上海大学

SHANGHAI UNIVERSITY

毕业设计（论文）

UNDERGRADUATE PROJECT (THESIS)

装订线

题目：基于预训练语言模型的自动文本摘要

学　　院：悉尼工商学院

专　　业：信息管理与信息系统

学　　号：16124320

学生姓名：钟支俊

指导教师：陈娟

起讫日期：2020.3.2-2020.6.12

目　　录

摘要

在信息爆炸的互联网时代，随着各类媒体平台的兴起与发展，人们的日常生活中充斥着庞大的信息，令人目不暇接，如何从海量信息中筛选找到自己需求的信息也愈加困难。在日益增长的需求与深度学习技术的兴起下，自动摘要技术研究也备受瞩目。文本自动摘要技术旨在通过输入文本信息，自动生成输出文章的简短摘要。该技术的研究，可以帮助人们高效快速地从文本中获取关键信息，降低信息搜索抽取所需的时间，有效提高信息获取效率。目前大多文本摘要技术生成的摘要冗余重复、词不达意，针对该问题，本文提出了基于预训练语言模型的文本自动摘要方法，并开展了一系列研究工作。

本文首先综述了抽取式摘要、生成式摘要和预训练语言模型的国内外研究现状及发展，并详细介绍了本文所用到的长短期记忆网路、注意力机制、先验知识和集束搜索等深度学习方法相关理论。论述了如何运用深度学习技术构建序列到序列的基准模型，以实现生成式的文本摘要。随后介绍了本文所用到的 LCSTS 中文短文本数据集和 Rouge 评价指标。在实验一采用融合注意力机制的 Seq2Seq 模型得到初步的结果，相比数据集论文提出的基准结果，本文的初步结果在 Rouge-1 和 Rouge-L 指标上都有一定提升。

本文随后论述了 Bert 预训练语言模型的机制及其技术原理，提出了结合 Bert 模型的文本摘要创新方法并构建了相应的模型和融合注意力机制的 Seq2Seq 框架，在编码端采用双层 BiLSTM 网络，使用双层单向 LSTM 网络作为解码端。通过 Bert 模型提取文本和摘要的词向量特征后，将提取的特征输入到对应的编码端和解码端训练学习，得出的实验结果在 Rouge 三项指标的测试分数上都优于之前的实验，训练时间也大幅减少。对比实验结果之后，说明了本文提出的基于预训练语言模型的自动文本摘要方法能够有效提高生成摘要的质量和可读性，并减少模型的训练时间和成本，有助于帮助人们提高阅读效率，节约时间精力。

关键词：深度学习；文本摘要；注意力机制；Seq2seq；Bert

ABSTRACT

In the Internet era of information explosion, with the rise and development of various media platforms, individuals daily live is filled with endless information. How to find the information demanded from the mass information is gradually challenging. With the increasing need and the development of deep learning technology, research on automatic summarization has attracted much attention. Automatic summarization technology aims to automatically output a short summary by input the article. Research on text summarization technology can help people obtain key information from text efficiently and quickly, reduce the time cost of information acquisition, and effectively improve reading efficiency. At present, most of the abstracts generated by text summarization technology are redundant and repetitive. In response to this problem, this paper proposes a novel automatic text summarization method based on a pre-trained language model, and carries out a series of research work.

The prior research of extractive summarization, generative summarization, and pre-trained models are first stated in this article. The details of the deep learning methods used in this paper, such as LSTM networks, attention mechanisms, prior knowledge, and beam search theory are also introduced. The methods using deep learning technology to build a sequence-to-sequence benchmark model to achieve a generative summarization is described. The LCSTS Chinese short text dataset applied in this article and Rouge metrics method are also interpreted. In the first experiment, the Seq2Seq model was used to obtain preliminary results. Compared with the baseline results proposed in the dataset paper, the preliminary results of this paper have improved on the Rouge-1 and Rouge-L metrics scores.

Subsequently, pre-trained model Bert and its technical principle are discussed. An innovative text summarization method combining Bert is proposed, and the corresponding Seq2Seq framework is constructed. A two-layer BiLSTM network is used at the Encoder and the LSTM network serves as the Decoder. After extracting the word vector features of the text and summary through the Bert, the extracted features are feed to the corresponding Encoder and Decoder for training. The experimental results obtained greatly exceed the previous experiments on the Rouge-1, Rouge-2 and Rouge-L scores, and the training time greatly reduce. It indicates the automatic text summarization method with the pre-trained language model can effectively improve the quality and readability of the generated summary, and reduce the training time and cost, which will effectively help people improve reading efficiency and save time.

Keywords: Deep learning; Text summarization; Attention mechanism; Seq2seq; Bert

第一章 绪论

伴随着互联网的发展和各类社交媒体的兴起，围绕在人们生活中的信息日益膨胀，人们有限的时间精力难以应对层出不穷的新闻、评论、文章等信息，进而对信息筛选汇总的需求也逐渐升高。自动文本摘要技术在此背景下应运而生，吸引了诸多学者的研究关注并得到快速发展。本章节主要讲述文本摘要技术的研究背景和意义，并综述国内外相关研究现状及发展趋势，最后介绍本文的主要研究内容、创新点及难点并叙述文章的整体结构安排。

第一节 研究背景及意义

当今世界科学技术发展迅速，诸多科技产品、信息化服务的问世也不断地丰富着人们的生活。信息化时代的到来，给人们的日常生活带来了巨大便利，同时也使得人们越来越依赖互联网获取自己所需的信息。随着互联网上的信息呈现爆炸式的快速增长，尤其是身处于大数据时代之中，真假信息的参杂更加令人眼花缭乱，获取所需的信息往往需要消耗人们更多的的精力和时间。如何高效地从海量信息中检索并提取有用信息也逐渐成为了一个备受瞩目的技术问题[1]。

自动文本摘要技术旨在不改变文本原意的情况下，利用算法模型自动地总结出文本的主要内容[2]。在如今信息泛滥的大数据时代，该技术能够以简明的摘要来传达信息的主要内容，协助解决生活中信息爆炸的问题，帮助人们节约大量的时间及精力。自动文本摘要技术的应用场景十分广泛，可以应用于商品评论总结、新闻标题生成、论文摘要生成和搜索结果扼要生成等各类场景[3-4]。在生活节奏日益加快的今天，人们对信息的收集概括需求逐渐升高，自动文本摘要的相关研究也因此备受关注，成为当下热门的话题。

当前自动文本摘要技术主要有两种实现方法，一种是抽取式摘要[5]，旨在将原文中已存在的重要句子抽取出来拼凑在一起作为摘要;另一种是生成式摘要[6]，是在理解文章主旨含义的基础上，使用自然语言技术生成新的句子作为摘要。抽取式摘要的特点是实现难度低、摘要句的生成过程简单，但摘要句可能出现上下文不匹配的问题[7];而生成式摘要虽然实现难度高，但其生成摘要句的过程更加拟人化，生成的摘要自然、质量高、连贯通顺[8]。

实现文本自动摘要的方法多种多样，但深度学习是目前研究此问题效果最好的方法，它有效降低了对人工的依赖，可以高效地进行语料训练，通过与多种神经网络结构、模型的结合，生成高可读性和高准确度的摘要[9]。然而，深度学习训练往往需要大量人工标注的数据集，又因为复杂的神经网络结构的引入，执行速度慢，实验需要花费相对较长的时间，对计算机性能有一定的要求。在自然语言处理研究领域的背景下，预训练技术旨在使用海量无标注的文本语料来训练语言模型，再将预训练好的模型参数应用到后续其他的特定任务上[10]。基于深度学习方法，将预训练语言模型应用到自然语言处理的下游任务上，可以大幅提升任务效果，同时缩短训练时间。因此，基

于预训练语言模型的自动文本摘要，不仅可以使模型收敛更快，缩短模型训练时间，同时还可以避免训练过程中陷入局部最优点，提升模型性能。

结合以上分析，针对自动文本摘要问题，本文将主要研究实现生成式文本摘要，并采用深度学习方法构建算法模型，实现中文短文本的自动摘要；同时结合预训练语言模型的特性，将其扩展应用到文本摘要问题，最后研究实现改进的算法模型，以提高生成的摘要质量。

第二节　国内外研究现状及发展趋势

自动文本摘要生成是一项充满挑战性的任务，围绕该问题，前后众多学者付诸行动，研究开发了一系列模型方法，该技术也因此得以快速发展并逐渐运用到实际生活中。自动摘要研究最早兴起于国外，也因此形成了以英文为主的研究趋势并开发了相应的数据集和模型方法；国内相关研究虽起步较晚，但学者们厚积薄发，不仅在英文领域占据了一席之地，也开拓了中文摘要研究领域并贡献出了诸多成果。针对自动文本摘要问题，本节主要从抽取式摘要、生成式摘要和预训练语言模型三方面综述国内外研究现状并探讨今后的发展趋势。

一、 抽取式摘要研究

自动文本摘要的概念最早由 Luhn[11]提出。在上世纪 50 年代，Luhn 利用文本的词频特征来计算文本中句子的权重并对其评分，进而将评分最高的句子作为摘要。尽管该方法的提出在当时非常超前，但其实现还是相对简单，获取的摘要冗余较大，同时也容易忽略一些比较重要的低频词信息。

上世纪末，Kupiec 等[12]在文本摘要研究领域中首次引用统计机器学习的方法，他们选取了主题词、句子长度、段落等特征，并使用朴素贝叶斯分类[13-14]方法训练得到一个分类器以给句子进行权重评分，再依据句子的得分高低依次抽取前若干个句子组成为文本的摘要。该方法可以通过选择不同特征，挖掘出文本中潜在的深层信息，但其摘要效果严格依赖于训练数据质量的好坏，实现方法采用监督或半监督方式，执行速度较慢。

在本世纪初，Mihalcea 和 Tarau 根据谷歌浏览器评价网页重要性排名的算法 PageRank[15]，提出了计算文本中句子重要性的方法 TextRank[16]。TextRank 算法优点在于无监督，尽管实现方式较为简单，但也获取了较高质量的摘要。然而在考虑语义方面的信息上，该算法仍需进一步改进。

2011 年，受到图排序算法在搜索、推荐等场景的广泛应用的启发，Erkan 和 Radev 提出了基于图论的 LexRank 方法[17]。通过计算句子之间的相似度对文本和词汇进行分类，并根据相似程度为每个句子权重评分，最终根据评分选择分数较高的句子作为文本的摘要。尽管该方法在计算句子权重的同时可以充分考虑到句子间的全局联系，但它忽略了文本篇章结构以及句子上下文的信息，相似度计算的好坏也直接决定了句子重要性排序的正确与否，需要进一步考虑改进。

针对现有的摘要方法无法全面考虑影响因素，摘要提取冗余，不能很好反应文本主题等不足，李娜娜[18]提出了基于 TextRank 的改进算法。在构建文本的 TextRank 网络图时,考虑文本的篇章框架结构和文章的上下文信息等因素,并将这些因素加以数字化处理和调整,结合改进的 TextRank 算法以生成文本摘要的候选句，最终得到的摘要结果冗余度低、概括性高且连贯流畅。然而，该方法主要针对的是新闻文本，面对其他类型的文本特征，需要重新调整设计算法，因此在泛用性方面，仍需进一步研究改进。

考虑到句子间的共现词汇，耿焕同等[19]在节点关系图的基础上提出了一种基于词共现图的自动文本摘要方法，并在词共现矩阵的构造中引用条件概率的概念。该方法通过连接不同主体之间的特征信息和词共现图形成的信息自动提取文本摘要，取得了不错的实验效果。

通过研究 LDA 在有监督学习研究领域中单文本自动文摘中的作用，吴晓峰等[20]提出了一种基于 LDA 的 CRF 自动文本摘要方法，将 LDA 提取的主题信息作为特征加入条件随机场模型中进行训练，并分析研究在不同主题下 LDA 对摘要结果的影响。其实验结果表明，对比以传统特征为输入的条件随机场文摘系统，引入 LDA 特征的方法能够有效提高生成摘要的质量。但是该方法并没有考虑到 LDA 模型对文本切割的作用，需要在文本初步切割的基础上进一步实验对比生成摘要的效果。

