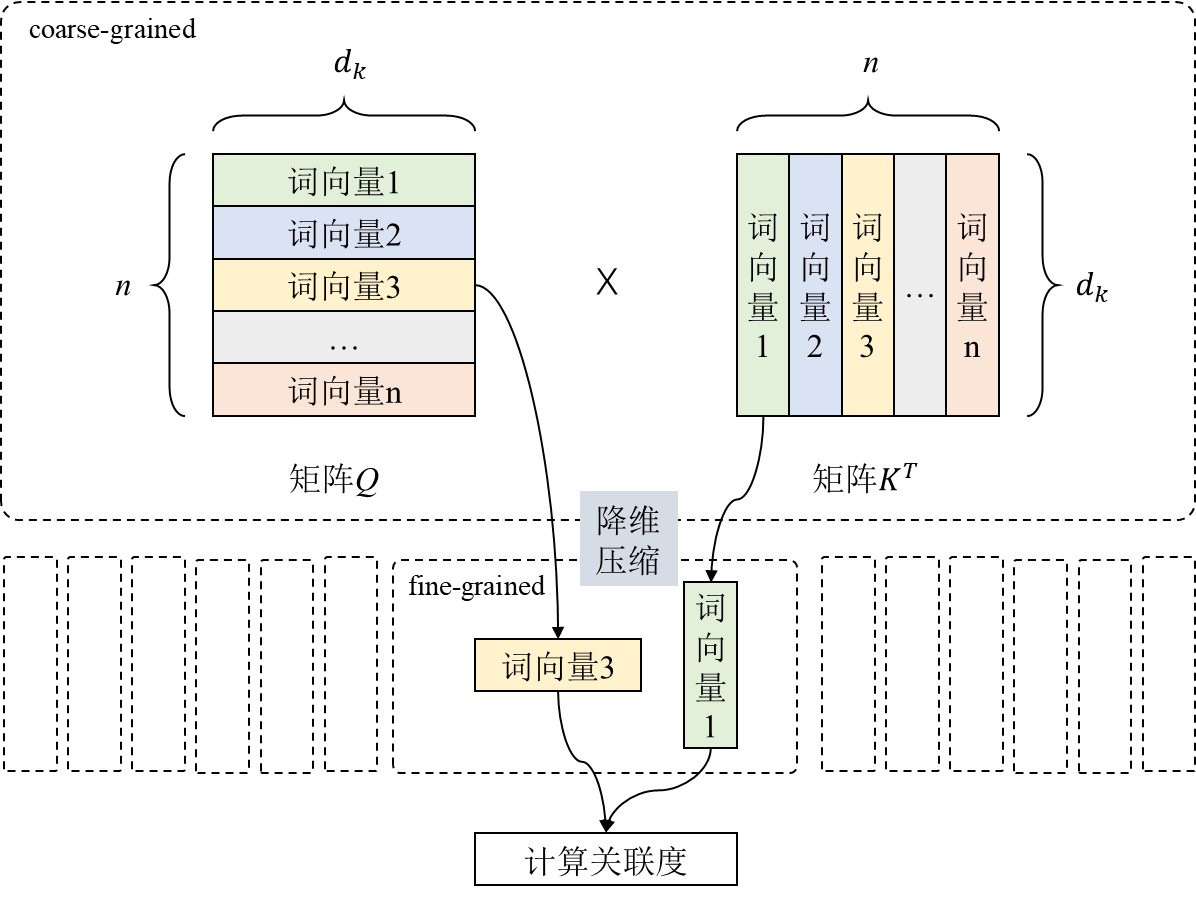


图3.2 基于coarse-to-fine的自注意力机制

Fig. 3.2 Self-Attention mechanism based on coarse-to-fine method

从图3.2中所示，我们在使用coarse-to-fine的方法进行自注意力机制的操作的时候，实际上相当于是先将原本的词向量映射到较低维空间后再进行关联度的计算，具体计算如式(3.6)所示。

(3.6)

其中和为对矩阵*Q*和的压缩矩阵，其维度为，其中为压缩后的词向量维度，该值小于。这样得到的压缩后矩阵将原本计算时候的复杂度将从降低为，能够有效加速注意力机制的运算速度。

根据之前分析的结果，我们知道基于自注意力机制的神经机器翻译系统实际上是一个对词向量不断浓缩的过程，每次浓缩的时候（即每层中的注意力子层），词向量将根据当前表示自身的词向量信息（或词性量+信息）计算其和句中每个词汇之间的关联关系，以得到从其他词中提取信息的权重。我们这种基于coarse-to-fine的注意力方式实际上认为每个词向量中所包含的信息量并不相同，我们在参数中设置的词向量维度只不过是使用者对词向量信息的一个主观估计，一个性能优秀的模型中该值所对应的空间一般来说会大于我们所要表达的词汇信息量。但在实际使用中，词向量的表达能力往往会远大于当前时刻表达词汇所需要的空间大小，这样做实际上在运算中就存在着大量的不必要的计算。面对这些词向量的冗余空间，我们基于coarse-to-fine的注意力机制通过一个线性变换将其映射到低维的空间中，实际上的思想就在于我们将当前词向量中的冗余空间压缩掉，得到一个信息量更饱满的词向量再进行关联度的计算。从某种程度上来说，如果我们能够准确地预估原本词向量中具有实质信息的部分大小，那么理论上我们就能够在完全不损失信息量的前提下对词向量的表示空间进行压缩。词向量压缩的过程如图3.3所示。

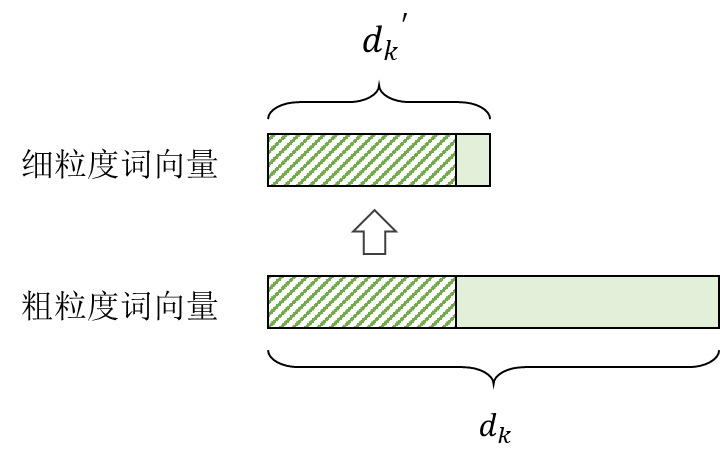


图3.3 粗粒度细粒度词向量转换

Fig. 3.3 Conversion between coarse-grain and fine-grained embedding

在具体实现的过程中，由于模型本身其实在进行关联度计算之前也进行了一个等维度的线性变换，因此我们不会向其中引入更多的运算，仅仅是将其中的线性变换维度降低到我们希望的大小中，保证系统的加速效果。

在之前的例子中我们都以编码端或解码端的自注意力机制为例进行说明，但实际上关于coarse-to-fine的注意力机制完全能够应用到编码器解码器之间的注意力机制上来。只不过我们在运算的时候矩阵*Q*来自于解码端当前层中自注意力机制的输出，矩阵*K*来自于编码端的输出，二者并非同一个矩阵。同样，在可解释性上这个过程相当于是对当前已翻译出的词汇序列所对应的词向量和源语端的序列词向量进行压缩，利用去除冗余后的向量进行关联度的计算。

3.2.2考虑信息增长的coarse-to-fine的注意力机制

前文的分析中提到，在基于自注意力机制的神经机器翻译系统中，对于源语或目标语的抽象过程可以看作是词向量的浓缩过程，在这个过程中，我们将利用每一层中现有的词向量状态分析词汇之间的相关程度，将其作为浓缩词向量过程中的权重，对向量信息进行抽取，得到更新后的词向量传递到上一层中继续计算。整个过程中我们可以知道每个词汇的词向量不断地收集到了来自于其它词汇的语义补充，因此在这个过程中词向量的信息是一个逐步增加的过程，因此表达该信息所需要的词向量空间也相应有着更高的要求。整个过程如图3.4所示。

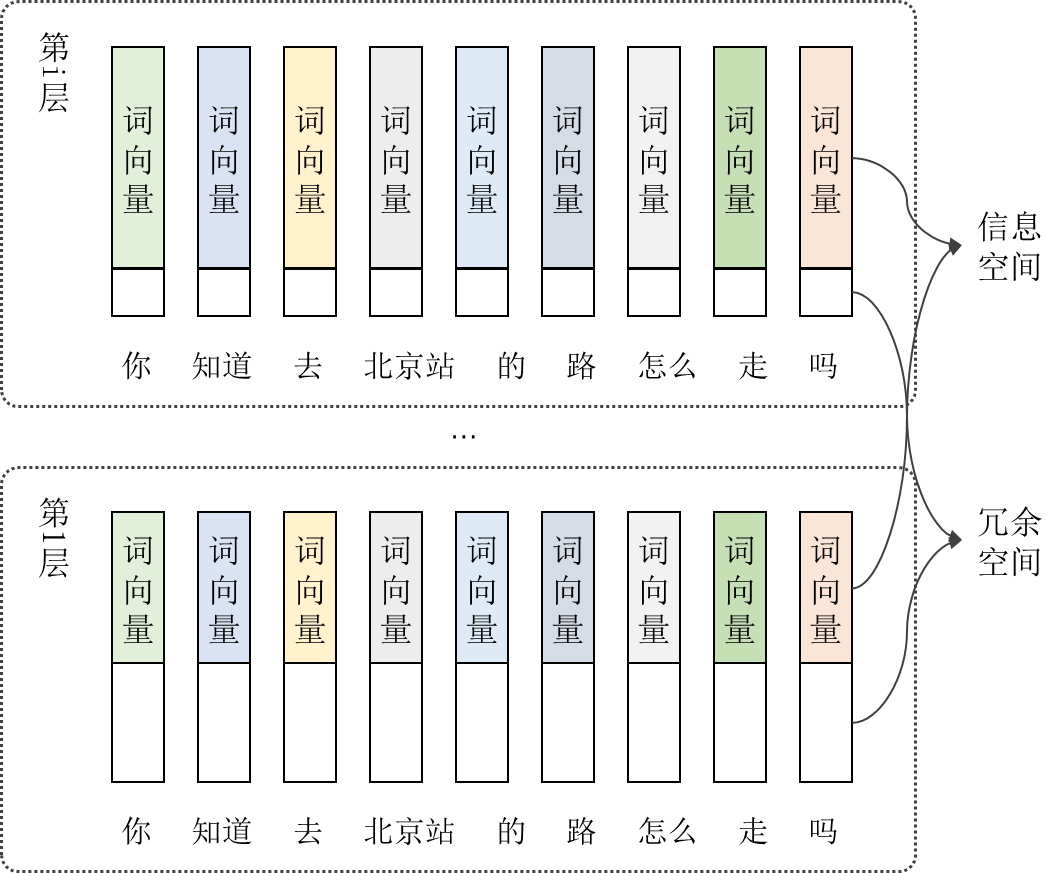


图3.4 词向量信息增长的过程

Fig. 3.4 Process of growth of embedding information

如图3.4所示的例子，我们可以看到最初在第一层中，“你 知道 去 北京站 的 路 怎么 走 吗”这句话里面每个词汇由于其词向量中的信息仅仅包含其自身词汇的语义表示，因此真实占用的信息空间相对较少，这样导致对于固定维度的向量空间中大部分空间仍处于一个空闲的状态，但经过了若干层的传递之后，我们发现句中每个词所对应的信息量更加丰富了，所表达的语义中除了自身的词汇本身的信息，还包含了这句话句中和当前词汇最相关的词汇的语义信息（权重占比相对自身而言较小），这样就使得每个词所对应的词向量中真正有用的信息量相对底层而言更多了，由于词向量的表示维度空间是固定的，因此这时我们词向量中冗余的空间就相对较小了，整体词向量的空间利用率相对更高了。

上一小节中提到的coarse-to-fine的方法对所有层都一视同仁，在计算词向量之间的相关程度时对于采用相同的削减率得到细粒度的，具体公式如式(3.7)和(3.8)所示。

(3.7)

(3.8)

其中为对于的保留率，也可以看作是对词向量中对真实有效的信息空间占比情况的预估，为对的削减率，也可看作是词向量中冗余度的估计。

但实际上我们通过分析可知不同层之间词向量的冗余度是不相同的，具体来说其中的真实的信息占比相对来说是越来越大的，而冗余的部分则越来越稀少，因此我们对于每一层都采用相同削减率就显得并不适合。因此本小节提出考虑信息增长的基于coarse-to-fine的注意力机制，这种方法将根据每层词向量不同的信息占比使用不同的削减率，使得我们的削减方式既能有效对冗余部分进行裁剪，同时也能将尽可能的保证信息的完整程度。

具体来说，在计算词汇之间相关度的时候，对于相对靠近底层一些的词向量而言，我们将采用一个较高的削减率来对进行削减；而对于相对高层一些的词向量我们将采用一个相对较低的削减率对进行调整，这样做可以保证在不损失（或较少损失）机器翻译模型性能的同时在整体上达到一个高削减率，使得模型在解码的过程中速度更快，效率更高。

3.3 实验

3.3.1 实验设置

对本文中所提出的基于coarse-to-fine的注意力机制方法的实验验证主要是在中英方向的任务中进行，评价方式使用目前机器翻译自动评价方法中的BLEU值对结果进行验证（该分数用来评估机器翻译出的译文和参考译文之间的相似性，分值越高越好），脚本采用开源的测试工具multi-bleu.perl。

在该翻译任务中，训练数据来自中英方向的LDC数据（LDC2000T46、LDC2000T47、LDC2000T50、LDC2003E14、LDC2005T10、LDC2002E18、LDC2007T19、LDC2004T08），该数据全称为Linguistic Data Consortium，是一个由图书馆、大学、政府、企业和研究机构合办的企业，现由宾夕法尼亚大学负责主要运营。本实验中采用的训练集合中包括3,900万左右中文词和4,500万英文词。测试集采用NIST2004、NIST2005、NIST2006和NIST2008对翻译效果进行评价。为提高翻译系统性能，本系统对分词后的语料进行更细粒度的切分，这里采用BPE算法[42-43]对训练及测试的数据进行子词切分，其中BPE过程中的merge次数为32000，最终得到的源语端（中文）词表大小为43211，目标语端（英语）词表为31586。

在实现方面，有关于模型的训练，我们在开源的基于自注意力机制的神经机器翻译系统中Tensor2Tensor（T2T）中进行二次开发，该系统是深度学习模型的一个开源库，其中包括图像识别、语言模型、语音识别、文本摘要等相关任务，当然本文所主要关注的基于自注意力机制的神经机器翻译系统也是其中的一个重要组成部分。该开源库基于tensorflow进行搭建，由谷歌大脑团队进行开发及维护，旨在加速机器学习领域各个任务的进一步发展。由于在进行此项研究的时候Tensor2Tensor的最新版本为1.0.14，因此在实验中我们也是基于该版本进行的二次开发。但由于该版本中的机器翻译系统在解码的过程中效率较低，特别是每当每翻译一个batch的句子的时候就会重新读取一遍模型的参数，因此我们仅使用Tensor2Tensor开源系统进行翻译模型的训练，然后使用实验室自研的解码器对文字进行翻译，过程中记录采用了基于coarse-to-fine方法的注意力机制的模型和原本的模型性能和速度上的差别。

关于实验设备我们采用带图形处理器的服务器对系统进行训练和解码，通过图形处理器在矩阵运算方面的优势来提升系统的训练效率和解码效率，该设备的具体信息如表3.3所示。

表3.3 服务器配置

Table 3.3 Server configuration

|  |  |
| --- | --- |
| **组件** | **配置** |
| 中央处理器 | Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz |
| 内存 | 128GB |
| 图形处理单元 | NVIDIA 1080TI |
| 显存 | 11GB |
| 硬盘 | 3TB |

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。在本实验中我们所使用的模型参数集为transformer\_base\_single\_gpu。该集合中具体的参数如表3.4所示。其中更新次数是根据训练数据中子词的数量以及batch大小统计而来，65000次更新相当于训练数据重复10轮左右。另外我们在实际中使用的模型采用了checkpoint ensemble的方法，集成了15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。

表3.4 模型参数配置

Table 3.4 Model parameter configuration

|  |  |
| --- | --- |
| 参数 | 值 |
| 隐层大小 | 512 |
| batch大小 | 8192 |
| 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[44] | 0.1 |
| 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 | 65000 |

实验中我们在进行解码实验的时候所使用的参数如表3.5所示。

表3.5 解码参数配置

Table 3.5 Decoding parameter configuration

|  |  |
| --- | --- |
| **参数** | **值** |
| batch大小 | 16 |
| 长度惩罚 | 1.3 |
| beam大小 | 6 |

3.3.2 实验结果

3.3.2.1 模型性能

本小节主要对比基于coarse-to-fine的注意力机制和传统的注意力方法在基于自注意力机制的神经机器翻译系统中的性能，其中基于coarse-to-fine的方法我们将对前文提出的两种模式进行实验，主要分为普通的模式和考虑信息增长的模式，我们将对它们在中英机器翻译的任务上的性能做详细对比。

首先我们对比普通的coarse-to-fine方式在不同削减率的情况下性能和传统的不对进行削减的方式性能的差异变化，当对模型中的自注意力机制（包括编码端和解码端两部分）进行削减的时候，在NIST的四个测试集上结果如表3.6所示。

表3.6 翻译系统在对自注意力操作采用不同保留率情况下的BLEU得分

Table 3.6 BLEU scores for translation system with different reservation rate on self-attention

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **系统** | **保留率α** | **测试集** | | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 1 | 52.04 | 45.72 | 46.36 | 37.7 | 45.46 |
| baseline+保留率0.75 | 0.75 | 52.02 | 45.93 | 45.7 | 37.68 | 45.33 |
| baseline+保留率0.5 | 0.5 | 52.06 | 45.81 | 46.07 | 37.71 | 45.41 |
| baseline+保留率0.375 | 0.375 | 51.4 | 44.32 | 44.8 | 37.23 | 44.44 |
| baseline+保留率0.25 | 0.25 | 51.47 | 43.9 | 44.31 | 37.28 | 44.24 |
| baseline+保留率0.1875 | 0.1875 | 50.9 | 43.48 | 43.32 | 36.87 | 43.64 |
| baseline+保留率0.125 | 0.125 | 49.69 | 42.23 | 42.58 | 35.99 | 42.62 |

除了对自注意力操作进行削减之外，我们也对比了普通的coarse-to-fine方式在对模型中的编码器解码器之间的注意力机制进行削减的时候，在NIST的四个测试集上结果，如表3.7所示。

表3.7 翻译系统在对编码器-解码器之间的注意力操作采用不同保留率情况下的BLEU得分

Table 3.7 BLEU scores for translation system with different reservation rate on encoder-decoder-attention

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **系统** | **保留率α** | **测试集** | | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 1.00 | 52.04 | 45.42 | 46.79 | 37.70 | 45.49 |
| baseline+保留率0.75 | 0.75 | 52.10 | 45.90 | 45.59 | 38.31 | 45.48 |
| baseline+保留率0.5 | 0.50 | 51.55 | 45.63 | 45.72 | 37.84 | 45.18 |
| baseline+保留率0.375 | 0.375 | 51.92 | 44.77 | 45.48 | 38.63 | 45.20 |
| baseline+保留率0.25 | 0.25 | 51.35 | 44.83 | 44.21 | 37.54 | 44.48 |
| baseline+保留率0.1875 | 0.1875 | 51.16 | 43.69 | 45.07 | 37.45 | 44.34 |
| baseline+保留率0.125 | 0.125 | 51.29 | 44.37 | 43.44 | 36.47 | 43.89 |

另外，当同时对模型中的编码器解码器内部的自注意力操作和其二者之间的注意力操作进行削减的操作的话，模型解码的速度将会进一步提升，但是其性能也将有所下降，在NIST的四个测试集上结果，如表3.8所示。最后，我们还对在上一章节提出的考虑信息增长的coarse-to-fine方法进行了实验验证，在保证整体参数量相同的基础上为下层提供较高的削减率，上层削减率较低。整体保留率和每一层中的保留率如表3.9所示。

表3.8 翻译系统在对全部注意力操作采用不同保留率情况下的BLEU得分

Table 3.8 BLEU scores for translation system with different reservation rate on all attention

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **系统** | **保留率α** | **测试集** | | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 1.00 | 52.04 | 45.44 | 46.78 | 37.57 | 45.46 |
| baseline+保留率0.75 | 0.75 | 51.82 | 45.91 | 45.19 | 38.04 | 45.24 |
| baseline+保留率0.5 | 0.50 | 51.81 | 45.90 | 45.70 | 37.92 | 45.33 |
| baseline+保留率0.375 | 0.375 | 50.76 | 43.30 | 43.65 | 37.21 | 43.73 |
| baseline+保留率0.25 | 0.25 | 50.43 | 43.07 | 43.22 | 37.07 | 43.45 |
| baseline+保留率0.1875 | 0.1875 | 48.94 | 41.55 | 42.77 | 35.38 | 42.16 |
| baseline+保留率0.125 | 0.125 | 48.04 | 42.23 | 41.41 | 34.78 | 41.61 |

表3.9 整体保留率与每一层的保留率之间的对应

Table 3.9 Corresponding relationship between the overall reservation rate and the reservation rate of each layer

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **整体保留率** | **每一层的保留率** | | | | | |
| **第1层** | **第2层** | **第3层** | **第4层** | **第5层** | **第6层** |
| 0.75 | 0.59 | 0.66 | 0.72 | 0.78 | 0.84 | 0.91 |
| 0.50 | 0.34 | 0.41 | 0.46 | 0.53 | 0.59 | 0.66 |
| 0.375 | 0.22 | 0.28 | 0.34 | 0.41 | 0.47 | 0.53 |
| 0.25 | 0.09 | 0.16 | 0.22 | 0.28 | 0.34 | 0.41 |
| 0.1875 | 0.03 | 0.09 | 0.16 | 0.22 | 0.28 | 0.34 |

由于基于自注意力机制的神经机器翻译系统由若干层组成，上层的输入为下层的输出，因此我们必须保证模型中每一层都有足够的能力处理当前的信息。而当保留率本身过低的时候（指的是保留很少的向量空间），在第一层中对于的压缩将会更为严重，很难保留足够的空间对信息进行提取，而如果第一层都不能把语句中的信息有效地抽象出来，那么接下来的层中即使有着优秀的表达能力，但也无法对信息进行补救。因此本实验并没有对整体保留率为0.125的情况采用考虑信息增长的coarse-to-fine方法进行实验。

我们针对模型中的自注意力机制，将考虑层与层之间信息增长的coarse-to-fine方式在不同保留率的情况下性能和普通的coarse-to-fine方式进行对比，在NIST的四个测试集上结果如表3.10所示。

表3.10 翻译系统在对自注意力操作采用不同保留率情况下的BLEU得分（考虑信息增长）

Table 3.10 BLEU scores for translation system with different reservation rate on self-attention