二、 生成式摘要研究

在生成式摘要研究中，Facebook 公司的 Rush[21]等首次引入深度学习技术方法，基于序列到序列（Sequence to Sequence，Seq2seq）模型[22]，采用卷积神经网络（Convolutional neural networks, CNN）[23]编码原文信息，同时引用注意力机制以生成摘要。该方法在 DUC-2004 数据集上表现优异，标志着运用深度学习解决自动文本摘要问题的可行性。

Chopra 等[24]使用同样的编码方式对原文进行编码，但采用循环神经网络(Recurrent neural network, RNN) [25]来生成摘要，该方法大大提高了摘要效果。他们的分析表明舍弃卷积神经网络而采用循环神经网络处理序列化信息会在一定程度上提升摘要质量。

针对深度神经网络并没有充分挖掘到文章的全部特征，IBM 公司的 Nallapati[26]等采用循环神经网络对原文进行编码，同时利用词特征、停用词和文档结构等信息生成摘要，其效果远超单纯使用深度神经网络的结果。该实验表明针对研究的问题将相应的特征加入到研究范围内，能极大地提升模型表现。

2017 年，考虑到 RNN 相关算法只能按左右顺序依次计算，谷歌团队提出了 Transformer 模型[27]，该模型摒弃了传统的 CNN 和 RNN 模型，仅仅采用注意力机制构建模型算法，有效缓解了传统模型中存在的长期依赖问题，同时提高了模型的并行能力。

Romain 等[28]结合强化学习进行训练，以解决文本摘要模型中存在的曝光

误差问题。同时对输入的每个词保留历史时刻的权重信息，抑制在不同的解码时刻关注到的相同部分的输入，使生成的内容信息覆盖全文，利用内解码器注意力机制，引入先前时刻的解码信息，来避免生成重复的内容。

由于中文社交媒体发布的信息大多是短文本，篇幅相对较短、存在较多噪声，针对该问题传统的文本摘要方法效果较差。为此，Hu 等[29]从新浪微博采集获取超过 200 万条的短文本新闻及摘要构成 LCSTS(Large Scale Chinese Short Text Summarization Dataset) 数据集，解决了中文文本摘要研究领域中大规模语料数据缺少的现象，同时提出了采用 RNN 生成摘要的方法，并在该数据集上实验得出了基准结果，以供后续研究比较。

针对当前生成式文本摘要方法存在的语义信息利用不充分、摘要精度不足等问题,丁等[30]提出一种基于双编码器的文本摘要方法。通过双编码器为序列到序列模型提供更丰富的语义信息,并对融入双通道语义的注意力机制和伴随经验分布的解码器进行了优化研究。所提方法优化了传统序列映射和词特征表示,即增强了模型对语义的理解,也提高了摘要的质量。但是，该模型在处理具有特殊名称和陌生名词文本时性能还有待提高。

施[31]提出构建了融合复制机制和覆盖机制的单字典自动文本模型,该模型采用堆叠的双向长短期记忆网络 BiLSTM[32]进行信息抽取,以提高模型理解语义的能力。模型融合复制和覆盖机制,增加文本摘要的连贯性和可读性,减少文本摘要的未录入词和词语重复问题。同时将集成学习应用到实验中，由于不同编码器理解语义不同,采用不同编码器进行模型训练,通过对多个模型进行集成,增加模型理解语义的多样性,提高实验预测结果的准确性和模型的泛化能力。然而，该模型训练特征单一，表达语义不够清楚，在模型泛化能力上有仍很大的提升空间。

三、 预训练语言模型研究

预训练技术最早应用于计算机视觉领域，大量实验证明使用预训练技术可以大幅提升下游任务的效果[33]。更重要的是，在深度学习时代，充分使用预训练模型能够极大地改善下游任务模型对标注数据数量的要求，可以有效处理一些难以获得大量标注数据的新问题。

针对 Word2Vec[34-35]、Glove[36]等静态的预训练技术在处理一词多义问题上的不足，Peters[37]等提出了 ELMo 模型。通过使用针对语言模型训练好的双向 LSTM 来构建文本表示，由此捕捉上下文相关的词义信息，因而可以更好地处理一词多义问题，也让学者们注意到基于大规模语料集的预训练语言模型的优异之处。

由于 Transformer 在处理长期依赖性方面比 LSTM 有更好的表现，Radford[38]等提出了采用 Transformer 进行特征抽取的 GPT 模型，并使用生成式方法来训练，这也是首次将 Transformer 应用于预训练语言模型。同时在微调阶段引入语言模型辅助目标,以解决微调过程中的信息遗忘。虽然 GPT 模型在多项自然语言处理领域达到了很好的效果，但其本质上仍然为单向自回归语言模型，对语义信息的建模能力有限无法获取上下文相关的特征表示。

Devlin[39]等在 2018 年提出了 BERT 模型，同样通过堆叠 Transformer 子结构来构建基础模型，但不同的是采用 Transformer 进行双向建模。同时结合多种特征抽取机制，引入掩码语言模型（Masked Language Model，MLM)预训练目标，以获取上下文相关的双向特征表示。尽管 BERT 模型在多项自然语言处理任务上创造了历史最佳记录，但其预训练过程和生成过程的不一致，导致在生成任务上效果不佳，无法处理文档级别的 NLP 任务，只适合于句子和段落级别的任务。

由于 ELMO、GPT 等预训练模型都是基于传统的语言模型，即自回归语言模型，自回归语言模型天然适合处理生成任务，但是其无法表征双向上下文信息，因此学者也开始转向自编码思想的研究。为此，Google 团队的 Yang[40]等提出一个框架 XLNet 模型来连接语言建模方法和预训练方法。XLNet 仍使用自回归语言模型，为解决双向上下文的问题，引入了排列语言模型，采取随机采样语言排列，只预测单个句子后面的词，以缓解计算量过大的问题，同时融合 Transformer-XL 的优点，以协助处理过长文本。

百度团队的 Sun[41]等提出基于知识增强的 ERNIE1.0 模型，通过建模大规模数据中的实体概念等先验语义知识，学习语义知识单元的完整语义表示。不同于 BERT 对字进行随机掩码，ERNIE 通过掩码词和实体概念等完整语义单元来训练 MLM，从而增强模型的语义表示能力。

由于语言表征的预训练过程和知识表征过程有很大的不同，会产生两个独立的向量空间。为解决上述问题，Zhang[42]等认为知识图谱中的多信息实体可以作为外部知识改善语言表征，并提出 ERNIE(THU)模型。基于 BERT 预训练原生模型，通过对齐文本中出现的实体和外部的知识图谱，再将知识嵌入得到实体向量作为 ERNIE 的输入；同时在有实体输入的位置，结合实体向量和文本表示的非线性变换，以融合词汇、句法和知识信息。

针对 ERNIE 1.0 模型在捕捉训练语料中词法、语法、语义等深层潜在信息上的不足，百度团队的 Sun 等[43]提出可持续学习的语义理解框架 ERNIE2.0，在预训练阶段构建多个层次的任务以全面学习文本深层的潜在知识。通过交替学习这些不同种类的任务，对模型不断训练更新，这种连续交替的学习范式使模型不会忘记之前学到的语言知识，持续提升模型效果。

Zhang 等[44]探讨了利用预训练语言模型提高文本自动摘要效果的可行性。基于编码器-解码器框架，给定输入序列以两阶段的方式生成输出序列。编码器采用 BERT 模型将输入序列编码为上下文语义表示，在解码器端，首先使用基于 Transformer 的解码器来生成输出序列的草稿；之后掩住草稿中的每个单词并将其提供给 BERT，然后基于 BERT 生成的输入序列和草稿的上下文语义表示，由一个基于 Transformer 的解码器来预测精化每个被掩盖住位置的单词。

针对传统词向量在自动文本摘要过程中因无法对多义词进行有效表征而降低文本摘要准确度和可读性的问题，岳等[45]提出了一种基于 BERT 的自动文本摘要模型构建方法。该方法引入 BERT 预训练语言模型用于增强词向量的语义表示，将生成的词向量输入 Seq2Seq 模型中进行训练并构建自动文本

摘要模型,实现对文本摘要的快速生成。但仍存在不足之处，比如由标准摘要转换为向量时会产生信息的损耗，导致句子不通顺、指代不明等问题。

第三节　本文创新点及难点

一、 创新点

针对自动文本摘要问题，以往的学者们已经提出实践了诸多方法及模型，现有的研究方法也仍在不断创新改进，以优化自动摘要的质量。本文以前人研究为基盘，结合现有的模型算法，提出了基于预训练语言模型的自动文本摘要生成的创新方法，其中本文创新点如下：

1.数据集的选择和词向量的训练，目前中文文本摘要语料库数量相对较少，而深度学习算法对语料库的规模依赖程度高，为此选用了 LCSTS 数据集，包含 200 多万条短新闻文本摘要。

2.目前，将预训练语言模型用于文本摘要生成任务上的研究较少。本文通过预训练模型提取词向量特征用于后续摘要生成任务，大幅缩短模型训练时间。

3.运用 BERT 获取文本上下文相关的双向特征表示，增强模型的语义表达能力，优化文本摘要效果。

4.通过比较使用预训练前后的模型，对实验结果进行分析改进，探讨了利用预训练语言模型提高文本自动摘要效果的可行性。

二、 难点

尽管文本摘要的研究日益成熟，针对各种问题的解决方法手段也很有效，但基于预训练语言模型的自动文本摘要研究，仍面临着诸多挑战，需要不断的优化改进。而本文的研究难点主要体现在以下几点：

1.基于预训练语言模型的文本摘要生成实现有一定难度，由于 BERT 模型本身在预训练过程和生成过程的不一致，并没有做生成任务的相应机制，不能直接应用于生成任务。

2.将 BERT 用于 Seq2Seq 的文本摘要生成任务，需要分别预训练编码器和解码器，但是编码器-注意力-解码器结构没有被联合训练，需要针对该问题进一步改进模型。

3.为保证模型摘要生成的质量，需要训练大规模数据集，耗费大量时间，实验周期长。

第四节　论文的研究内容及组织结构

一、 主要研究内容

针对文本摘要问题，本文以生成式文本摘要为起点，采用深度学习的方

法，研究了基于 Seq2seq+Attention 模型和预训练语言模型的实现方法。简而言之，本文展开的研究内容主要包括以下几点：

1.介绍文本自动摘要的相关技术原理及实现方法，对本文应用的循环神经网络模型、先验知识、集束搜索和注意力机制等深度学习技术进行详细讲解。

2.介绍序列到序列框架原理，阐述融合注意力机制的序列到序列模型的方法及算法流程，实现基于该模型的自动文本摘要生成，得到初步的实验效果作为基准实验结果。

3.讲解预训练语言模型 Bert 的原理机制，采用 BERT 模型获取文本的上下文特征，将获取的文本特征输入到融合注意力机制的序列到序列模型中以生成摘要。

4.介绍数据集、预处理方法、实验流程和文本摘要的评价指标，通过比较两种模型方法生成的文本摘要质量及对应评分，对实验结果进行分析比较，探究预训练模型在文本摘要生成领域的可行性和有效性。