(considering the growth of information)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **系统** | **保留率α** | **测试集** | | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 1.00 | 52.04 | 45.49 | 46.84 | 37.73 | 45.52 |
| baseline+保留率0.75 | 0.75 | 52.26 | 45.94 | 45.57 | 37.57 | 45.34 |
| baseline+保留率0.5 | 0.50 | 51.75 | 46.01 | 45.38 | 37.73 | 45.22 |
| baseline+保留率0.375 | 0.38 | 51.78 | 44.22 | 44.84 | 38.21 | 44.76 |
| baseline+保留率0.25 | 0.25 | 51.59 | 44.21 | 44.32 | 38.13 | 44.56 |
| baseline+保留率0.1875 | 0.19 | 51.13 | 43.46 | 44.79 | 37.27 | 44.16 |

3.3.2.2 模型速度

上一小节中我们通过实验对比了不同的削减对象（对编码器、解码器内部的自注意力操作进行削减还是对二者之间的注意力操作进行削减）、不同削减程度（保留率设置为多少）以及不同的削减方式（是否根据层之间的信息增长对不同层的保留率进行调整）对基于自注意力机制的神经机器翻译系统的性能影响，发现在对自注意力操作进行削减时，当保留率大于0.5的时候，coarse-to-fine的操作对性能几乎没有影响。本小节在该设置基础上继续研究，通过实验对该设置下的加速效果进行测试。在实验系统中我们统计了基于采用coarse-to-fine的方式的自注意力机制的神经机器翻译系统中各个模块的时间消耗，如表3.11所示。

3.3.3 结果分析

3.3.3.2 模型性能

关于将基于coarse-to-fine的注意力机制应用到神经机器翻译系统之后，根据上一小节的实验结果我们可以得到一些经验性的结论。

(1) 当使用coarse-to-fine对模型中自注意力机制的进行削减的时候，保留率为0.5及以上几乎不影响模型性能。

如图3.5所示，我们可以看到当我们对基于自注意力机制的神经机器翻译系统进行中的自注意力机制进行削减的时候，随着保留率的逐渐降低，机器翻译模型的性能是逐渐下降的，但当保留率为0.5及0.5以上时，这种程度的削减几乎不影响模型的性能。该现象从侧面说明在中英翻译的任务中，模型每一层的词向量信息所需要的表示空间仅为的一半，因此我们在计算注意力权重的时候对其进行削减后性能几乎不受影响。

表3.11 解码过程中各部分操作时间占比

Table 3.11 Percentages of operating time for each part of the decoding process

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作名称 | 次数/次 | 总时间/ms | 平均时间/ms | 占比/% |
| Build-Batch | 104 | 22.77 | 0.22 | 0.04 |
| Encoder | 104 | 852.38 | 8.20 | 1.68 |
| Beam-Search-Prepare | 4640 | 52.31 | 84.72 | 0.10 |
| Decoder-Embedding | 4640 | 177.84 | 0.01 | 0.35 |
| Decoder-Att-QKV | 55680 | 4424.95 | 0.04 | 8.74 |
| Decoder-Att-Split | 55680 | 4460.39 | 0.08 | 8.81 |
| Dot-Pro-Scale | 55680 | 371.44 | 0.01 | 0.73 |
| DotPro-BMM | 111360 | 19506.06 | 0.18 | 38.53 |
| DotPro-Softmax | 55680 | 1179.06 | 0.02 | 2.33 |
| Decoder-Att-DotPro | 55680 | 21359.18 | 0.38 | 42.19 |
| Decoder-Att-Merge | 55680 | 662.40 | 0.01 | 1.31 |
| Decoder-Att-Out | 55680 | 1157.64 | 0.02 | 2.29 |
| Decoder-Block-SelfAtt | 27840 | 16030.92 | 0.58 | 31.66 |
| Decoder-Block-LN | 83520 | 3307.22 | 0.04 | 6.53 |
| Decoder-Block-EncDecAtt | 27840 | 16419.27 | 0.59 | 32.43 |
| Decoder-Block-FNN | 27840 | 2567.61 | 0.09 | 5.07 |
| Decoder-Multi-Block | 4640 | 38497.35 | 8.30 | 76.04 |
| Update-Cache | 4640 | 1351.65 | 0.29 | 2.67 |
| Decoder | 4640 | 40127.66 | 8.65 | 79.26 |
| Softmax | 4640 | 3651.22 | 0.79 | 7.21 |
| TopK-in-update | 4640 | 1894.18 | 0.41 | 3.74 |
| Beam-Search-Candidate | 59111 | 332.74 | 0.01 | 0.66 |
| Beam-Search-Update-Beam | 59111 | 123.92 | 0.00 | 0.24 |
| Beam-Search-Update | 4640 | 2504.47 | 0.54 | 4.95 |
| Model-Update-State | 4640 | 2616.29 | 0.56 | 5.17 |
| Trans-Batch | 104 | 49938.46 | 480.18 | 98.64 |
| Rerank | 1664 | 197.65 | 0.12 | 0.39 |
| Get-Best-Translation | 1664 | 440.12 | 0.26 | 0.87 |
| Beam-Search-Forch-Finish | 2 | 0.01 | 0.00 | 0.00 |

图3.5 削减自注意力机制时性能随保留率的变化

Fig. 3.5 Variation of performance with different reservation rates when self-attention parameter is reduced

(2) 编码器解码器之间的注意力机制相对其内部的自注意力机制包含更多需要表达的信息量。

图3.6中展示出了随着保留率的变化，在模型中不同的削减对象（编码器解码器内部的自注意力机制和二者之间的注意力机制）会有着不同的变化趋势。我们从图中可以看出当保留率为0.5的时候，对于自注意力机制进行削减尚未对性能产生明显的影响，但当以同样的保留率对编码器解码器之间的注意力机制进行削减的时候则出现了下降的趋势。我们从实验结果上能够看出，对于的削减程度而言编码器解码器之间的注意力机制相对来说对其的变化会更加敏感。从理论上对该现象进行解释也能够看出，编码器解码器内部的自注意力机制是在对同一种语言进行建模，实际上是为了去学习到该种语言中词汇之间的联系，而编码器解码器之间的注意力机制实际上是在对跨语言对进行建模，它的目的在于学习两种语言之间的对应关系。理论上来说后者由于存在语言之间的差异性，因此相较自注意力的机制所要学习的语言内逻辑更难以学习，同时也具有更大的信息量。由于上述原因，编码器解码器之间的注意力机制则需要更大的表示空间对语言对之间的信息进行抽象，因此当我们赋予它们相同的削减率的时候会发现编码器解码器之间的注意力操作对于削减的大小更为敏感，需要一个较高的保留率。

图3.6 不同削减对象时性能随保留率的变化

Fig. 3.6 Variation of performance with different reservation rates for different reduced objects

(3) 考虑信息增长的coarse-to-fine注意力机制相对普通的方式具有更优的性能。

如图3.7所示，我们可以看到当我们使用考虑信息增长的coarse-to-fine方式对模型中的进行削减时，当保留率小于0.5的时候相对普通的削减方式（每层固定保留率）具有优秀的性能这是因为位阶相对较低的层中词向量的信息相对来说较少，还没有收集大量的非自身的词汇信息，因此使用较高的削减率是没有问题的，而反过来高层中的词向量由于已经集合了大量其它的语义信息，因此我们为它们提供一个更高的保留率会有效减少信息的损失。

但是从下图中我们也看到，当保留率大于0.5的时候考虑信息增长的方法对于性能的提升似乎并没有一个正向的效果，而且还会使得性能有降低的趋势。根据上一点的结论我们可以看到在自注意力操作中，在每一层的词向量空间里都有着50%左右的冗余，而对于考虑信息增长的coarse-to-fine方法，其核心思想在于让下层具有更低的保留率，上层的保留率更高，因此当整体保留率为0.5的时候，下层的削减率为低于0.5的状态，上层的为高于0.5的状态。但是对于高层而言，由于50%的空间已经足以表达词向量所要包含的信息了，因此高于0.5的保留率并不会对性能有着正向的作用。与此同时，相对低层的词向量均获得了更低的保留率，使得性能一定程度上是有损的。因此这种情况下使用了考虑信息增长的coarse-to-fine方法反而对模型性能有着负面的作用。

图3.7 不同削减方法时性能随保留率的变化

Fig. 3.7 Variation of performance with different reservation rates for different reducing ways

3.3.3.2 模型速度

从实验结果来看通过使用coarse-to-fine的方式对翻译系统的注意力机制进行改进可以从一定程度上加速解码的效率。我们可以看到相对于未经改变的网络结构，使用了coarse-to-fine的自注意力机制后，模型在解码的过程中编码器部分提速6.01%，解码器部分提速4.93%。具体来说，在解码部分，注意力机制中*Q*、*K*、*V*三个矩阵的运算提速13.57%，矩阵拆分提速6.2%，矩阵合并提速10.39%，点乘操作提速2.27%。此外系统中为保留中间结果还临时变量的存储，更新这部分缓存提速了27.22%。

由于我们仅仅改变了在计算注意力操作中权重计算时的矩阵维度，因此在系统中的加速效果并没有十分巨大。但目前的这种方式可以有效地保证系统性能和未做削减的性能完全一致，因此可以视为是一种近乎无损的模型加速方式。

3.4 本章小节

本章主要包括三部分内容：基于coarse-to-fine方法的研究动机、两种该方法的具体实现描述、实验设置以及实验结果和分析。

首先我们从时间占比的角度分析了现有解码器在对基于自注意力机制神经机器翻译系统进行解码的时候各部分操作的时间占比，通过该分析得出结论，如果我们要对其进行加速主要应该关注的问题点是哪。通过分析后我们确定了对注意力操作进行改良可以有效提升解码器的解码速度。

然后我们从问题出发，提出了使用coarse-to-fine的方式对注意力操作中的权重计算部分进行改进，提出了两种具体的实现方式。其核心思想在于将计算注意力权重前的词向量先进行维度上的压缩后再进行相关度的计算，这样有效压缩了词向量中的冗余空间，同时减少了其在实际中的计算量，加速了系统的运行速度。

最后我们通过实验对两种方法进行验证，从实验结果中我们可以看到，当使用coarse-to-fine对模型中自注意力机制的进行削减的时候，保留率为0.5及以上几乎不影响模型性能；编码器解码器之间的注意力机制相对其内部的自注意力机制包含更多需要表达的信息量；考虑信息增长的coarse-to-fine注意力机制相对普通的方式在低保留率的情况下具有更优的性能。

本章中提出的基于coarse-to-fine的方式通过降低注意力操作本身的复杂度来提升模型的性能，下一章中将从减少注意力操作的使用次数的角度来对解码速度进行提升，二者角度不同，但目的都是为了提升翻译系统的解码速度。

第4章 基于共享权重的注意力机制

4.1 研究动机

现有的基于自注意力机制的神经机器翻译系统放弃使用传统的神经网络结构（比如循环神经网络、卷积神经网络等），因此其结构中除了简单的前馈网络之外几乎都是通过注意力机制对序列的转换进行建模的。在编码器、解码器内部通过自注意力的机制分别对源语言和目标语言进行建模，在编码器、解码器之间通过传统的普通注意力机制对两种语言词汇之间的对应关系进行学习。因此在解码的过程中大部分的时间消耗也是集中在了注意力操作中了。

前文已经分析过在解码器进行解码的过程中，注意力机制的时间占比占到整体解码时间的60%以上。针对注意力操作的大量耗时，在上一章中已经提出一中对其的优化方法，该方法旨在通过对注意力权重的计算部分的操作数（词向量）进行降维达到加速的目的。