二、 论文的组织结构

围绕基于预训练语言模型的自动文本摘要生成这一研究主题，本文展开了一系列相关工作，其中本文技术路线图如图 1-1 所示：

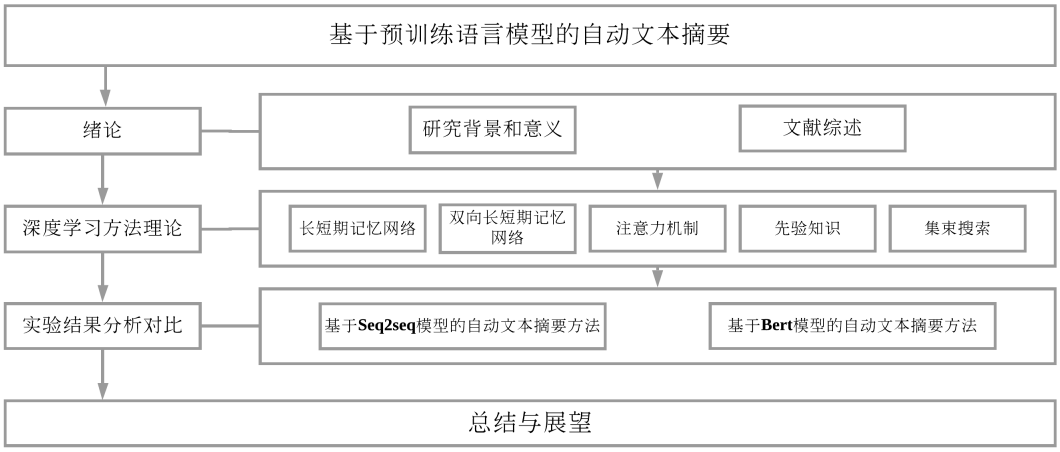

图 1-1 论文技术路线图

围绕上述技术路线图，本文内容主要分为六个篇章结构，每个章节的内容概括如下：

1、绪论，主要阐述文本自动摘要的研究背景和意义，同时总结综述国内外研究现状及未来发展趋势，说明本文研究的创新点及难点，最后概括了本文开展的研究内容及章节结构安排；

2、方法理论介绍，叙述采用深度学习实现文本自动摘要的方法，介绍了深度学习相关技术，阐明本文中所采用的长短期记忆网络，双向长短期记忆网络，注意力机制，先验知识和集束搜索等深度学习相关方法的机制原理；

3、实现基于 Seq2seq+Attention 模型的自动文本摘要生成，介绍序列到序列框架，叙述融合注意力机制的 Seq2seq 模型的原理和机制；

4、基于预训练语言模型的自动文本摘要生成，介绍预训练语言模型 Bert 的创新之处，叙述采用 Bert 预训练词向量以构建模型的原理及机制；

5、实验比较以及结果分析，介绍了论文中使用的数据集及数据预处理流程，论述了文本摘要的评价指标；针对两种模型方法生成的文本摘要，对实验结果进行分析比较；

6、总结与展望，总结本文的主要研究工作及研究过程遇到的难点和不足之处，然后据此提出优化问题的一些方法设想，进一步规划之后的研究工作。

第二章 方法理论介绍

实现自动文本摘要生成的方法多种多样，各有千秋，但其中应用最为广泛的是深度学习方法。随着计算机性能的不断提高，各种大规模数据的出现以及各类算法模型的问世，深度学习研究也愈加火热，被广泛应用于各个领域。深度学习的模型有很多种，常用的有 CNN[23],RNN[25]，以及深度神经网络（Deep neural network，DNN）[46]。其中 CNN 常用于计算机视觉方面的研究，RNN 则主要用于研究自然语言处理，DNN 则常用于搜索、推荐、广告等大量抽象特征的场景，对于自动文本摘要生成，本文主要使用的则是 RNN 模型。本章将围绕深度学习方法，详细叙述解决文本摘要问题的相关模型及技术方法。

第一节 长短期记忆网络 LSTM

循环神经网络模型 RNN 是一种常用的神经网络结构，它尤其适合处理序列数据，已经成功应用于聊天机器人、语音识别、图像标注、机器翻译等众多时序问题。然而，RNN 中存在对输入信息的长期依赖，梯度消失和爆炸等问题。长短期记忆网络[47]（Long short term memory，LSTM）是针对传统循环神经网络的一个改进模型，用于解决其存在的长期依赖问题。与传统的 RNN 相比，LSTM 模型在结构上的独特之处是它精巧地设计了循环体结构。

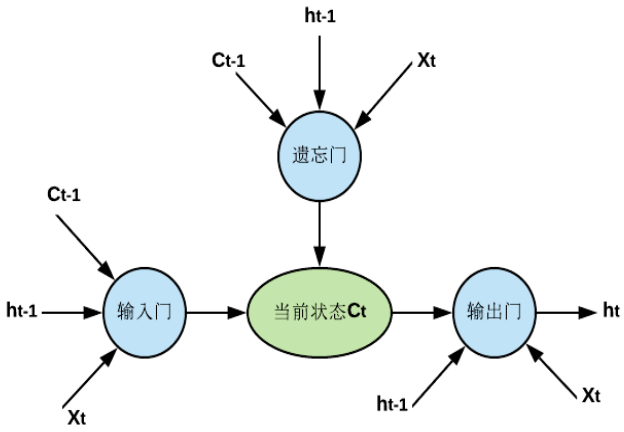

图 2-1 LSTM 的循环体结构

LSTM 的循环体结构如图 2-1 所示，其中 C、h 和 X 分别对应当前状态值、上一单元的输出值和当前时刻的输入值，t 表示网络当前时刻。LSTM 采用遗忘门和输入门两个门来控制当前单元状态的信息内容，其中遗忘门决定了上一时刻的单元状态信息的保留比例；输入门则用于决定当前时刻网络的输入信息的留存比重；最后，结合前面两个门的信息留存比例，通过输出门来控制单元状态输出到当前网络的信息比重[47]。简而言之，LSTM 网络可以通过控制三个门单元来决定信息流入流出的权重，从而根据上下文的改变更新单元参数，即使模型参数固定，输入序列同样可以改变历史的累积信息，以避免训练过程中梯度消失或爆炸的问题。

LSTM 的特殊结构使 LSTM 的学习能力大幅增强且能够有效避免循环神经网络训练过程中的梯度问题，更加容易训练，其在语音识别、机器翻译、图像标题生成等诸多应用都获得了出色表现。

第二节 双向长短期记忆网络 BiLSTM

虽然 LSTM 模型有效解决了 RNN 中存在的长期依赖问题，但是它在时刻t只能考虑之前的历史信息和当前的输入信息。然而，在自然语言处理任务中，对于输出的预测不仅仅需要之前的输入信息，同时需要之后的信息。充分利用前后的上下文信息能够更全面的提取文本整体的特征信息，进一步提高模型的性能和效果。

双向长短期记忆网络[32]（Bidirectional long short term memory，BiLSTM）通过堆叠两个独立的 LSTM 模型，可以同时分析当前输入序列的上下文信息。图 2-2 展示了 BiLSTM 网络的具体结构，其中，A 为 LSTM 中的隐层单元，i 为时刻，X、O 和 h 分别对应输入、输出和隐层单元状态。前向 LSTM 用于接收当前输入的上文信息，后向 LSTM 接收当前输入的下文信息，通过将前向和后向的隐藏层向量拼接在一起，BiLSTM 可以同时做到双向的语义依赖捕捉，显著提升模型的表达能力。

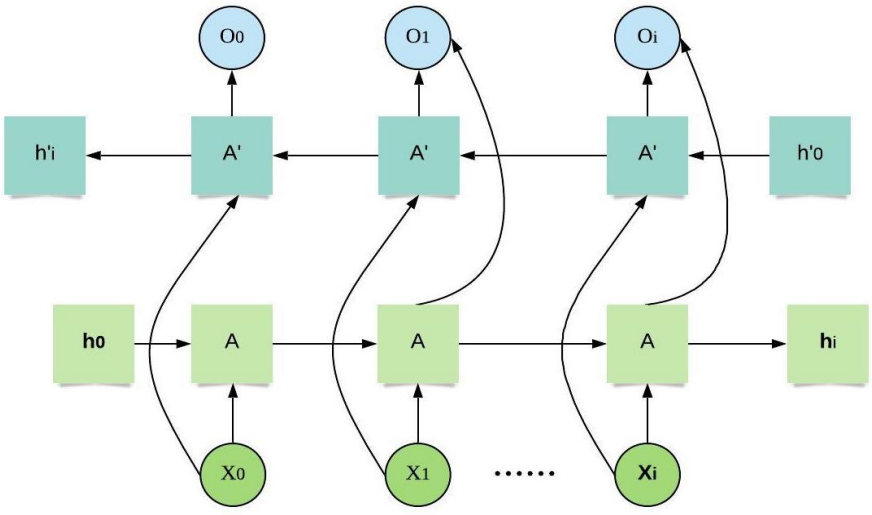

图 2-2 BiLSTM 网络结构示意图

第三节 注意力机制

深度学习中的注意力机制的本质类似于人类视觉的注意力机制，当人们在注视一件东西的时候，其关注点一定是其当前看的这件东西的某一个地方或部位，这意味着人们在注视某个目标或场景时，会利用有限的注意力资源从大量信息中快速筛选并注意到其中的关键信息，采用视觉注意力机制可以极大地提高人类处理视觉信息的效率和准确性。本文所引用的注意力机制，其核心目标也是从众多信息中选择出对当前任务目标中更关键的信息，即在摘要生成过程中模型会更加关注文本中比较关键的词汇和主语等信息。

一、点积注意力网络

近几年注意力机制广泛应用于深度学习各领域，无论是做图像处理、语音识别还是自然语言处理的相关任务中，都可以运用到这一机制。传统的注意力机制，也叫点积注意力网络，其计算公式[27]定义为：

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2-1}$$

其中$Q \in \mathbb{R}^{n\times d_k}, K \in \mathbb{R}^{m\times d_k}, V \in \mathbb{R}^{m\times d_v}$；在公式(2-1)中，d代表向量长度，Q为目标文本句子的当前隐含状态，K为源文本句子对应的用于计算相似度的隐含状态，V则是被加权的隐含状态。不考虑 softmax 激活函数的话，其过程就是$n\times d_k, d_k\times m, m\times d_v$三个矩阵的相乘运算，最后得到结果为$n\times d_v$的矩阵。即注意力层将$n\times d_k$的序列Q编码成了一个新的$n\times d_v$的序列，下式[27]说明了注意力权重的计算公式：

$$Attention(q_t,K,V) = \sum_{s=1}^{m}\frac{1}{Z}\exp\left(\frac{<q_t,k_s>}{\sqrt{d_k}}\right)v_s \tag{2-2}$$

在公式（2-2）中，Z是归一化因子，d_k为向量长度，q_t, k_s, v_s与公式（2-1）对应。首先通过q_t与各个k_s内积计算得到q_t与各个v_s的相似度，在使用激活函数 softmax 将得到的相似度转化为归一化的权重时，如果其权重的分布非常不均匀，那么在反向传播时就很有可能会得到非常高的梯度值。因此，引入$\sqrt{d_k}$作为调节因子，使得内积结果不至于太大。上述的注意力机制便是缩放的点积注意力网络，其结构如下图所示：

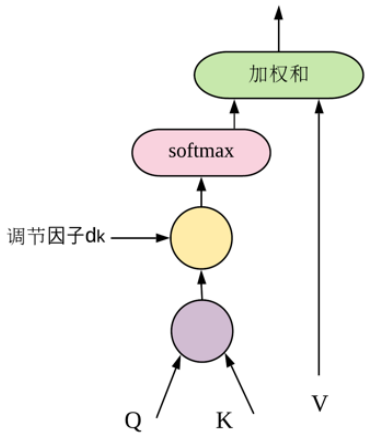

图 2-3 缩放的点积注意力网络

二、多头注意力网络

本文中采用的注意力机制为多头注意力机制(Multi-Head Attention)，不同于传统的点积注意力结构，多头注意力网络通过拼接多个点积注意力网络而得到，具体结构如下图所示：

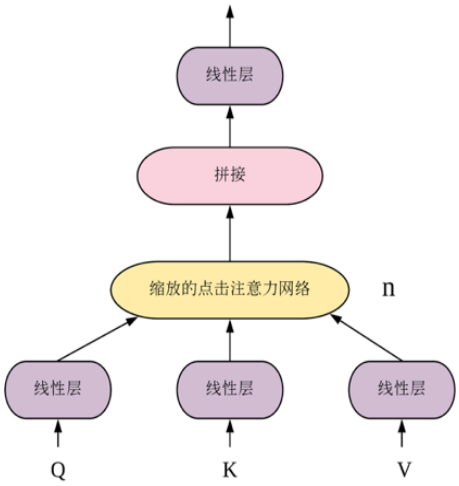

图 2-4 多头注意力机制

多头注意力机制首先通过参数矩阵映射Q,K,V，然后再做缩放点积注意力网络，将此过程重复h次，然后通过拼接得到最终结果。具体计算如公式（2-3）和（2-4）[27]所示：

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{2-3}$$

$$\mathrm{Mu}ltiHead(Q,K,V) = Concat\left(head_1, \ldots, head_h\right)W^O \tag{2-4}$$

其中$W_i^Q \in \mathbb{R}^{d_k \times \tilde{d}_k}, W_i^K \in \mathbb{R}^{d_k \times \tilde{d}_k}, W_i^V \in \mathbb{R}^{d_V \times \tilde{d}_v}$, i 表示注意力头，d 为向量长度，$W_i^Q, W_i^K, W_i^V$分别对应 Q、K、V 的权重矩阵，$W^O$为公式（2-3）计算出来的注意力结果拼接之后生成的最终向量的线性映射参数，最后计算得到一个$n \times (hd_v)$的序列。多头，即重复多次同样的计算，但参数不共享，最后将结

果拼接。多头注意力机制的本质是多个独立的注意力计算，作为一个集成的作用，它可以有效防止模型在训练过程中的过拟合问题。

注意力机制的作用是能够完整捕捉到文本全局的联系，因为它直接将输入序列进行两两比较，虽然相应的其计算量也相对变大，但由于是纯矩阵运算，整体影响也不大。总而言之，在自动文本摘要任务中引入多头注意力机制，可以让模型在关注到输入序列的全局联系的同时，注意到文本中的关键字，并给它们分配更高的权重，使得模型输出的摘要内容更加贴近原文，正确匹配文本的内容及含义。

第四节 先验知识

在文本自动摘要任务中，需要机器像人一样在通读全文、理解文本的语义内容的基础上生成摘要。而按照人的一般思维，总结一篇文章的中心思想内容往往需要联系到原文，这就导致了很多标题的关键字词大多是在文本中出现过的，少部分则在原词意思的基础上生成新的词汇概括。针对此点，在本文中引入了先验知识[48]，因为标题中的大部分字词都在文章中出现过，这样一来，就可以通过文章中的词集构造一个先验分布，并加到字词解码的分类过程中，使得模型在解码输出时更倾向选用文章中出现过的字词。

具体来说，在每一步预测字词时,模型得到总向量X，即编码器的编码向量、解码器当前的隐藏层向量、和当前解码端与编码端的注意力编码三者的拼接，将其接入到全连接层后最终得到一个大小为$|V|$的向量$y=(y_1,y_2,\dots,y_{|V|})$，其中$|V|$是词表的词数。再通过激活函数 softmax 计算得到原本的概率：

$$p_i=\frac{e^{y_i}}{\sum_i e^{y_i}} \tag{2-5}$$

这就是原始的字词预测分类方案。引入先验分布的方案是，对于每篇文章，我们得到一个大小为$|V|$的独热向量$x=(x_1,x_2,\dots,x_{|V|})$，其中$x_i=1$意味着该词在文章中出现过，否则$x_i=0$。将这样的一个独热向量经过一个缩放平移层变换，计算公式为：

$$\hat{y}=s\bigotimes x+t=\left(s_1x_1+t_1,s_2x_2+t_2,\dots,s_{|V|}x_{|V|}+t_{|V|}\right) \tag{2-6}$$

其中s和t为训练参数，然后如下式所示，将这个向量与原来的y取平均后再输入到激活函数 softmax 中：

$$y\leftarrow\frac{y+\hat{y}}{2},p_i=\frac{e^{y_i}}{\sum_i e^{y_i}} \tag{2-7}$$

最后，当模型在解码端将每一步的字预测结果解码时，实际上是结合经过学习训练后的隐藏向量和先验知识而预测出词分布概率得出最终结果的。将先验知识融合进网络中构建的缩放平移层，相对于完全连接网络，其调节

训练的参数是较少的，因此学习更快且泛化性能也更强。先验分布的引入可以让模型加快收敛，生成的标题也更贴近原文本，内容含义表达的也更为具体恰当。

第五节 集束搜索

根据自然语言句子生成的特性，模型需要在每一步中预测出当前最大概率的字符，最终结合所有步找到可能性最大的组合生成句子。所以说模型的解码输出，也是一个建模过程：

$$p(Y|X) = p(Y_1|X)p(Y_2|X,Y_1)p(Y_3|X,Y_1,Y_2)\ldots \quad (2\text{-}8)$$

如公式（2-8）所示，其中Y为需要解码输出的序列，X为输入序列的编码。在解码过程中，模型目标是找到最大概率的序列Y，如果在解码第一步$p(Y_1|X)$时，即预测输出序列的第一个词时，直接选择概率最大的第一个词，得到期望目标词Y_1，然后将其代入到第二步$p(Y_2|X,Y_1)$的解码过程中，再次选择概率最大的词Y_2，依此类推。像这样在每一步解码过程中直接选择概率最大的词输出，就称为贪心搜索[49]。贪心搜索是一种简单且低成本的解码方案，在每步中直接选择最大概率的词输出，直到出现终结符或达到最大句子长度。但是这种方案得到的结果不一定是最优的，假如在第一步的解码过程中选择非最大概率的词Y_1，代入第二步时或许会得到更大的条件概率$p(Y_2|X,Y_1)$，使得两者的乘积会超过逐位取最大的算法。然而要枚举所有路径取最优的结果，计算量将是巨大的。为此本文中采用了集束搜索（Beam Search）方法。

集束搜索算法[50]类似动态规划，但即使在能用动态规划的问题下，它仍比动态规划要简单，该算法可以通过控制集束大小（Beam size）参数来限制在每一步保留下来的可能性词的数量。具体而言，在每步计算时，只保留当前最优的几个候选结果。比如在本文中集束大小取值为 10，那么第一步时，模型只保留使得$p(Y_1|X)$最大的前 10 个Y_1，然后分别代入$p(Y_2|X,Y_1)$，然后各取前 10 个Y_2，这样一来模型就有 100 个组合了，这时模型再计算每一种组合的总概率，然后还是只保留前 10 个，依次递归，直到出现了第一个<end>标识符。集束搜索同样属于贪心算法的范畴，虽然该方法不能保证一定能够找到全局最优解，但因为它考虑到足够大的搜索空间，而采用一个相对的较优解，使得它在解码过程中保留了更多的可能性，对于生成摘要的字词选择空间也更加大。一般来讲，贪心搜索可以认为是集束大小为 1 时的集束搜索特例。而贪心搜索由于每次只考虑当前词的概率，对于预测摘要中组合在一起的词语表现不足，会导致模型最终生成的句子过于冗余，因此采用集束搜索算法会有助于模型选择生成更加连贯且准确的摘要。

第三章 基于注意力机制和序列到序列模型的自动文本摘要生成

近年来，序列到序列模型在自然语言处理领域各任务上都获得了优异的表现，包括机器翻译、问答系统、客服机器人和自动文本摘要生成等。序列

到序列模型在其编码器-解码器的框架结构上，可以将一条不定长度的输入序列转变为长度固定的向量表达，通过训练学习后再将其解码为不定长度的目标序列。自动文本摘要旨在通过输入原文生成一段简短扼要的语句来表达文本的关键信息和整体含义。本章节将在序列到序列框架的基础上引入多头注意力机制[27]构建初步的模型，以实现文本摘要的自动生成，同时将Seq2Seq+Attention 模型作为基准模型得到初步的实验结果，用以对比下章节中使用预训练语言模型生成文本摘要的创新方法。

第一节 序列到序列框架 Seq2seq

序列到序列架构最早用于机器翻译，其中心思想是将可变长的序列映射到另一可变长度的序列，由于输入序列长度和输出序列长度是可变的，这种架构也开始用于自动文本摘要生成领域，也慢慢发展成目前在文本摘要任务中使用最为广泛的方法。Seq2seq[22]模型采用编码器-解码器的框架结构，如图 3-1 所示，其中 Encoder 代表编码端，Decoder 为解码端，C 表示中间状态向量。如下图所示，Encoder 负责读取输入序列并将其编码为固定维度的向量，该固定大小的向量即包含了句子的全部信息。然后，Encoder 通过对输入模型的数据进行学习得到一个中间状态向量 C,再将 C 传递给 Decoder，Decoder 通过对中间状态向量 C 的学习，解码出最终的输出。

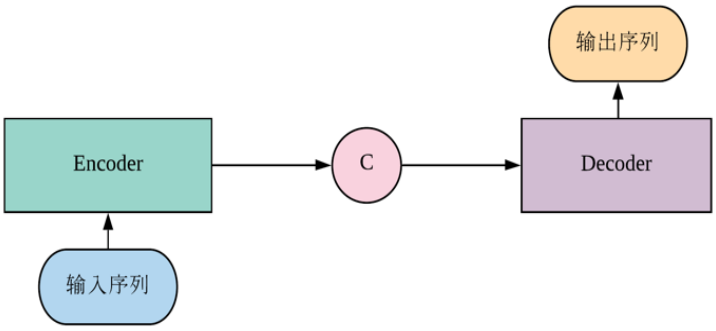

图 3-1 Seq2Seq 框架

由于 Seq2seq 框架非常灵活，Encoder 和 Decoder 可以使用多种神经网络，所以该框架广泛应用于多种领域，不单单是自然语言处理，在语音和图像领域它也有着出色的表现。在自然语言处理领域，编码器和解码器通常使用循环神经网络表示，结合实际应用则经常使用改进的 LSTM、BiLSTM 或者多层 RNN 等网络。针对文本自动摘要研究，在本文中则采用 BiLSTM 网络作为编码器和单向的 LSTM 网络作为解码器。

第二节 Seq2Seq+Attention 模型

由于输入和输出序列的可变性，Seq2Seq 框架也同样适用于自动文本摘要生成研究领域。对于文本摘要来说，输入的序列是一篇文本，输出的序列则为文本对应的标题。假设输入的文本句子为$X = (a, b, c, d, e)$，输出摘要为$Y = (P, Q, R)$，其中每个字母对应一个字符。基于前述的深度学习方法，本文

构建的结合注意力机制的 Seq2Seq 模型结构如下图所示：

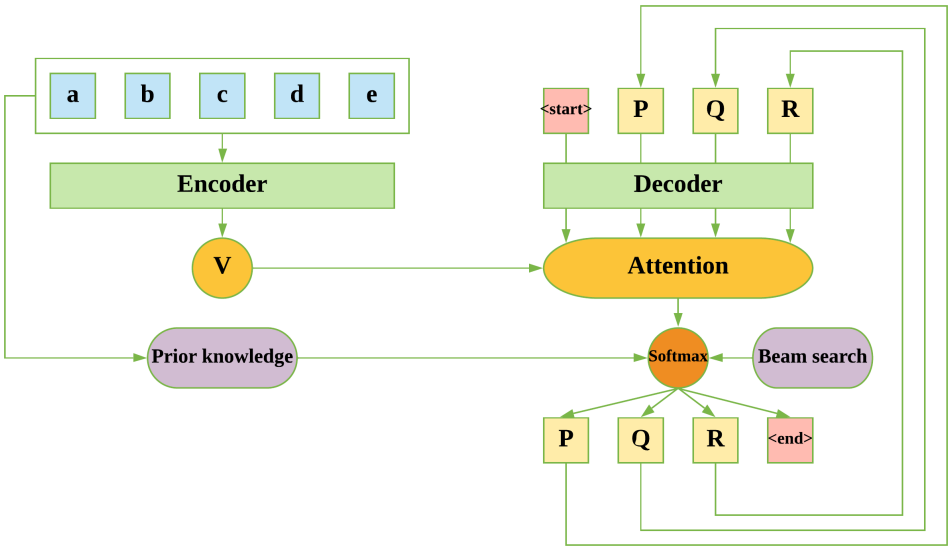

图 3-2 基于 Seq2Seq+Attention 的自动文本摘要生成模型结构

如图 3-2 所示，左边是输入的 Encoder 端，它负责把输入的文本句子编码为一个固定大小的向量；而右边的 Decoder 端则负责将刚才编码出来的向量解码为期望的输出。为了捕捉输入文本的上下文信息和整体特征，本文选择 BiLSTM 网络作为编码端，另一方面，为了解决句子的长期依赖问题，选择 LSTM 作为解码端。因为整个模型解码过程是递归进行的，所以与 Encoder 不同，Decoder 是单向递归的，图上模型具体的运行流程为：