本章将从另一个角度出发，通过减少计算注意力权重的次数提升系统在解码过程中的效率。注意力机制的计算过程可以分为两个部分，首先是query矩阵*Q*和key矩阵*K*之间的交互以得到词汇之间的关联程度，然后是利用该关联度和value矩阵V相乘得到新的矩阵（词向量的集合）。本文提出的基于共享注意力权重的方式主要是集中在如何有效利用其它层计算好的权重来作为本层中的注意力权重，节约掉计算该值的时间代价。后续将对该方法的思想和实现作具体介绍。

4.2 方法描述

4.2.1 面向自注意力部分的权重共享机制

在基于自注意力机制的神经机器翻译系统中，我们在前文中提到了其核心思想在于通过多层的叠加，不断从关联度较高的词向量中收集信息来逐渐丰富自身的词向量，通过该方法对源语或目标语进行建模。这种方法的目的在于不断地对句子中的每个单词信息进行充实，且在收集信息的过程中仍保留词汇和词汇之间的界限，这样有利于通过编码器解码器之间的注意力机制进行源语和目标语词汇之间的映射。因此在这种模式下，第一层和最终层之间每个词汇位置虽然存储信息上虽然有所不同，但是它们之间仍然对应着同一个词汇片段，信息的重叠率相对其他词而言还是相对较高的。那么基于这种情况，我们可以反向推出，对于同一个词向量而言，在一层一层地不断传递过程中，相邻的若干层之间词向量的变化程度其实并不大，如图4.1所示。

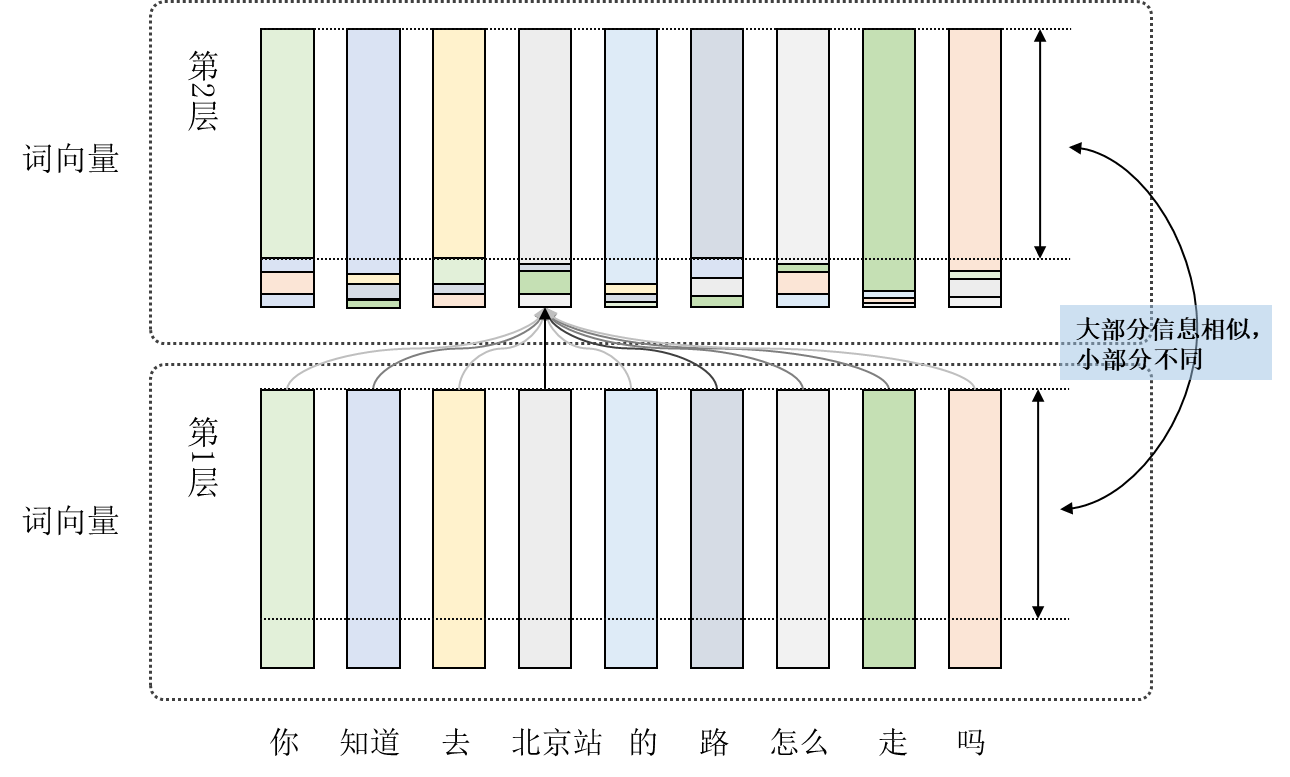


图4.1 相邻层之间词向量信息的变化

Fig. 4.1 Change of embedding information between adjacent layers

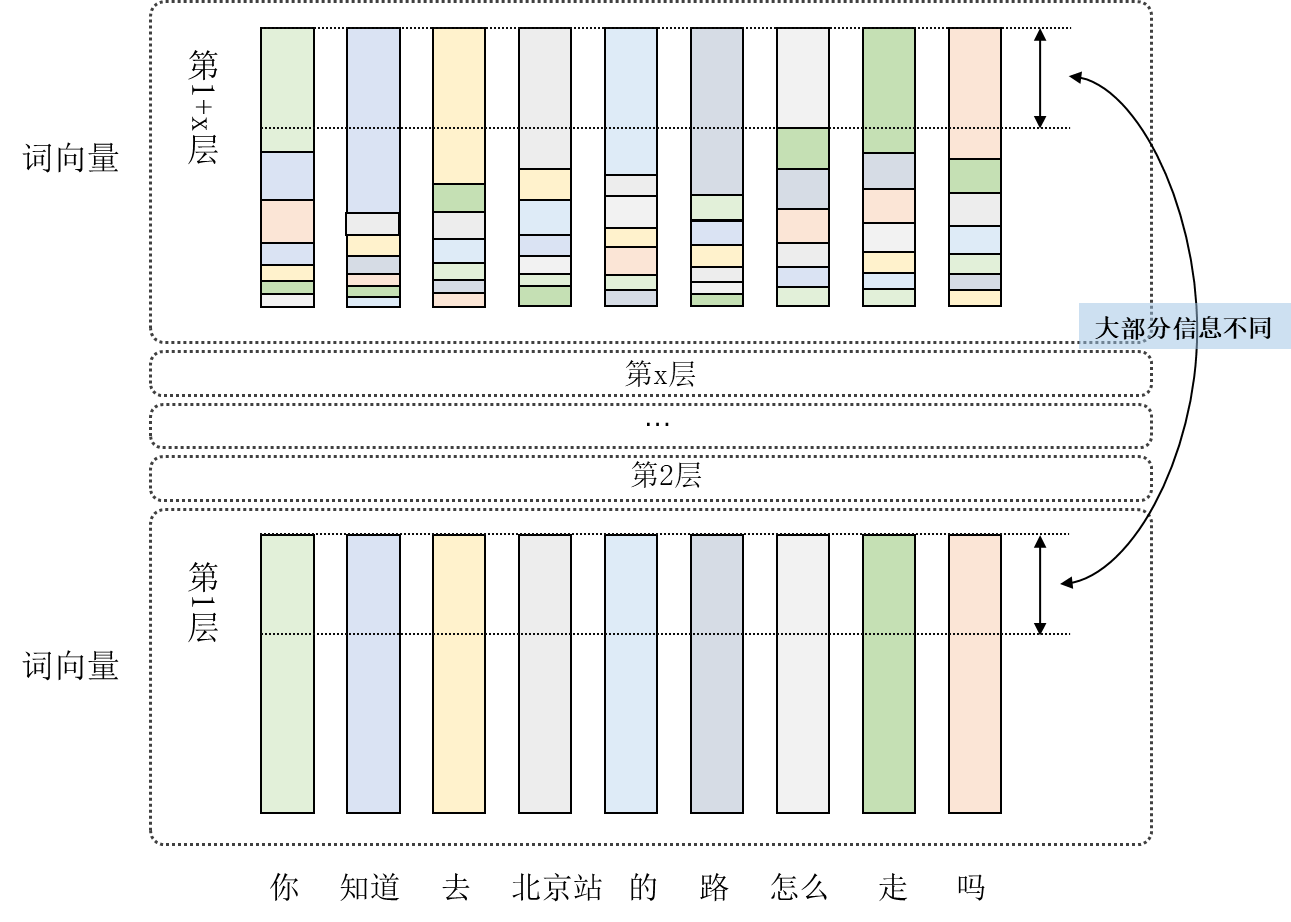


图4.2 相隔较远的层之间词向量信息的变化

Fig. 4.2 Change of embedding information between layers far away from each other

而随着层数的不断叠加，模型中每个位置对应的词向量经过不断地和其他词向量进行交互，得到新的词向量，因此其中将不断融入更多其他词的信息，使得在像个较远的两个层之间，同一个词所对应的词向量将发生较大的变化，如图4.2所示。

我们可以看到在相邻较近的层之间同一个词汇的词向量相对来说变化幅度并不大，但对于相隔较远的两层其变化的程度还是不可忽略的。根据这个特性，我们提出一种共享注意力权重的方式对模型进行加速。该方法每隔几层计算一次词汇之间的注意力权重，而对于不计算权重的层则直接复用下层计算好的权重向量来对本层的词向量进行注意力的操作（指的是权重乘以value矩阵*V*的过程）。举个例子来说，对于句子“你 知道 去 北京站 的 路 怎么 走 吗”，我们在第i层中计算出“北京站”的词向量和句中每个词的关联程度分别为：“你”🡪0.01，“知道”🡪0.02，“去”🡪0.02，“北京站”🡪0.8，“的”🡪0.01，“路”🡪0.02，“怎么”🡪0.01，“走”🡪0.02，“吗”🡪0.01。在第i+1层中，虽然词向量发生了一些变换，但由于变化程度较小，因此不对句中词向量之间的关联关系进行重新的计算，采用直接复用下层计算好的注意力权重来进行词向量的更新，即“北京站”这个词和其它词的关联关系仍为“你”🡪0.01，“知道”🡪0.02，“去”🡪0.02，“北京站”🡪0.8，“的”🡪0.01，“路”🡪0.02，“怎么”🡪0.01，“走”🡪0.02，“吗”🡪0.01，并没有改变。过程如图4.3所示。

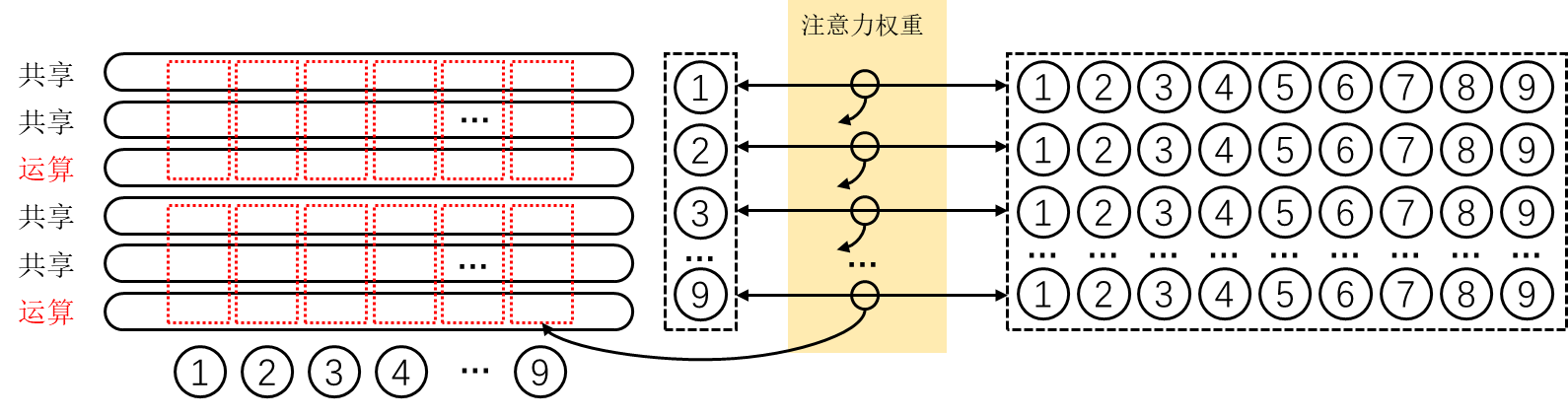


图4.3 注意力权重共享的例子

Fig. 4.3 Example of attention weights sharing

在具体实现中，我们在运算层中和普通模型一样通过计算来获得词汇之间的关联关系，但不同的在于我们会在全局对该值进行记录，使得我们在接下来的共享层中能够直接对相应词汇之间的关联关系进行使用。具体来说在共享层中，我们将不进行的运算，直接利用运算层计算好的权重矩阵和对应位置的value矩阵*V*进行乘法操作，得到更新后的词向量。通过这种方法我们可以有效地减少注意力机制中关联关系这部分的计算，从而降低模型解码过程中的运算量，实现加速的目的。整个过程如图4.4所示。