1) 将输入序列$X = (a, b, c, d, e)$输入到 Encoder 编码，同时得到输入序列的先验知识信息；

2) Encoder 通过学习输入序列，得到一个中间状态向量V；

3) 所有输出端，都以一个通用的<start>符号作为序列开头的标识符，以<end>作为序列结尾的标识符，这两个标识符也属于一个字符；

4) 将<start>字符输入到 Decoder 得到隐藏层向量；

5) 解码时，结合 Encoder 编码出来的中间状态向量V，同时使用注意力机制往回查阅原来的每一个字符得到注意力编码；

6) 将中间状态向量V与 Decoder 的输出及注意力编码混合，然后输入到 Softmax 分类器；

7) Softmax 结合输入序列的先验知识和集束搜索方法预测输出分类结果P；

8) 将P输入 Decoder，得到新的隐藏层向量，再次结合 Encoder 输出的中间状态向量和注意力编码的输出并送入分类器，分类器采用同样方法预测输出Q；

9) 依此递归，直到分类器输出<end>标识符。

上述步骤便是本文构建的基于 Seq2Seq+Attention 的自动文本摘要生成模型的一个基本的编码和解码过程，在解码的过程中，将每步的解码结果送入到下一步中去，直到输出<end>结尾标识符。上述流程图也展示了一般的序列到序列模型的训练过程，因为训练数据有对应标注，即输入文本对应的摘要内容，因此模型在解码端能够提前预测每步的输入和输出，即整个流程实际上是输入文本序列 X 和目标摘要序列 Y，预测 $Y_{[1:]}$，即将目标摘要 Y 错开一位来训练。因此 Decoder 在执行每一步时，不能提前使用后面步的输入，所以 Decoder 采用单向的 LSTM 网络而非双向 LSTM。

Seq2Seq 框架经常搭配注意力机制使用，其思想是在每一步解码时，不仅仅要结合 Encoder 编码出来的固定大小的向量，即通读全文，还要往回查阅原来的每一个字词，即精读局部，两者配合来决定当前步的输出。本文中构建的 Seq2Seq 模型，在编码输入序列的同时还需编码输入对应的先验知识，在最后注意力层输出学习结果到 Softmax 时，Softmax 再结合先验知识和注意力层输出的权重结果预测当前步的字符分类概率。与此同时，在解码端采用集束搜索方法，在每一步解码时搜索保留预测概率最大的前 N 个字符，以保证生成摘要的字词有足够大的选择空间。针对自动文本摘要问题，本文结合一系列深度学习方法，构建了上述所示的 Seq2seq+Attention 的基准模型，通过输入短文本到模型中，模型能够学习输出对应的摘要内容。

第三节 模型参数设置

在构建上述的 Seq2seq+Attention 模型的编码器与解码器部分时，本文采用了一种新型的组合方式，Encoder 使用双层 BiLSTM 网络，Decoder 则使用双层单向 LSTM 网络，LSTM 中隐藏层维度都设为 512。实验中将 LCSTS 数据集中 Part1 的短文本和摘要数据作为训练集，part3 的数据作为验证集，其中训练样例的选取是随机的。通过对文本进行逐字符分词之后，从中选取出现次数大于 20 的高频词作为编码器的词汇表，不在词汇表内的字符使用“UNK”表示，过滤低频词汇之后得到的解码器字表大小为 6800。下表展示了基于注意力机制和序列到序列的自动文本摘要生成模型的参数设置：

表 3-1 实验一模型参数设置

参数名称	英文名称	参数值
词向量维度	Embedding_dim	300
优化器	Optimizer	Adam
学习率	Learning_rate	0.001
批尺寸	Batch_size	256
训练步数	step	10000
训练回合	Epoch	20
集束大小	Beam size	10

如表 3-1 所示，本文嵌入的词向量维度设置为 300，并采用交叉熵函数计算损失，使用 Adam 优化器来减少损失和更新模型参数，并将学习率设置

为 0.001。又因为数据量庞大，有限的计算机内存无法一次性加载，所以实验中本文采取生成器批量加载数据，批尺寸大小为 256，一个 Epoch 加载 10000 批次，即 2560000 条数据，以保证模型能够训练全部数据，同时设置 Epoch 为 20，训练 20 个迭代以保证模型能够收敛。其他参数均随机初始化，在解码器输出端，本文采用集束搜索方法，束大小设定为 10，即每一步预测词时，都有 10 个备选答案，以保证模型最终的输出最贴近摘要内容。以下图表展示了基于上述参数设置的模型各层的架构和参数数量：

表 3-2 基于 Seq2seq+Attention 的自动文本摘要生成模型参数架构设置

网络层	英文名称	输出维度	参数数量
输入层	InputLayer	（None，None） （None，None）	0
词嵌入层	Embedding	(None, None, 300)	5726208
标准化层	LayerNormalization	(None, None, 300)	329472
双向长短期记忆网络层	BiLSTM_1	(None, None, 512)	2101248
	BiLSTM_2	(None, None, 512)	1576960
标准化层	LayerNormalization	(None, None, 300)	263424
长短期记忆网络层	LSTM_1	(None, None, 512)	2625536
	LSTM_2	(None, None, 512)	2101248
注意力层	Attention	(None, None, 128)	196608
密集层	Dense_1	(None, None, 300)	492288
	Dense_2	(None, None, 6800)	5229200
缩放平移层	ScaleShift	(None, 1, 6800)	13600
激活输出层	Activation	(None, None, 6800)	0

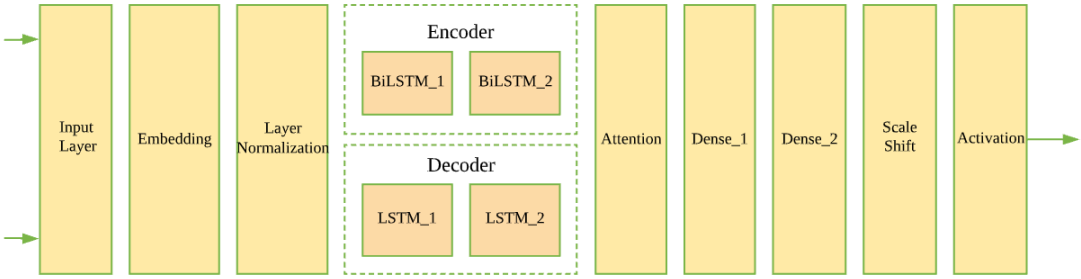

图 3-3 基于 Seq2Seq+Attention 的自动文本摘要生成模型架构

如表 3-2 和图 3-3 所示，模型首先接收输入的短文本和摘要内容，再将其送入到词嵌入层输出 300 维的词向量特征，其中维度前面的 None 表示输入对应的批次大小和最大长度。在模型嵌入层后，连接标准化层并紧跟编码端的双层 BiLSTM 网络和解码端的双层 LSTM 网络。注意力层链接到编码和解码端以结合两者输出，随后紧跟两层密集层，融入先验知识的缩放平移变换层，最后激活输出。以上就是基于 Seq2seq+Attention 的模型的整体架构，

模型各层的参数数量都不同，但最终训练的参数总量达到了两千万个，使得模型训练每轮迭代数据时都需要大量时间，为了达到更好的效果，本文设置迭代次数为 20 并在下文进行了实验。

第四章 基于 Bert 模型的自动文本摘要生成

预训练语言模型一直是自然语言处理界里热门的话题，从 Word2vec[34-35] 到 ELMo[37]再到 BERT[39]，各类表现优异的预训练语言模型相继问世，让学者们逐渐将处理问题的目光从下游的 NLP 具体任务转移到预训练产生词向量上。采用预训练语言模型训练提取词向量，然后将其应用于下游具体的 NLP 任务上再进行模型精调的思路随着 BERT 模型的出现及其卓越表现而逐渐热门起来。基于预训练语言模型的自动文本摘要，采用 BERT 模型预训练语料中的词向量，然后将训练好的词向量输入到序列到序列模型中用以生成文本摘要。本章节将介绍预训练语言模型 BERT 的技术原理，然后基于 BERT 构建实现自动文本摘要生成模型，用以对比上章的 Seq2seq+Attention 模型的表现，探讨将预训练语言模型用于摘要生成任务的可行性。

第一节 Bert 预训练语言模型

Google 团队在 2018 年提出的 BERT 模型[39]，在多项自然语言处理任务中都获得了卓越表现，刷新了以往模型方法得到的结果，证明了采用预训练语言模型做自然语言处理任务的好处。BERT 模型，即基于 Transformer[27]的双向编码器特征表示，首先通过堆叠 Transformer 子结构来构建基础模型，然后进行双向建模，其模型结构如下图所示：

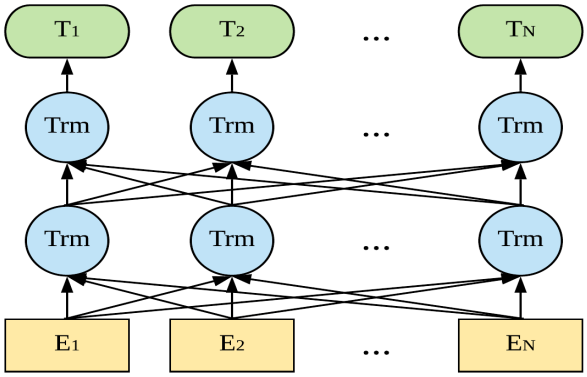

图 4-1 BERT 模型结构

图 4-1 中的 E 为输入序列映射的词向量，Trm 为 Transformer 子结构，T 为其输出，N 为序列长度。Bert 模型中的双向，是指模型在预测某个词时，能够同时利用前面部分的词和后面部分的词的双向信息。不同于以往的预训练语言模型，BERT 并不是结合前文或者后文的所有词信息来预测当前可能性最大的词，而是通过随机遮掩一定比例的词汇，利用没被遮住的词去预测这些词；同时它结合多种特征抽取机制以及采用双向 Transform 作为 Encoder 端以获取上下文相关的双向特征表示，其具体的技术机制主要包括以下几点：

1） 训练语料过程中采用掩码语言模型（Masked Language Model, MLM）机制，随机遮住一定比例的词，尽管模型仍旧看到所有位置信息，但需要预测的词已被特殊符号代替，让模型预测遮住的词，使得模型在训练过程可以做到真正的双向编码，增加词向量模型的泛化能力；

2） 通过上一步中的掩码机制，可以让模型学习理解句子语义，但在很多下游任务中，仅仅获得句子内的语义信息是难以获得优异表现的，还需要捕捉连续句子的整体语义。对此，BERT 又引入句子级负采样来试图学习预测句子级别的任务。即首先给定一个句子，将它的下一句作为正例，随机采样一个句子作为负例，然后在句子级上来做二分类问题，判断句子是当前句子的下一句还是噪声。

3） 采用 Transformer 编码而非 BiLSTM，可以使得模型层数更深、而且具有更好的并行性。相较 RNN 而言更加高效、能够捕捉到更长距离的依赖。线性的 Transformer 比 LSTM 更不容易受到掩码标记的影响，利于捕捉到上下文相关的语义特征表示；

BERT 模型通过上述机制训练大规模的语料，进一步提高了词向量模型的泛化能力，使得模型可以充分描述字符级、词级、句子级甚至句间关系的特征。BERT 模型将传统大量在下游具体 NLP 任务中做的操作转移到预训练词向量中，在获得使用 BERT 训练得到的词向量后，最终只需在词向量上加简单的多层感知器或线性分类器即可应用到其他下游任务中并取得优异的实验结果。使用 BERT 预训练语言模型提取词向量特征，并将其迁移用至下游任务，可以增加下游模型语义表达能力并且获取更加丰富的字、词和句级别的特征表示，而下游模型只需针对实际任务微调即可，节省训练时间。