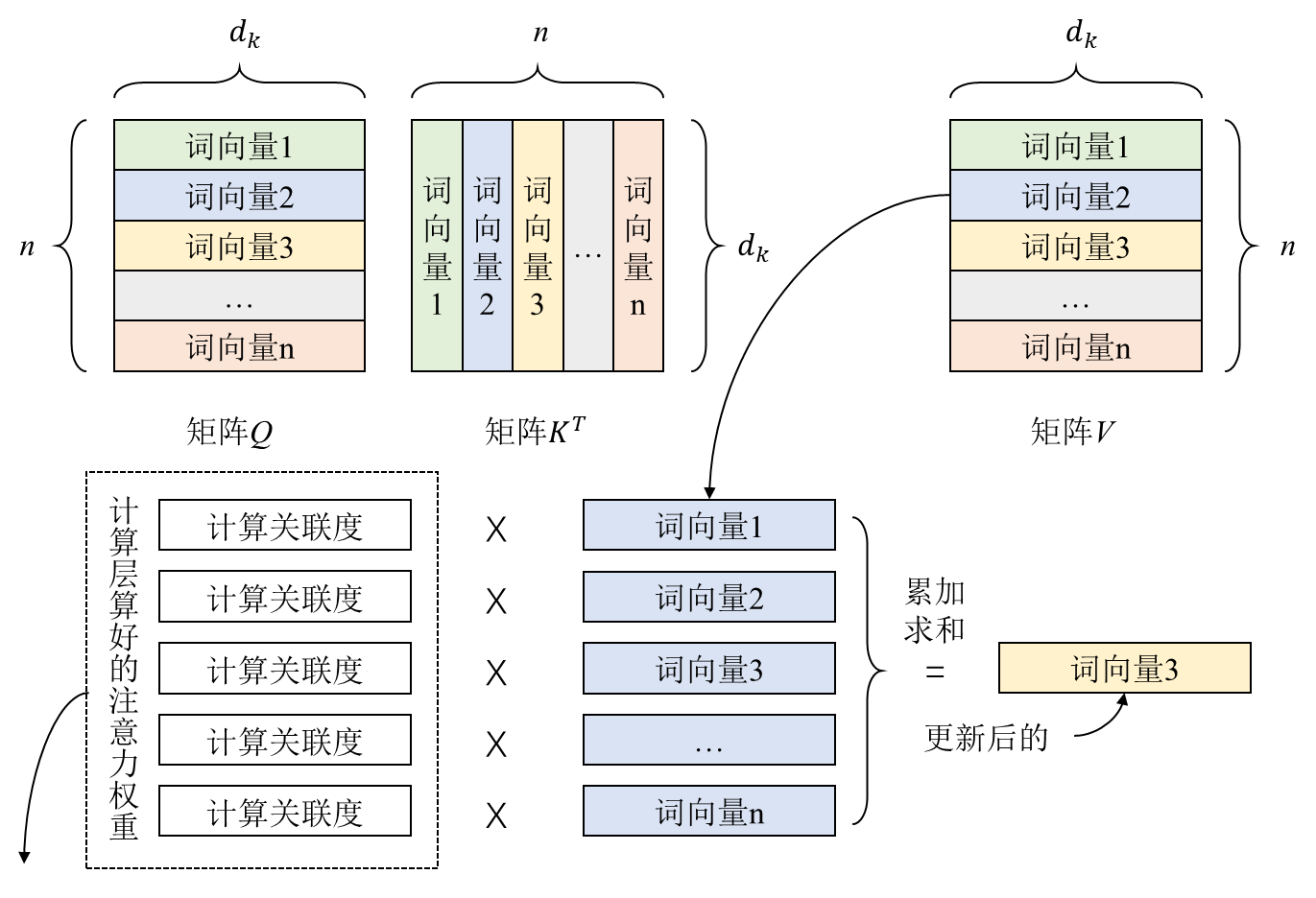


图4.4 注意力权重共享的基本思想

Fig. 4.4 Basic idea of attention weights sharing

本小节将共享注意力权重的方式应用到了基于自注意力机制的神经机器翻译系统中的编码器、解码器内部的自注意力操作上，该方法的使用相当于是将下层运算层中的信息引到当前层中，而对于其它模型中的结构并没有进行修改，如图4.5所示。我们可以从图中看到在模型原本的计算包括编码端的自注意力操作、前馈神经网络操作以及每个子层中的残差、正则化等，解码端有对于现已翻译出的词汇进行的自注意力操作、编码器解码器之间的注意力操作、前馈神经网络操作以及每个子层中的残差连接、正则化等。这些运算单元仍然存在于改进后的模型中，只不过我们在计算自注意力操作的时候，其中的权重来自于下层运算层的计算结果，而非根据现有的词向量进行计算得到的。

当模型中不采用共享注意力权重的方式对源语或目标语进行建模时，自注意力部分的输出计算如公式(4.1)所示：

(4.1)

其中等式左边代表第*i+1*层中自注意力部分的输出（未经残差和正规化），为第*i+1*层中词汇之间的注意力权重，该值的具体计算公式如式(4.2)所示。

(4.2)

式中代表第*i*层的输出，将式(4.1)和(4.2)结合后可得到第*i+1*层中自注意力部分的输出，如式(4.3)所示。

(4.3)

当使用了本小节提出的自注意力权重共享的方法后，在自注意力权重的计算中仅是注意力权重的值而已，改其中的为，*m*为计算层和共享层之间相邻的层数（当*m*为1时代表计算层和共享层相邻），最终得到的自注意力部分输出如式(4.4)所示。

(4.4)

通过对比(4.3)和(4.4)两个公式我们可以看出，使用了共享注意力权重的方法后结果的变化只和和之间的差异性相关。由前文分析我们可知，当计算层和共享层之间距离较近时其输出结果变化并不大，因此采用了共享注意力权重的方式后对性能的影响也较小。

这种方法虽然在使用了下层的权重，但由于在注意力操作计算的时候value矩阵*V*的值是得到更新了的，因此不会出现完全复用下层自注意力操作结果的情况出现，因此保证了模型中多层结构的有效性。

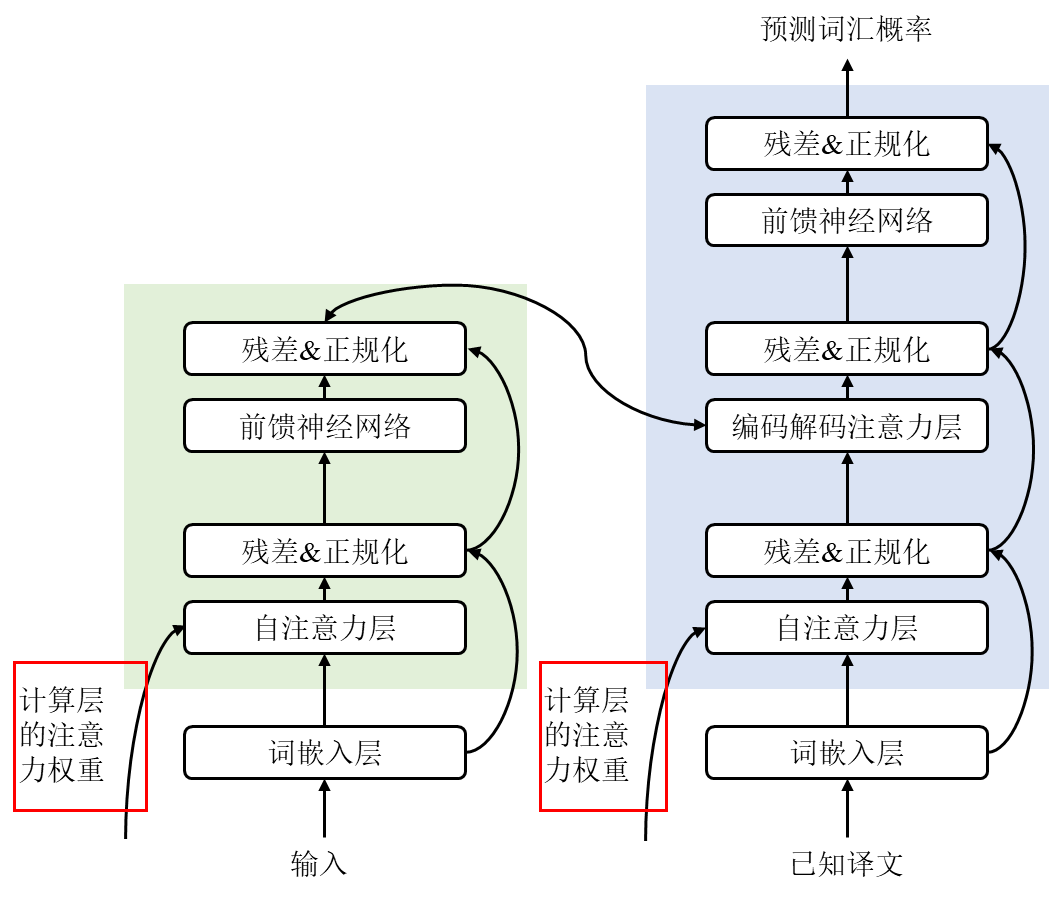


图4.5 自注意力权重共享对网络结构的改变

Fig. 4.5 Network structure changes by self-attention weights sharing

4.2.2 面向编码解码注意力部分的权重共享机制

和上一小节类似，我们可以将注意力权重共享的方式应用到编码器解码器之间的注意力机制中来。编码器解码器之间的注意力操作实在解码器这一端存在的，该结构的主要目的在于通过现有已经翻译出的译文词汇中包含的信息来推断下一个待预测的词汇和源语端（编码端）哪些词汇序列更相关，这样做可以使得我们可以更有针对性地从源语中捕获信息来对当前待翻译的词汇进行翻译。

基于共享的编码器解码器注意力权重共享的思路在于我们可以利用之前层中计算出的解码端词汇和编码端词汇之间的对应权重作为当前层中的注意力权重，这种方法中进行的假设是认为解码端的自注意力得到的词向量信息在经过较少层的传递之后并不会发生特别大的改变，因此它们和源语端序列的对应关系一般来说变化也不会特别大。在这种情况下，我们将下层计算层中计算好的权重作为本层中的权重来计算的话，对性能的影响将不会特别大，尤其是当计算层和共享层之间距离较近的时候。整个基于共享编码解码部分的注意力权重的机器翻译模型框架如图4.6所示。