第二节 基于 BERT 模型的自动文本摘要生成

基于 BERT 预训练语言模型的自动文本摘要实现，同样需要用到序列到序列框架。不同的是，基于 Seq2seq+Attention 的自动文本摘要生成模型中的 Encoder 和 Decoder 的词嵌入层是根据数据集中的语料训练而成的，而且解码器和编码器两端的嵌入层共用参数，使得模型的参数量大幅度减少，训练的词向量的特征表达能力也有限。而基于预训练语言模型的自动文本摘要，采用 BERT 预训练的词向量输入，作为模型中的词嵌入层输出，能够节省模型训练词向量的时间，还能够提取更加丰富的词向量特征表达。对于自动文本摘要问题，假设输入的文本序列为$X = (a, b, c, d, e)$，对应的摘要序列为$Y = (P, Q, R)$，其中每个字母都代表一个字符，结合 Bert 预训练语言模型，本文构建的基于 Bert 的自动文本摘要生成模型具体结构如下图所示：

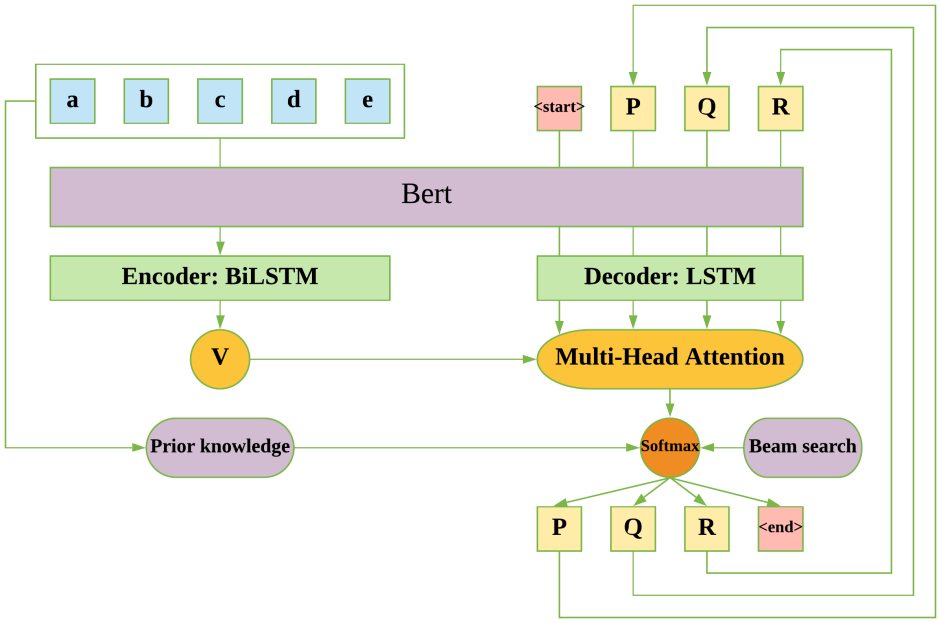

图 4-2 基于 Bert 的自动文本摘要生成模型结构

如图 4-2 所示，在模型的编码端，采用 BiLSTM 网络以捕捉输入文本的上下文特征；在解码端，则采用 LSTM 网路以高效地获取更长距离的依赖；同时将多头注意力机制应用到模型中，让模型能够捕捉到输入序列中的关键字词。基于上述模型结构，具体的编码和解码流程如下：

1） 将输入序列$X = (a, b, c, d, e)$输入到 Bert 预训练语言模型中提取词特征，同时得到输入序列的先验知识信息；

2） Encoder 通过对 Bert 输出的词特征的学习，输出一个中间状态向量V；

3） 将所有输出都以一个通用的<start>标记开头，以<end>标记结尾，这两个标识符也视为一个字符；

4） 将<start>输入到 Bert 模型中，然后得到特征向量；

5） 将 Bert 输出的特征向量输入到 Decoder 解码；

6） 将中间状态向量V与 Decoder 的输出混合输入到多头注意力层中；

7） 将注意力编码、中间状态向量V和 Decoder 的输出三者结合输入到 Softmax 分类器；

8） Softmax 结合输入序列的先验知识和集束搜索方法预测输出分类结果P；

9） 将上步的分类结果P输入 Bert 模型中，再次得到词向量特征，然后输入到 Decoder，得到新的隐藏层向量，再次将其与 Encoder 输出的中间状态向量V和注意力编码的输出混合，送入分类器并采用同样方法预测输出Q；

10）依次递归输出 R 和<end>，当输出<end>标识符即结束。

以上就是基于 Bert 的自动文本摘要生成模型的编码解码过程，与前述

的基于 Seq2seq+Attention 的自动文本摘要生成模型不同，该模型首先需要将输入序列及输出序列送入 Bert 模型中提取词特征，然后在输入到对应的编码和解码端。使用 Bert 模型提取词特征，可以获得更加准确的特征表示，因为 Bert 作为一个预训练语言模型，已经提前学习训练过大批的语料，得到的词向量模型不仅泛化能力强，而且可以充分描述文本中字符级、词级、句子级甚至句间的关系特征。另一方面，将 Bert 提取的词向量特征直接作为输入，模型不再需要训练额外的词嵌入层，可以进一步缩短模型的训练时间，节约资源。前述的 Seq2seq+Attention 模型针对训练语料，添加词嵌入层将输入输出映射为 300 维的特征表示，其特征表达能力很大程度上受到了训练语料的限制。而 Bert 模型通过对海量文本语料的学习，输出 768 维的词向量特征，可以最大程度地保留输入文本的语义表示，使得模型生成的摘要更加准确，贴近文本原义。

第三节 模型参数设置

基于 Bert 预训练语言模型的自动文本摘要实现，本文同样将编码器-解码器框架应用到其中，其中在 Encoder 端使用双层 BiLSTM 网络，Decoder 端使用双层单向 LSTM，网络中隐藏层维度都设为 256。实验中将 LCSTS 数据集中 Part1 的短文本和摘要作为训练数据，part3 的数据作为验证集，其中训练样例的选取是随机的。通过对文本进行逐字符分词之后，输入到预训练模型 Bert 中提取词向量特征，Bert 输出的词向量维度为 768，然后将其输入到编码和解码端中训练得到隐藏层向量。下表展示了基于 Bert 的自动文本摘要生成模型的具体参数设置：

表 4-1 实验二模型参数设置

参数名称	英文名称	参数值
词向量维度	Embedding_dim	768
优化器	Optimizer	Adam
学习率	Learning_rate	0.001
批尺寸	Batch_size	64
训练步数	step	40000
训练回合	Epoch	20
集束大小	Beam size	10

如表 4-1 所示，在实验二，由于采用 Bert 语言模型提取的特征维度为 768，所以词嵌入层维度为 768。训练过程中同样采用交叉熵计算损失，选择 Adam 优化器来减少损失并更新模型参数，学习率设置为 0.001。又因为数据量庞大，随机储存内存无法一次性加载，所以在实验中采取生成器批量加载训练数据，每轮迭代分 40000 步加载文本数据，每批数据包含 64 条文本内容，即 2560000 条数据，以覆盖所有训练数据，同时设置实验的迭代次数为 20，以保证模型能够收敛。其他参数均随机初始化，在解码器输出端，同样采用集束搜索方法，束大小设定为 10，使得模型在每一步预测词的时候，能

从 10 个概率最高的词中选择组合，以保证最终输出最符合摘要标签。以下图表展示了基于上述参数设置的模型各层的架构和参数数量：

表 4-2 基于 Bert 的自动文本摘要生成模型参数架构设置

网络层	英文名称	输出维度	参数数量
输入层	InputLayer	（None，None） （None，None）	0
Bert 特征提取层	Bert_Embedding	(None, None, 768)	5222400
双向长短期记忆网络层	BiLSTM_1	(None, None, 256)	1142784
	BiLSTM_2	(None, None, 256)	919552
长短期记忆网络层	LSTM_1	(None, None, 256)	1667072
	LSTM_2	(None, None, 256)	1576969
注意力层	Attention	(None, None, 128)	196608
密集层	Dense_1	(None, None, 768)	295680
	Dense_2	(None, None, 6800)	5229200
缩放平移层	ScaleShift	(None, 1, 6800)	13600
激活输出层	Activation	(None, None, 6800)	0

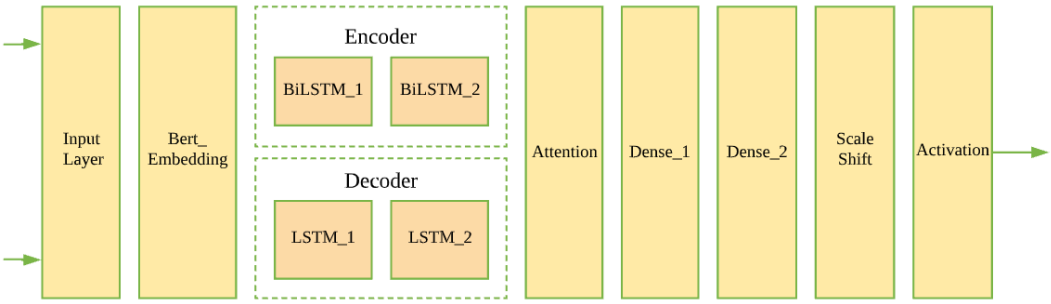

图 4-3 基于 Seq2Seq+Attention 的自动文本摘要生成模型架构

如表 4-2 和图 4-3 所示，模型首先将输入的短文本和摘要内容输入到 Bert 模型中预训练提取词向量，其中两个 None 分别代表批次大小和输入最大长度。在 Bert 词向量嵌入层后，紧接编码端的双层 BiLSTM 网络和解码端的双层 LSTM 网络，此处省略了前者实验模型中的标准化层，因为 Bert 输出的词向量是已经进行归一化处理的，因而省去其中的大量训练参数。接下来 Attention 层结合编码端和解码端的网络层输出后紧跟两层密集层，在融合先验知识的缩放平移层后最终激活输出。以上就是基于 Bert 的模型整体架构，相较于前者的 Seq2seq 模型，该模型各层的参数量都进行了一定削减，训练的参数总量也只有一千多万，因此能够缩短实验所需的周期。

第五章 实验与结果分析

针对自动文本摘要生成问题，前面两章已经构建了相关的模型算法。在

本章节中，将围绕这两种方法展开具体的实验，讲解实验中所用到的数据集及相关的预处理步骤，并叙述实验环境流程及文本摘要中常用的评价指标，最后通过对比两种方法的实验结果，探讨将预处理语言模型用于自动文本摘要生成研究的可行性。

第一节 数据集介绍

本文采用的数据集是在中文自动摘要领域比较权威且广泛使用的短文本摘要数据集[29]LCSTS。该数据集是从社交网站新浪微博上采集爬取的，数据规模超过 200 万，其特点是文本篇幅较短，且存在一定噪声，数据具体的构成情况如表 5-1 所示：

表 5-1 LCSTS 数据集分布

Part 1	2400591	
Part 2	Number of Pairs	10666
	Human Score 1	942
	Human Score 2	1039
	Human Score 3	2019
	Human Score 4	3128
	Human Score 5	3538
Part 3	Number of Pairs	1106
	Human Score 1	165
	Human Score 2	216
	Human Score 3	227
	Human Score 4	301
	Human Score 5	197

LCSTS 数据集划分为三个部分，其中： Part1 包括 2400591 条文本摘要数据； Part2 是从 Part1 中随机采样的 10666 条数据，分为 5 份，分别由 5 位志愿者对每份数据进行标注，标注准则是文本与摘要的匹配程度，其中 1 表示最不相关，5 是最匹配的；Part3 是三名志愿者同时对 2000 条数据进行标注后，从中挑选出三个标注一样的数据，从而得到了 1106 条数据，此部分数据并不包括在 Part1 和 Part2 中。本文实验中采取 Part1 的数据作为训练集，Part2 的数据作为验证集，Part3 的数据作为测试集。

表 5-2 LCSTS 数据集示例

短文本	摘要
本文总结了十个可穿戴产品的设计原则，而这些原则，同样也是笔者认为是这个行业最吸引人的地方：1.为人们解决重复性问题；2.从人开始，而不是从机器开始；3.要引起注意，但不要刻意；4.提升用户能力，而不是取代人	可穿戴技术十大设计原则
2007年乔布斯向人们展示iPhone并宣称“它将会改变世界”，还有人认为他在夸大其词，然而在8年后，以iPhone为代表的触屏智能手机已经席卷全球各个角落。未来，智能手机将会成为“真正的个人电脑”，为人类发展做出更大的贡献。	经济学人：智能手机将成为“真正的个人电脑”
雅虎发布2014年第四季度财报，并推出了免税方式剥离其持有的阿里巴巴集团15%股权的计划，打算将这一价值约400亿美元的宝贵投资分配给股东。截止发稿前，雅虎股价上涨了大约7%，至51.45美元。	雅虎宣布剥离阿里巴巴股份

表 5-2 展示了实验数据的具体几个示例，可以看出其中的文本内容是相对较短的，因为新浪微博的发文字数限制，文本篇幅都控制在 140 个字符以内，而对应的摘要也精辟简洁，长度不超过 30 个字符。

第二节 数据预处理

由于 LCSTS 数据集存在一定噪声，直接输入模型训练将影响模型表现。为此，需要进行数据预处理操作，针对 LCSTS 短文本数据集，本文采取的预处理操作流程如下图所示：

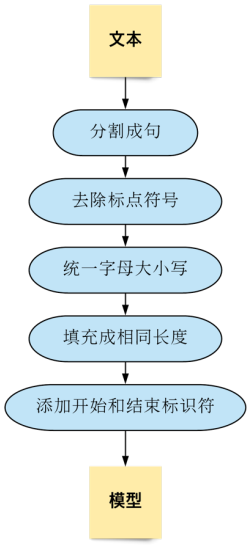

图 5-1 数据预处理步骤

如图 5-1 所示，本文采取的数据预处理操作主要分为 5 步：

1） 将文本分割成句：将读取的 txt 文件按照内容及摘要一条条对应提取，且分割成句；

2） 去除中英文标点符号：由于短文本内容断句较多，且文本中的标点符号中英混杂，而摘要包含的标点符号较少，为了减少标点符号对模型性能的干扰，去除文本中的全部标点符号；

3） 将所有字母转化为小写，重复的字母增加无意义的词维度，增加训练成本，需要去除；

4） 将所有输入填充至一样长度：通过对数据的分析，全部训练数据中文本长度最大的为 140，摘要最大长度为 30。由于文本内容长短不齐，字数不一，为了统一模型的输入维度，将所有文本长度不足 140 的在末尾添加标签<pad>直至长度达到 140；

5） 在文本开始及结尾处分别补上<start>和<end>标签，当模型检测到<start>，即开始输入文本，同理，当检测到<end>，即标记文本输入结束。

在本文实验的预处理步骤中，并没有对文本采取分词操作，因为分词操作会增加词的维度，提高训练成本，且得到的效果其实与基于字符为单位、不分词而输入模型训练的效果相差不大。另一方面，由于 Bert 模型的中文词向量特征提取同样是以字符为单位的，为了更好地对比实验，所以不对数据进行分词操作。在预处理数据之后，直接将其输入到模型中进行训练学习。

第三节 评价指标

对于文本摘要任务，最为权威、应用最为广泛的评价指标是Rouge[51]，该评价体系自提出以来便被广泛应用于自动摘要任务的评价当中，是文本摘要研究领域中公认的评价标准。Rouge主要是根据摘要中n元词$(n-\text{gram})$的共现信息来评价摘要质量，是一种计算摘要内容中n元词的召回率的评价方法。该方法首先由多个专家针对文本分别撰述对应摘要，构成标准摘要集，再将系统生成的摘要内容与专家撰写的标准摘要相对比，通过统计二者之间重叠的n元词数并计算，以评价生成摘要的质量。

Rouge[51]评价体系由一系列的指标组成，包括$\text{Rouge}-\text{N}$，$\text{Rouge}-\text{N}$，$\text{Rouge}-\text{S}$，$\text{Rouge}-\text{W}$和$\text{Rouge}-\text{SU}$等。本文主要选择$\text{Rouge}-\text{N}$以及$\text{Rouge}-\text{L}$作为评价指标，在此详细介绍下两种评价方法：

1）$\text{Rouge}-\text{N}$中的N指的是N元词的模型，通常情况下取值 1～4，$\text{Rouge}-\text{N}$的具体计算公式[51]如下：

$$\text{Rouge}-\text{N}=\frac{\sum_{S\epsilon\{References\}}\sum_{gram_n\epsilon S}Count_{match}\left(gram_n\right)}{\sum_{S\epsilon\{References\}}\sum_{gram_n\epsilon S}Count\left(gram_n\right)} \tag{5-1}$$

在公式（5-1）中，References为专家撰写的所有摘要，S对应其中的一条摘要，gram为摘要中的N元词组，分母为标准摘要中$n-\text{gram}$的个数，分子则是人工摘要和机器生成的自动摘要重合的$n-\text{gram}$的个数。$\text{Rouge}-\text{N}$的优点在于其评价方法直观简洁，能够反映词序，缺点则是区分度不高，而且

随着词组数N的提升，其值会越来越小。

2）Rouge－L是基于最长公共子序列的一种评价指标，其计算公式[51]如下所示：

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{5-2}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{5-3}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \tag{5-4}$$

其中，序列X和Y为参考摘要和机器生成的摘要，$LCS(X,Y)$是X和Y的最长公共子序列的长度，m,n分别表示专家撰写的摘要和机器生成的摘要长度，R_{lcs}，P_{lcs}分别对应召回率和准确率，β^2为调节参数，最后计算得到的F_{lcs}即是Rouge－L。Rouge－L的优点在于其对词的连续性不作要求，只要词出现的顺序匹配即可，能够反映句子级的语序，而且可以自动匹配最长公共子序列，但因为其只计算一个最长的子序列，容易忽略其他备选的最长子序列以及较短的子序列的影响。

在本文中，主要采用Rouge－1、Rouge－2和Rouge－L三种方式对模型生成的摘要质量进行评价，三种方式分别从字的相似度、词的相似度和最长公共子序列的匹配程度三个方面来评价摘要质量，对比模型性能表现。

第四节 实验环境与流程

由于本文中所用的语料规模大，且构建的模型也相对复杂，特别是 Bert 对语料的预训练需要耗费大量时间资源，相应的计算量也十分巨大，需要用到 GPU，且对硬件的要求也较高。因此，本文所有的实验都在 Google 的云服务平台 Colab 上完成，具体的实验环境硬件如下表所下：

表 5-3 实验环境软硬件配置

名称	配置
操作系统	Ubuntu18.04
RAM	25.51GB
内存	256GB
CPU	Intel(R) Xeon® 2*2.30GHZ
GPU 型号	Tesla P100
GPU 个数	1

此外，实验所有代码都使用 python3.6 编写，并使用 Keras 深度学习框架构架算法模型。在搭建准备好实验环境后，基于上述两章设计的模型算法开始实验，具体的实验流程如下图所示：

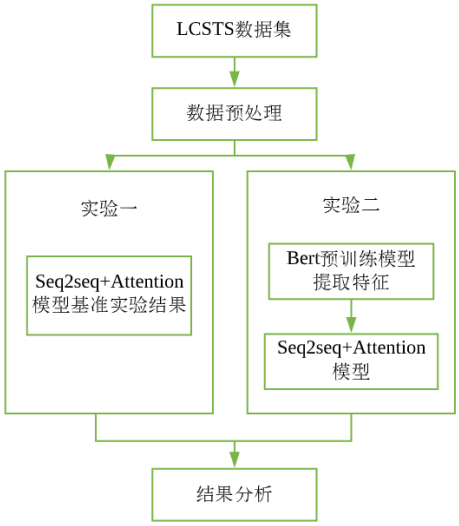

图 5-2 实验流程图

如图 5-2 所示，针对自动文本摘要问题，本文采用了 LCSTS 数据集作为实验语料，在数据预处理操作之后，将其输入到提出设计的两个实验之中进行训练，其中实验一为基于注意力机制和序列到序列模型的自动摘要生成方法,；实验二为基于 Bert 预训练语言模型的自动文本摘要生成方法。实验一中构建实现的基准模型，用于得出初步的实验结果，而实验二中的模型则是结合预训练方法提出的创新模型，通过两个模型方法的实验结果对比，最后分析将预训练语言模型用于实现自动文本摘要技术的可行性和有效性。

第五节　实验结果分析比较

针对文本自动摘要的实现，本文在上述两章分别设计了基于 Seq2Seq+Attention 和 Bert 的自动文本摘要生成模型。在本节中，将根据两个方法的实验结果进行详细分析说明。在前述的数据预处理步骤和模型参数设置之后，将文本数据输入到模型中并训练 20 个迭代，下图展示了两个实验的模型训练过程中损失的变化曲线：

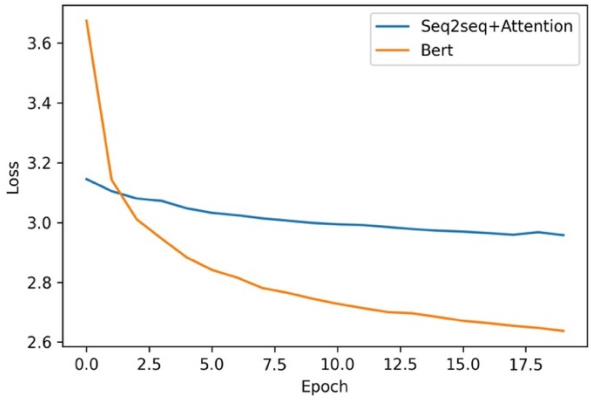

图 5-3 模型训练损失比较

在图 5-3 中，横坐标为训练迭代数，纵坐标为交叉熵损失值；如图可知，在同样的迭代次数下，基于 Bert 的自动摘要模型，在最开始的迭代中其损失略高于基于 Seq2seq+Attention 的模型，但后面迭代中收敛更快，最终的损失也更低。因为基于 Bert 模型的实验起始时使用的是 Bert 提取词向量而非由原始数据训练提取的词向量，所以使得模型在开始时损失偏大。但随后的几个 epoch 中，采用 Bert 提取词向量的方法的优势逐渐凸显出来，其后续训练中 loss 的下降速度明显快于基于 Seq2seq+Attention 的实验。在训练 10 个迭代之后，两个模型 loss 都开始收敛并趋于稳定，最终稳定下来后，后者模型的损失值更低 0.5，说明基于 Bert 的模型方法更加有效，能够获得更低的损失说明模型预测的误差相对更小。通过分析比较模型训练过程中损失的变化图，可以推断基于 Bert 预训练语言模型的文本摘要方法能够合理捕捉词向量特征，有效降低训练过程中的损失，生成的摘要准确性也超过基于 Seq2seq+Attention 的自动文本摘要生成模型的结果。

表 5-4 模型测试结果比较

模型方法	Rouge-1	Rouge-2	Rouge-L	Time/Epoch
RNN[29]	0.177	0.085	0.158	/
Seq2seq+Attention	0.183	0.06	0.176	4110s
Bert	0.21	0.09	0.19	1580s

从表 5-4 展示的 Rouge 测试分数可以看出，本文提出的基于 Bert 的文本摘要模型的实验结果明显优于基于 RNN 的文本摘要模型以及基于 Seq2seq+Attention 的文本摘要模型，Rouge 各项评价分数都有一定的提升。具体而言，在模型训练过程中，追踪验证集损失的变化并选择保存使得验证集损失最低的模型参数，再将训练好的模型在 LCSTS 数据集中 part2 数据上的测试 Rouge 分数。表 5-4 中的第一行的实验结果为 LCSTS 数据集论文提供的基准实验结果[29]，该实验中采用了 RNN 网络构建编码-解码端，示例中为采取了分词操作的实验结果，作为本文的研究结果比较。第二行的实验结果对应本文中的基于 Seq2seq+Attention 模型实验，第三行为基于 BERT 预训练语言模型的实验结果。由表可知，相较于基准 RNN 模型的实验结果，基于 Seq2seq+Attention 的方法模型虽然在 Rouge-1 和 Rouge-L 的指标上有一定的提升，但 Rouge-2 的测试分数上仍然低于基准方法。这是由于基准 RNN 模型在实验中采取了分词操作，所以在每一步结果预测的时候，实际上是逐词生成的，而基于 Seq2seq+Attention 的文本摘要模型在实验中未采取分词操作，所以其结果生成是逐字符的，在预测连续字符时会在一定程度上受到模型表现的影响。因此，在二元词组的评价指标上，基于 Seq2seq+Attention 的模型实验结果会略低于基准 RNN 模型。而在基于 Bert 预训练语言模型的自动文本摘要生成方法的实验结果中，其三项 Rouge 指标都高于前面两个实验，且相对基准 RNN 模型的实验结果，在 Rouge-1 和 Rouge-L 指标的测试结果上，都有大幅提升，Rouge-2 分数也有一定提升，这也证明了分词操作对于一般的实验结果影响并不大。通过分析模型在测试数据上的 Rouge 评分，可以推断基于预训练语言模型的文本摘要方法优于一般的 RNN 模型及 Seq2seq+Attention 方法，可以有效提取文本中的深层语义信息和词向量特征表示，生成高质量，连贯准确的摘要内容。