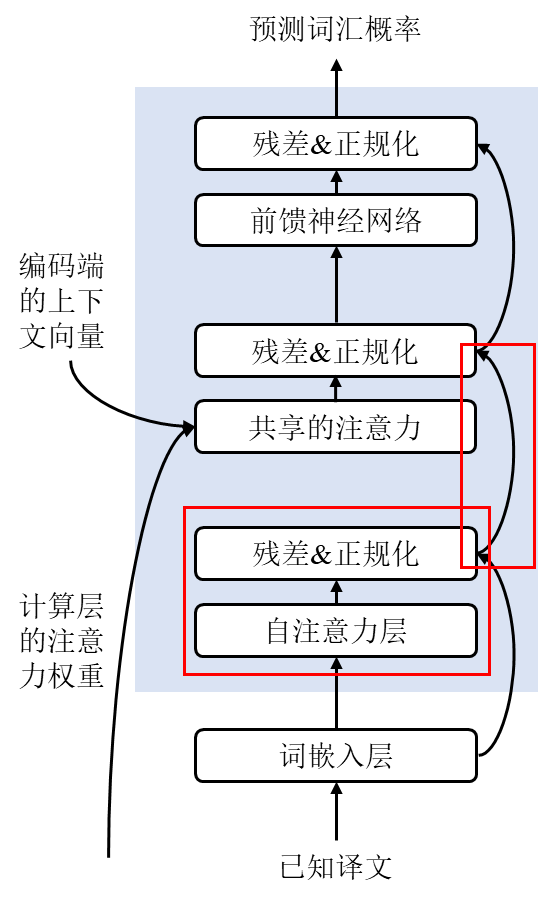


图4.6 编码解码注意力权重共享对解码端网络结构的改变

Fig. 4.6 Decoder structure changes by encoder-decoder-attention weights sharing

我们从图4.6中可以看出在编码解码器之间的注意力机制中进行权重的共享可以有效地复用下层计算层计算出的注意力权重来达到减少注意力操作次数的目的。类似于模型中的自注意力操作，在编码器解码器之间的注意力操作里，注意力部分的输出可参照公式(4.1)计算得出。其中第*i+1*层中注意力权重的具体计算如式(4.5)所示。

(4.5)

式中为第*i+1*层中的自注意力操作的输出（经过残差和正规化），为编码器端顶层的输出。第*i+1*层中注意力部分的输出如式(4.6)所示。

(4.6)

当对模型中的编码器解码器注意力权重进行共享后，该值直接取到了下层计算层中得到的权重作为当前的概率值，改其中的为，最终得到的第*i+1*层的注意力部分输出如式(4.7)所示。

(4.7)

从公式(4.7)中可以看出，这种简单的将下层计算层中的编码器解码器注意力权重拿到上层中使用的方式存在这一定的问题，共享后的注意力输出和下层计算层的输出结果完全一致，这将导致计算层和共享层之间的计算所得信息无法通过原有的方式有效地传递上来。从图4.6中我们也能看到，其蓝色部分代表一层内部的结构，我们可以看到当使用了共享机制后，下层计算层中计算好的信息当传递到共享层的自注意力层之后（包括对应的残差和正则化的部分），并没有有效地继续传递上去，仅仅通过残差的方式传递了部分信息上去，因此会导致层和层之间信息传递的断连。我们可以和传统的未使用共享机制的方式进行对比，共享的模式去掉了原本自注意力部分计算出的结果到编解码注意力模块之间的连接，因此信息传递的量相对少了很多。

而在编码器、解码器内部所使用的共享机制则没有这个问题，虽然同样使用了下层计算层中的注意力权重进行本层的计算，但是由于得到的权重是乘到下层送入的词向量上的（即value矩阵*V*是下层的输入），因此信息在传递的时候并没有因此受到损失。

针对编码器解码器之间的注意力共享中存在的下层信息传递不上去的问题，我们提出了两种解决方案。第一种为将计算出的注意力结果和本层中的自注意力结果进行连接后，通过线性变换得到当前编码器解码器之间的注意力结果；第二种同样为向本层中计算出的注意力结果中融入自注意力操作的信息，只不过该信息的计算方式和第一种方式有所不同，该方法将计算层的自注意力结果和本层中的自注意力结果做差值得到的向量作为信息加入到编解码之间的注意力结果上，这种方法的目的在于认为我们直接将下层的注意力权重拿到本层来用会导致编码解码之间的注意力结果和下层的完全一致，因此我们需要通过本层和下层计算层之间词向量的不同来对共享上来的注意力结果进行修改，因此才有了这里的作差操作。

首先我们对第一种方法进行介绍，在这种方法中，我们最基本的实现为直接将本层中自注意力的结果与计算层中共享上来的注意力权重计算结果连接到一起，然后通过一个前馈网络的线性变换将矩阵的维度压缩回原来的大小，再接下去进行残差以及正规化的操作。整个过程的实现方式较为简单，如图4.7所示。若第*i+1*层为共享层，那么这种方式的得到的输出为如式(4.8)所示。

(4.8)

其中的计算方式如(4.7)所示。

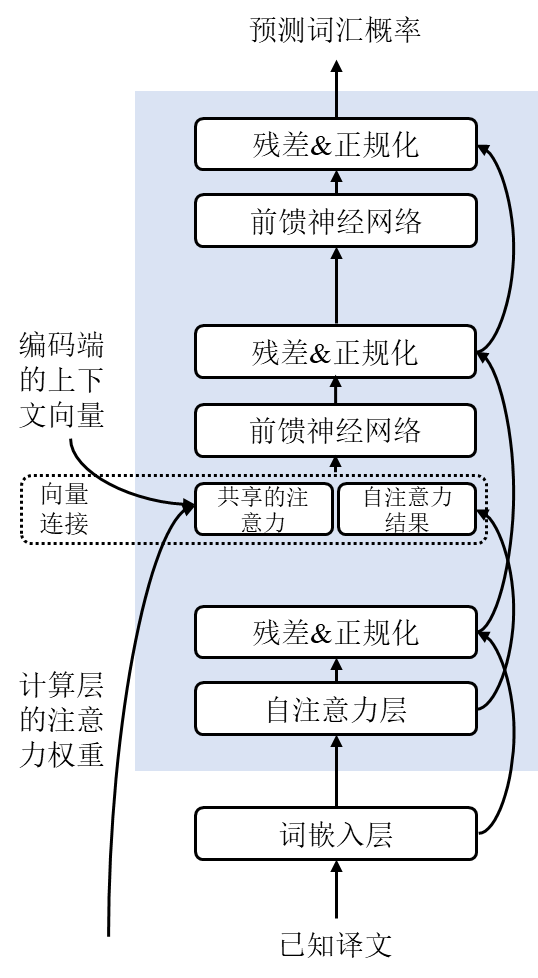


图4.7 基于连接的编码解码注意力共享方式

Fig. 4.7 Concatenate based encoder-decoder-attention sharing mode for decoding process

在上述实现方法中我们直接将本层中的自注意力操作得到的上下文向量和利用下层共享上来的注意力权重计算出的上下文向量进行连接的操作，并没有额外的对这部分信息进行处理，在这个基础上我们还尝试了其它的方式对这些信息进行整合，旨在找到一种性能更好的方式进行权重的贡献。我们主要尝试了两种信息转换的方式，它们分别为增加线性变换的连接方式和使用残差信息的数据传输方式。这两者分别从转换方式和数据来源的角度对模型进行改变，其结构如图4.8所示。

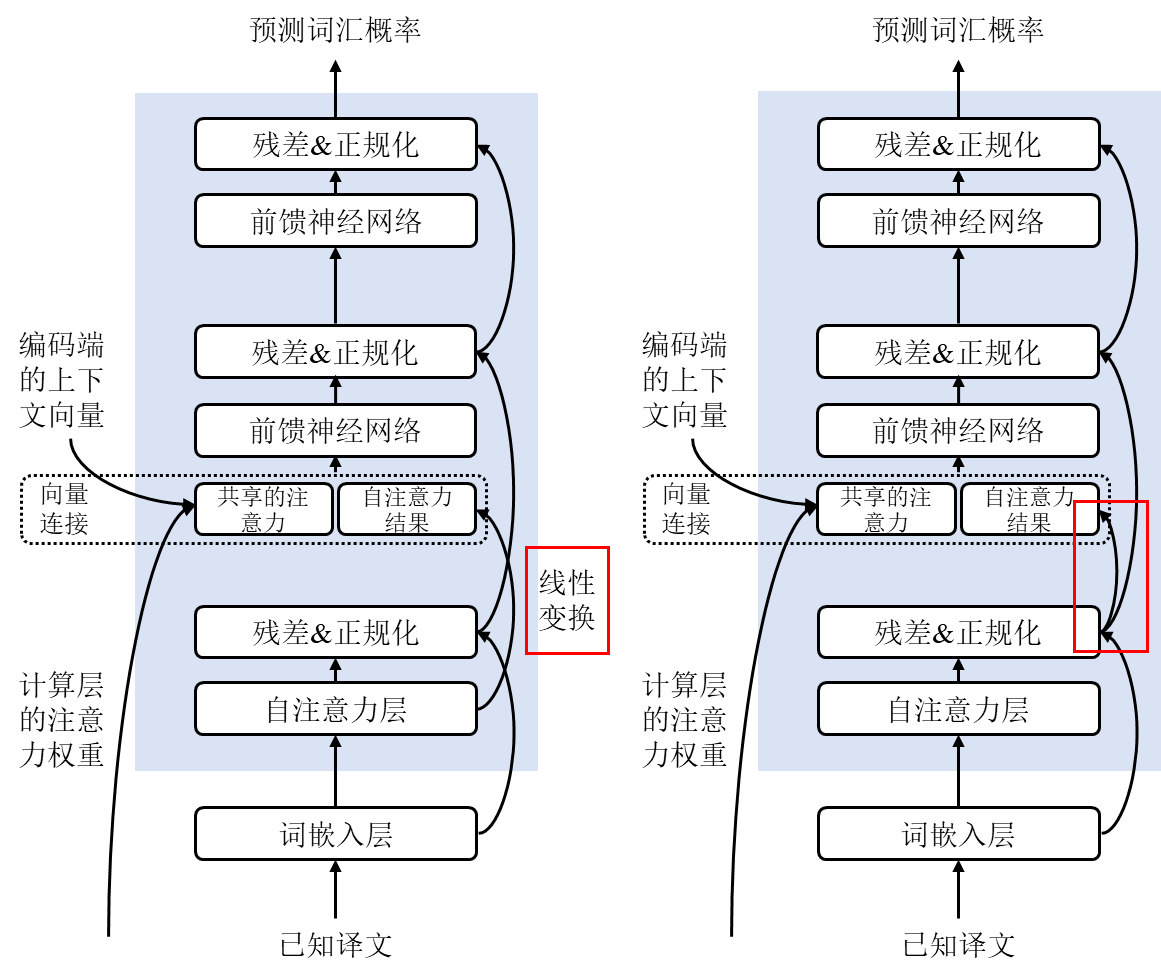


图4.8 基于连接的编码解码注意力共享方式的两个变种

Fig. 4.8 Two modes of concatenate based encoder-decoder-attention sharing for decoding process

图4.8中左侧部分为基于连接的编码解码注意力共享方式的一种变种，该方法在原本连接方式的基础上在自注意力结果进行连接之前增添了线性变换的步骤，通过该方法对自注意力操作得到的信息进行了转换，使得其中所包含的信息能够更适合于共享的编码解码注意力结果进行连接。若第*i+1*层为共享层，那么这种方式的得到的输出为如式(4.9)所示。

(4.9)

图4.8中右侧部分的方式改变了连接操作的信息来源，原本的方式里我们将自注意力操作本身的结果与下层计算层中的注意力权重计算得到的结果进行级联，这里新的方式采用自注意力操作的结果同共享的结果进行级联。那么这种方式的得到的第*i+1*层（共享层）输出为如式(4.10)所示。

(4.10)

其中为第*i+1*层中经过了残差和正规化的自注意力输出结果。

除了将本层中自注意力的结果与计算层中共享上来的注意力权重计算结果连接到一起的方法外，我们还尝试了将计算层的自注意力结果和本层中的自注意力结果做差值得到的向量作为信息与编解码之间的注意力结果进行级联的方式，这种方法的思想在于利用计算层和共享层之间自注意力结果的差异性来对共享得到的编码解码结果进行一定的调整，使得共享得到的编码解码注意力结果和真实的更相近，在简化模型计算量的同时保证性能。其模型的结构如图4.9所示，该方式的得到的输出为如式(4.11)所示。

(4.11)

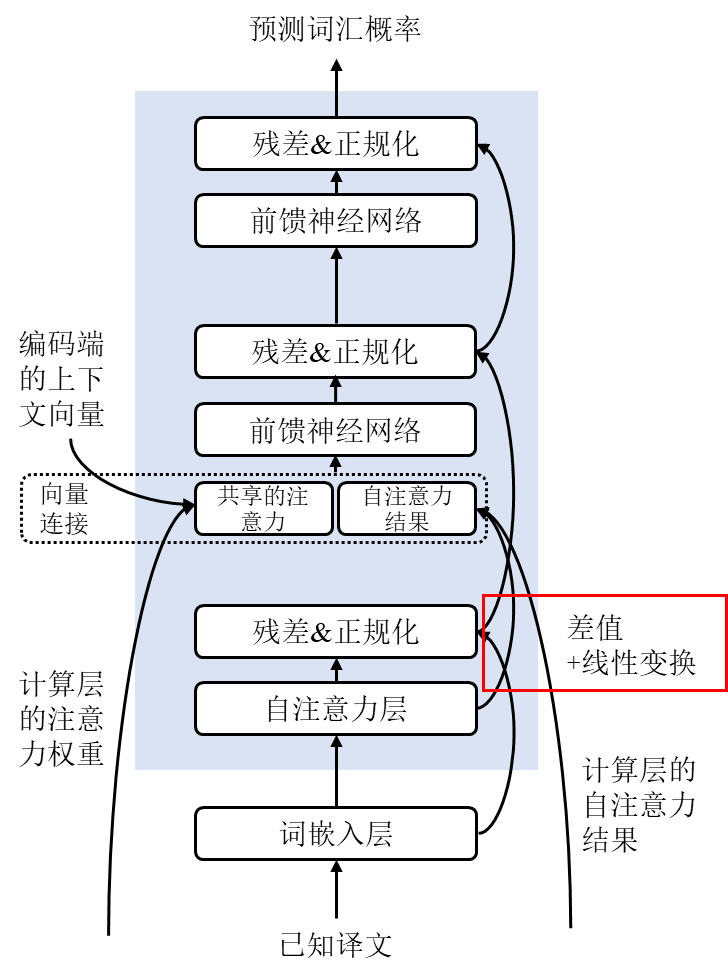


图4.9 基于自注意力差值连接的编码解码注意力共享方式的两个变种

Fig. 4.9 Self-Attention contexts subtract concatenation based encoder-decoder-attention sharing for decoding

4.3 实验

4.3.1 实验设置

对本章中所提出的基于共享权重的注意力方法的实验验证主要是在中英方向的任务中进行，评价方式使用目前机器翻译自动评价方法中的BLEU值对结果进行验证，脚本采用开源的测试工具multi-bleu.perl。

在该翻译任务中，训练数据来自中英方向的LDC数据（LDC2000T46、LDC2000T47、LDC2000T50、LDC2003E14、LDC2005T10、LDC2002E18、LDC2007T19、LDC2004T08），其中包括3,900万左右中文词和4,500万英文词。开发集使用NIST2004对模型中的超参数进行调整，测试集采用NIST2005、NIST2006和NIST2008对翻译效果进行评价。整个训练以及测试的数据均通过BPE算法对分词后的数据进行子词切分，其中BPE过程中的merge次数为32000，最终得到的源语端（中文）词表大小为43211，目标语端（英语）词表为31586。

表4.1 服务器配置

Table 4.1 Server configuration

|  |  |
| --- | --- |
| **组件** | **配置** |
| 中央处理器 | Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz |
| 内存 | 128GB |
| 图形处理单元 | NVIDIA 1080TI |
| 显存 | 11GB |
| 硬盘 | 3TB |

表4.2 模型参数配置

Table 4.2 Model Parameter Configuration

|  |  |
| --- | --- |
| **参数** | **值** |
| 隐层大小 | 512 |
| batch大小 | 8192 |
| 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 |
| 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 | 65000 |

在实现方面，有关于模型的训练，我们同样在开源的基于自注意力机制的神经机器翻译系统中Tensor2Tensor（T2T）中进行二次开发，使用的代码版本为1.0.14。由于该版本中的机器翻译系统在解码的过程中效率较低，因此我们仅使用Tensor2Tensor开源系统进行翻译模型的训练，然后使用实验室自研的解码器对源语言序列进行翻译，过程中记录采用了基于共享注意权重方法的自注意力翻译模型和原本的模型在性能和速度上的差别。实验中我们所使用的模型参数集为Tensor2Tensor中的transformer\_base\_single\_gpu。该集合中具体的参数如表4.2所示。

关于实验设备我们采用单个NVIDIA 1080TI对系统进行训练和解码，通过GPU设备在矩阵运算方面的优势提升系统的训练和解码效率，设备的具体信息如表4.1所示。其中更新次数为65000，相当于训练数据重复10轮左右。使用的模型采用了checkpoint ensemble的方法，集成了15个最近保存的模型参数以提升实验结果的稳定性。

实验中我们在进行解码实验的时候所使用的参数为：batch大小16，beam大小6。

4.3.2 实验结果与分析

4.3.2.1 模型性能

本章中提出了一种通过共享注意力权重的方式对模型进行加速的方法，关于使用该方法后对性能的影响本小节将从两个角度进行实验。

首先是对比不同共享层数的实验，该实验在6层的基于自注意力机制的神经机器翻译系统中进行实验，主要包括三种共享的方式：每2层共享一次（第一层计算，第二层共享；第三层计算，第四层共享；第五层计算，第六层共享）、每3层共享一次（第一层计算，第二层、第三层共享；第四层计算，第五层、第六层共享）以及每6层共享一次（第一层计算，第二层、第三层、第四层、第五层、第六层共享）。实验中主要针对基于自注意力机制的神经机器翻译系统中解码器端的自注意力机制部分进行共享处理，实验结果如表4.3所示。

表4.3 不同共享层数对性能的影响

Table 4.3 Effect of different shared layers on performance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **系统** | **每几层计算一次** | **开发集** | **测试集** | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 1 | 52.34 | 46.05 | 46.9 | 39.37 | 44.11 |
| baseline+每2层计算一次 | 2 | 52.17 | 45.8 | 46.65 | 38.91 | 43.79 |
| baseline+每3层计算一次 | 3 | 52.01 | 45.85 | 46.48 | 39.1 | 43.81 |
| baseline+每6层计算一次 | 6 | 51.43 | 45.54 | 46.47 | 38.97 | 43.66 |

根据上表中的结果，我们可以看到，当采用每2层或每3层计算一次注意力权重的时候，共享的方式相对基线系统大约有0.3个BLEU值的下降，而当每6层计算一次注意力权重的时候（相当于只在第一层中计算权重，在上面的每一层中直接用底层计算好的权重进行计算），性能损失大约为0.45个BLEU值。在翻译系统中BLEU值下降0.3以上在用户将可能感受到译文质量的变差，因此在后续的实验中我们均使用性能可接受的每3层计算一次的方式进行试验。

此外，关于性能方面我们还对上一节中提出的编码解码注意力机制中不同的共享方式对性能的影响进行了实验，如表4.4所示。表中“baseline”代表直接将下层计算层中编码解码器之间计算好的注意力权重用在共享层中的结果，“baseline+连接”代表将根据共享上来的注意力权重计算好的上下文向量和本层中的自注意力结果进行连接后的方法；“baseline+连接+线性变换”为在“baseline+连接”的基础上在使用本层自注意力机制之前先对其结果进行线性变换的方法；“baseline+连接+残差”代表我们在使用一个本层的信息和编码解码注意力的上下文向量进行连接的时候，使用自注意力机制通过残差和正则化后的结果的方法；“baseline+连接+差值”指的是我们利用计算层中的自注意力结果和本层的自注意力结果进行差值后与共享上来的注意力结果进行连接的方法。

表4.4 不同共享方法对性能的影响

Table 4.4 Effect of different shared methods on performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **系统** | **开发集** | **测试集** | | | |
| **MT04** | **MT05** | **MT06** | **MT08** | **平均** |
| baseline | 51.82 | 45.37 | 46.3 | 38.73 | 43.47 |
| baseline+连接 | 51.89 | 45.86 | 46.64 | 38.7 | 43.73 |
| baseline+连接+线性变换 | 52.03 | 45.92 | 46.23 | 38.98 | 43.71 |
| baseline+连接+残差 | 51.6 | 44.75 | 45.79 | 38.29 | 42.94 |
| baseline+连接+差值 | 51.96 | 45.73 | 46.13 | 37.77 | 43.21 |

从上表中的实验结果中我们可以看出，使用普通的方式对编码器解码器之间的注意力权重进行共享将对性能产生较大影响，性能上大约会有0.64个BLEU值的下降，而当使用了连接操作将本层中的自注意力信息引入进来之后，性能有了较大程度的提升。当我们将共享层中的自注意力结果作为一部分和共享的编码解码结果进行连接之后性能上相对不进行共享的方式下降了0.38-0.4左右，其中增加了线性变换的方式在测试集上和不含线性变换的方式相差不多，但是在开发集上的性能有着0.14个BLEU点的提升。除此之外我们在实验中测试的“baseline+连接+残差”和“baseline+连接+差值”的方式在性能上并没有得到预期的提升，反而不如简单地线性变换的方式，这一点问题需要在后续研究中进行解决。

4.3.2.2 模型速度

关于在基于自注意力机制的神经机器翻译系统中引入权重共享的机制之后，我们分别在自注意力权重的共享和编码器解码器之间注意力机制的共享上进行实验，使用了相对来说对性能影响较小的共享层数（每3层计算一次，在6层的网络结构中共计算两次权重，共享4层权重）对加速效果进行检验。实验结果如表4.5所示。

表4.5 权重共享方式对解码速度的影响

Table 4.5 Effect of sharing Attention weights on speed of decoding precess

|  |  |  |
| --- | --- | --- |
| **系统** | **解码器端部分时间消耗** | |
| **自注意力机制** | **编码解码注意力机制** |
| original | 17696.92 | 16347.75 |
| original+共享 | 11207.45 | 9971.67 |

从表中我们可以看到采用了贡献权重的方式对模型进行加速能够使得在损失很小性能的前提下减少相应的注意力计算耗时约40%。虽然我们通过使用共享的方式可以有效地减少在注意力操作中的权重计算次数，但根据前文所分析出的注意力机制里的时间复杂度占比，另外一部分耗时集中在权重矩阵和value矩阵之间的运算上（占整个注意力机制计算中的比约为50%），因此这部分时间并不会因为我们采用了共享的方式计算注意力结果而得到改变。

4.4 本章小结

本章主要介绍了三部分内容：基于共享权重的注意力机制研究动机、具体的方法描述和实验分析。

首先我们介绍了基于共享权重的注意力机制的合理性，由于在基于自注意力机制的神经机器翻译系统中，相临近的若干层之间词向量变化不大并不大，因此词汇之间的注意力权重也相互接近，因此我们可以通过共享权重的方式对模型计算量进行简化，以达到我们加速解码速度的目的。之后我们分别从自注意力权重共享以及编码器解码器之间注意力权重共享的角度出发，提出两种解决方案。对于编码器解码器之间注意力权重共享的问题，我们发现简单地将下层的权重提供给共享层用会导致信息传递上的减少，因此我们通过若干种方式对这部分的共享结构进行改进，以达到以最小的性能损失为代价实现加速的目的。最后我们通过相应的实验对注意力权重共享的方式进行了验证，并对结果进行了相应的分析。

第5章 总结与展望

5.1 工作总结

随着深度学习技术的发展，机器翻译技术的性能获得了较大程度的提升，在编码解码框架下，由最初的基于循环神经网络的模型结构，逐步发展到基于卷积神经网络的结构和基于自注意力机制的结构，后两种方法利用模型本身的特性，使得网络能够讲不同位置词汇之间的信息传输距离拉近，能够更加有效地对语言序列中的信息进行提取。本文中主要围绕如何对基于自注意力机制的神经机器翻译系统解码过程进行加速进行介绍，旨在不损失或以较小性能损失为代价降低翻译系统在解码过程中的时间消耗，提升模型在实际中的可用程度。

通过分析得知，对于基于自注意力机制的神经机器翻译系统，由于舍弃了传统的网络结构（如循环神经网络和卷积神经网络等），因此在解码的过程中时间消耗主要集中在注意力操作的运算上。针对该问题，本文中提出两种面向注意力计算的解码加速方法，分别为基于coarse-to-fine的注意力机制和基于共享权重的注意力机制。两种方法分别从提高注意力操作运算速度和减少注意力操作执行次数的角度出发对翻译模型进行改进，最终达到解码加速的目的。

首先，基于coarse-to-fine的注意力机制的核心思想在于降低注意力机制中权重计算部分的时间消耗，主要通过将计算注意力权重前的词向量先进行维度上的压缩后再进行相关度的计算，有效压缩了词向量表示中的冗余空间，减少了其在实际中的计算量，加速了系统的运行速度。关于其中对于词向量冗余空间的压缩，我们提出了两种具体的解决方案，第一种为基于coarse-to-fine的注意力机制中的简单实现，该方法为每一层中的词向量提供相同的削减率，而第二种方法我们则将不同层叠加后词向量中信息的增长考虑进去了，简单来说就是随着层数的越来越高，每个词向量中所要表示的信息量也越来越大，因此需要使用的表示空间也相应变多起来，这时我们在进行维度压缩的时候就要为高层的词向量提供一个相对较高的维度保留率，不对其进行过分的压缩，而对于底层一些的层，我们则可以使用一个较高的削减率，使得模型整体的计算保持一个较低的状态，同时性能损失较小。通过实验我们发现，将基于coarse-to-fine的注意力方式应用到翻译系统中来当保留率为0.5及以上几乎不影响模型性能。另外编码器解码器之间的注意力机制相对其内部的自注意力机制包含更多需要表达的信息量，我们需要给它们一个相对较高的保留率以减少coarse-to-fine方式对性能的影响。

此外，我们还提出一种基于共享权重的注意力机制，这种方法的核心思想在于有效利用相邻的若干层之间注意力权重的相似性来对权重进行共享，最终达到减少注意力权重计算次数的目的。根据基于自注意力机制的神经机器翻译系统结构我们可以得知，其主要是通过词向量信息的不断更新达到对源语和目标语建模的目的，这种方法使得在临近的层之间词向量信息的变化幅度其实并不大，因此使得根据它们所计算出的关联程度也比较接近，因此我们这种权重共享的方式就利用了网络模型中的这个特点，在网络中的某一层（称之为“计算层”）对权重进行计算，然后在接下来的一层或多层中直接使用这个计算好的关联度进行注意力机制结果的运算，节省了计算注意力权重的时间。这种方式我们分别在翻译模型中的自注意力机制和编码解码机制中进行应用，针对编码器解码器之间注意力计算的特殊性，我们对普通的共享方式进行了相应改良，尝试了基于连接本层自注意力上下文向量在内的四种方法，并通过实验对方法的有效性进行了验证，发现在基于连接的方式中，对自注意力结果进行了线性变换后再与编码器解码器之间的注意力上下文向量进行级联可以得到最好的结果。

本文中提出了两种方法对基于自注意力的神经机器翻译系统解码过程进行加速，分别从降低注意力操作的耗时以及操作次数的角度出发，最终使得翻译系统能够在保持性能变化不大的前提下减少解码过程中的耗时，提升模型的执行效率。

5.2 创新点分析

本文中最大的创新点在于创新性地提出从词向量信息累积的角度对基于自注意力的神经机器翻译系统进行解释，该思想认为在该模型结构中，源语和目标语端分别通过词向量相互之间关联度的计算得到和它们所表达的语义最接近的词语，然后按照关联程度收集它们的词向量来对自身的信息表示进行更新，得到更新后的词向量。通过使用这种方式对结构进行分析，我们提出了基于coarse-to-fine的方式以及权重共享的方式对注意力结构进行改进，分别通过降低词向量中的冗余空间和利用相邻层之间词向量的相似性对注意力操作中的权重计算部分进行修改，使得模型本身的计算量大大减少，提升在解码过程中不必要的时间消耗。

5.3 未来工作

本文中提出了的两种注意力机制的修改方式，但即使使用了这些方式之后注意力机制在基于自注意力的神经机器翻译系统中时间占比仍旧较大，因此我们对于解码加速的空间还有很多

5.3.1 基于局部信息的注意力机制

基于coarse-to-fine的注意力机制的出发点在于使用更少的信息进行注意力权重的计算，那么加速的角度则可以从如何减少注意力操作中的操作对象出发来对模型解码过程进行加速。其核心思想在于在进行编码器、解码器内部的自注意力操作以及二者之间的注意力操作的时候，我们可以仅一个局部窗口内的词汇，与它们进行关联程度的运算，而非对序列中每个词进行操作。这样可以有效减少注意力操作中的运算矩阵大小，实现解码加速的目的，而与此同时如果可以有效地确定关联窗口的位置，那么该方法对于翻译系统性能的影响也不至于太大。

5.3.2 其他注意力权重共享的方式

在基于共享权重的注意力机制中，我们针对编码器解码器之间的注意力权重计算过程提出了四种方法，其中通过实验验证有效的为根据将下层计算层得到的注意力权重计算出的注意力结果和本层中的自注意力上下文向量进行连接后作为编码器解码器之间的注意力上下文向量。其中提出的将下层计算层中的自注意力上下文向量和本层的自注意力部分的上下文向量作差方法并没有起到一个正向的作用，但其存在的问题尚且没有得到合理的解释，与此同时，我们目前在编解码器之间的注意力权重共享中最好的结果仍然会使得模型性能下降较多，是否能够找到一种更加有效的结构使得这部分在当前计算量的情况下有更好的性能还值得我们进一步研究。

参考文献

1. 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.
2. 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.
3. Makoto Nagao (1989). Machine Translation: How Far Can it Go? Oxford University Press, Oxford, U.K. Translated by Norman Cook.
4. Nagao M. A framework of a mechanical translation between Japanese and English by analogy principle[J]. Artificial and human intelligence, 1984: 351-354.
5. Brown R D, Hutchinson R, Bennett P N, et al. Reducing boundary friction using translation-fragment overlap[J]. 2003.
6. Kim J D, Brown R D, Jansen P J, et al. ‘Symmetric Probabilistic Alignment for Example-Based Translation[J]. 2005.
7. Doi T, Yamamoto H, Sumita E. Graph-based retrieval for example-based machine translation using edit-distance[C]//MT Summit X second workshop on example-based machine translation, Phuket, Thailand. 2005: 51-58.
8. 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.
9. Somers H. Example-based machine translation[J]. Machine Translation, 1999, 14(2): 113-157.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 肖桐. 树到树统计机器翻译优化学习及解码方法研究[D]. 东北大学, 2012.
19. Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//Advances in neural information processing systems. 2014: 3104-3112.
20. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.
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. Paulus R, Xiong C, Socher R. A deep reinforced model for abstractive summarization[J]. arXiv preprint arXiv:1705.04304, 2017.
25. Graves A. Generating sequences with recurrent neural networks[J]. arXiv preprint arXiv:1308.0850, 2013.
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. Gehring J, Auli M, Grangier D, et al. Convolutional sequence to sequence learning[J]. arXiv preprint arXiv:1705.03122, 2017.
30. 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.
31. Bradbury J, Merity S, Xiong C, et al. Quasi-recurrent neural networks[J]. arXiv preprint arXiv:1611.01576, 2016.
32. Kalchbrenner N, Espeholt L, Simonyan K, et al. Neural Machine Translation in Linear Time[J]. 2016.
33. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in Neural Information Processing Systems. 2017: 6000-6010.
34. Kim Y, Denton C, Hoang L, et al. Structured attention networks[J]. arXiv preprint arXiv:1702.00887, 2017.
35. Parikh A P, Täckström O, Das D, et al. A decomposable attention model for natural language inference[J]. arXiv preprint arXiv:1606.01933, 2016.
36. Lin Z, Feng M, Santos C N, et al. A structured self-attentive sentence embedding[J]. arXiv preprint arXiv:1703.03130, 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. 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.
40. 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.
41. Ba J L, Kiros J R, Hinton G E. Layer normalization[J]. arXiv preprint arXiv:1607.06450, 2016.
42. Britz D, Goldie A, Luong T, et al. Massive exploration of neural machine translation architectures[J]. arXiv preprint arXiv:1703.03906, 2017.
43. Sennrich R, Haddow B, Birch A. Neural machine translation of rare words with subword units[J]. arXiv preprint arXiv:1508.07909, 2015.
44. 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.

致谢

在此毕业之际，向实验室的所有老师、同学们表示衷心的感谢。

首先我想感谢我的导师肖桐老师，您可以说是我人生路上十分重要的一个人。您缜密的思维逻辑，认真的治学态度值得每个人学习。您告诉了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。感谢肖桐老师这两年多的照顾和认真指导，您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，从您身上学到了很多为学为人的道理，真的非常感谢您！

然后，我要感激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。

还有我要感谢实验室的其他老师，感谢你们日复一日的辛苦付出，才让实验室有了家一样的感觉。去年的时候我不小心腿受伤了，走不了路，张俐老师特地给我带了好用的药膏，韩蕴霞老师借我拐杖还给我送了她亲手做的饭菜，我真的特别感动。我只是实验室普通的一个学生，老师们却用百倍的关心照顾我们，就像对待自己的孩子一样，还是那句话，感谢实验室的各位老师，谢谢。

还有感谢实验室的各位同学，感谢你们在学习和生活上给我的帮助。特别要感谢王强学长、杜权学长在我遇到问题时耐心的指导，感谢各位学长、学姐、学弟、学妹们在我遇到问题的时候的帮助。因为有你们，实验室才越来越好，谢谢大家。

最后还要感谢我的父母，感谢你们给我自由的空间去发展，也谢谢你们一路上的体谅与关怀。读研之后因为工作学习的压力回家的机会越来越少了，真的十分愧疚，希望在之后的学习生活中可以更好地分配自己的时间，常回家看看。谢谢小的时候你们陪我长大，现在也轮到我陪你们变老了。

硕士期间参加的科研项目

科研项目：

1. 面向多层次篇章语义的机器翻译理论、方法与实现。

硕士期间取得的学术成果

学术成果：

1. 于2018年7月在《中文信息学报》上以第一作者发表《基于数据并行的神经语言模型多卡训练分析》论文一篇。