表 5-4 最后一栏展示了实验中训练每轮迭代所需的时间，由表可知，基于 Bert 的模型实验训练时间大幅快于基于 Seq2seq+Attention 的模型实验。具体而言，基于 Seq2seq+Attention 的模型实验训练每轮数据都需一个小时以上，而采用 Bert 预训练语言模型的实验二训练每轮数据所需时间则不超过半小时，相比之下缩短了一倍以上。这是因为采用预训练模型提取词向量的方法，只需在开头提取词向量时耗费一些时间，可以省去后续的词嵌入层，减少训练该层所需的参数。本文中基于 Bert 模型的实验训练参数相较于基于 Seq2seq+Attention 的模型的实验一，减少的参数量超过了 500 万，而模型总共所需训练的参数也只有 1200 万，因此可以大幅减少整个实验的训练时间，并且采用 Bert 提取的词向量特征也更加丰富，因此整体的实验结果也相对更好。通过表三的 Rouge 测试分数分析，可以说明采用预训练语言模型提取词向量特征并用于后续训练的方法能够有效提高模型表现和摘要质量，同时大幅减少实验所需时间。

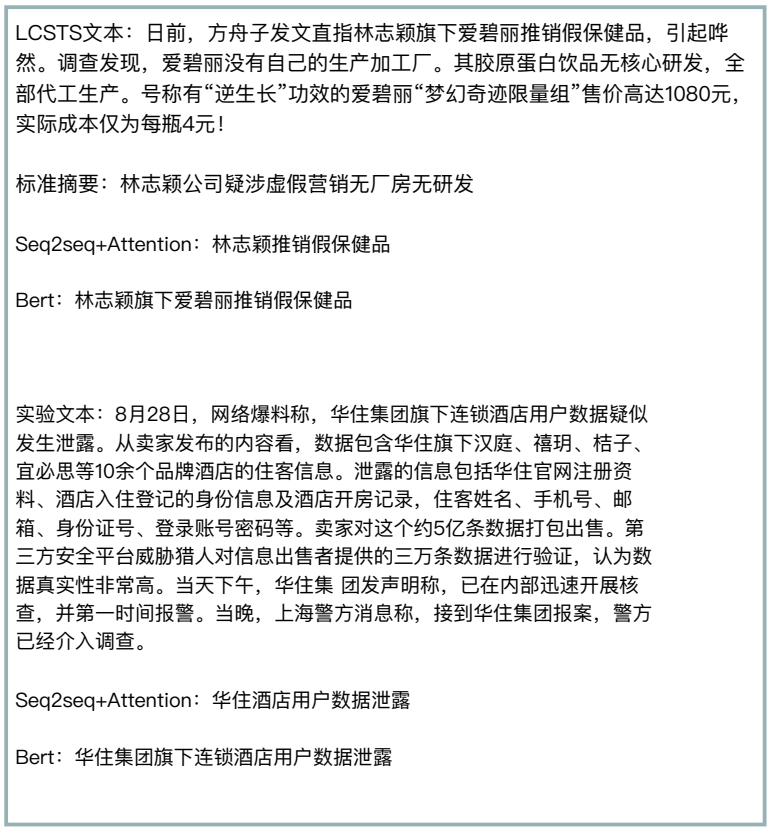

图 5-3 模型生成摘要对比

图 5-3 展示了两个实验模型生成的摘要示例，由图可知，基于 Bert 模型生成的文本摘要贴切通顺，比基于 Seq2seq+Attention 的模型生成的摘要内容更加丰富具体、连贯流畅。具体而言，上图中第一条文本为 LCSTS 数据集中的一个示例，两个方法生成的摘要示例都言简流畅，表明模型能够把握住全文大意及主谓宾成分，摘要内容准确贴合文本含义。采用 Bert 模型的文本摘要方法可以精确捕捉到词向量之间的联系及上下文语义信息，所以能够预测到“旗下爱碧丽”这样的细节信息，而 Seq2seq 模型则捕捉不到这样

的细节内容。但数据集提供的标准摘要更加抽象，语义也更复杂丰富，其手动摘要内容甚至远超常人水平，所以尽管模型生成的摘要比较准确，参照标准摘要后其 Rouge 评价分数也相对会低些。第二条文本采集自网络新闻，以考察模型对于实际新闻的摘要生成效果。可以看出，针对该条信息，两个模型方法都可以准确生成高可读性的新闻摘要。不同的是，基于 Bert 模型的方法可以预测出更具体的内容，比如“旗下连锁酒店”这种二级信息。实验说明基于 Bert 模型的自动文本摘要方法比一般的 Seq2seq 模型表现更加出色，可以捕捉预测到更深层次的语义信息，生成的摘要内容也更加准确连贯。

进一步分析两项示例，由于短文本主旨内容大多出现在开头，因此模型生成的摘要会更加偏向于文本前面的内容，而忽略了后面的内容信息，使得模型生成的摘要难以涵盖到之后的次级信息。综上分析，尽管基于预训练语言模型的自动摘要方法在预测文本的细节信息方面仍需进一步优化改进，但总体而言，该方法生成的摘要内容质量较高、连贯流畅，也准确贴合文意，这也证明了将预训练语言模型用于自动文本摘要生成研究的可行性及有效性。

第六章 总结与展望

第一节 工作总结

随着互联网的普及发展，围绕在人们日常生活中的信息呈现爆炸性增长，信息抽取总结技术也愈发重要。针对此问题，本文先后查阅大量国内外研究资料，学习总结了前人的研究方法及模型后，提出了基于预训练语言模型的自动文本摘要方法，通过将中文短文本输入到模型中，自动输出对应的摘要。该摘要生成技术可以有效帮助人们总结并抽取所需信息，提升阅读效率，节约大量时间精力。简而言之，本文主要做了以下几点研究工作：

1) 论述了文本自动摘要技术的相关研究、实现方法和技术手段，并在本文中采用深度学习技术，研究生成式文本摘要的实现方法。选择中文大规模文本数据LCSTS作为实验数据，其数据量超过200万条，并构建Seq2Seq框架，融合注意力机制、先验知识和集束搜索等深度学习技术构建基准模型，以实现文本的自动摘要并得到初步的实验结果。采用 Rouge 评价体系评估生成的摘要质量，对比数据集论文提出的基准结果，本文的初步实验结果在 Rouge-1 和 Rouge-L 指标上都有一定提升。

2) 阐述预训练语言模型的发展历程，并介绍说明了 Bert 预训练模型的关键技术及其原理，提出设计了基于 Bert 预训练语言模型的自动文本摘要的创新方法。构建相应的 Seq2Seq+Attention 的框架，其中编码端采用双层双向 BiLSTM 网络，解码端则采用双层单向 LSTM 网络。通过 Bert 模型提取文本和摘要的词向量特征后，将提取的特征输入到对应的编码端和解码端训练学习，在学习迭代同样的次数后得出实验结果。对比数据集论文提出的基准结果和本文的初步实验结果，基于 Bert 模型的方法在 Rouge 三项指标上的测试分数都大幅超过之前的实验，而且实验的训练

时间也大幅减少，说明了基于预训练语言模型的自动文本摘要方法能够有效提高生成摘要的质量和连贯性，并减少训练时间和成本。

第二节 工作展望

尽管现有的一些文本摘要技术已经发展成熟并投入到实际使用中，但在生成摘要的可读性和准确性上，仍需不断研究改进。本文研究总结了基于预训练语言模型的文本摘要方法，该方法生成的摘要质量高，可读性强，内容也更加合理准确，但仍有一些不足的地方需要进一步改进。为此，本文对之后的研究作出了以下几点展望：

1) 本文提出设计的方法都是基于字符单位的，数据未采取分词操作，即模型在解码端逐字符生成对应的文本摘要，同时预训练语言模型提取的词向量特征也是字符向量特征。因为英文数据不需分词操作，以单词为单位向量即可，同样地中文数据采用字向量特征可以有效减少数据维度规模，但在一定程度上忽略了中文 2 元词组中字符之间的连续性，这也使得实验结果在 Rouge-2 指标上略低。如果在之后工作中考虑分词操作并采取对应的实验，或许能够进一步验证分词操作的影响，也能在词级层面生成更加连贯贴切的摘要。

2) 采用 Bert 预训练语言模型提取的词向量特征可以表征文本的上下文信息，但其对语义的建模能力仍是有限的。考虑到 Bert 预训练过程和生成过程的不一致，直接在编码端和解码端使用 Bert 提取词向量仍存在一定缺陷。在之后的研究工作中，在编码端使用 Bert 或者其他表现更优的 XLNet、AlBert 等预训练模型，在解码端采用不一样的模型，能够进一步提升生成摘要的质量和准确性。

3) 中文文本摘要研究起步较晚，相应的模型方法、文本数据也远不如英文领域全面。尽管哈工大收集发布的大规模中文短文本数据集 LCSTS 弥补了中文数据集的缺失，但中文语言相对抽象，数据集中文本对应的摘要也并非标准答案，而 Rouge 指标只针对标准摘要和模型生成的摘要评价，使得标准摘要极大地影响了模型的评价分数。在之后的实验工作中，考虑摘要内容的同义多样性，并在此基础上优化评价指标，能够更加准确地评估模型的性能。

致　　谢

匆匆大学四年，回看恍如昨日。在这漫长而又短暂的岁月中，我收获了很多，对所学专业知识的掌握，对大学生活历程的感悟，对课题研究的认知，展望未来道路时的坚定，结识良师益友的喜悦……种种一切，铭记在心。为此，我要感谢大学以来关心帮助我的同学老师并向他们致以最真挚的感谢。

感谢导师陈娟对我的悉心教导，从最初的选题到最后的查重答辩，老师都一直跟进关心我的研究进度，并在我遇到问题时及时给予帮助，提供建议。在我实验遇到问题时耐心指导，让我少走了很多弯路。老师严谨的学术研究态度也是我论文写作的榜样，让我端正自己的态度，更加认真地对待毕业论文和研究工作。在此，我要特别感谢陈娟老师一直以来对我的指导与帮助。

最后，我要感谢学校，感谢在上海大学学习生活的四年，感谢每一位老师的教诲，感谢同学们的帮助，正因为你们，我才能不断进步成长，直面困难，勇于挑战自我。

参考文献

【1】. 张少迪.基于深度学习的文本摘要生成技术研究[D].新疆大学,2019:1-58.

【2】. Krishnaveni P, Balasundaram S. Automatic text summarization by local scoring and ranking for improving coherence. 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, 2017,59-64.

【3】. Lebret, Rémi, David Grangier, Michael Auli. Neural text generation from structured data with application to the biography domain. arXiv:1603.07771,2016.

【4】. 户保田.基于深度神经网络的文本表示及其应用[D].哈尔滨工业大学，2016:91-94.

【5】. CHENG J, LAPATA M. Neural summarization by extracting sentences and words[J]. arXiv :1603.07252,2016.

【6】. NEMA P, KHAPRA M, LAHA A, et al.Diversity driven attention model for query-based abstractive summarization. arXiv:1704.08300,2017.

【7】. S. R. Rahimi, A. T. Mozhdehi, M. Abdolahi. An overview on extractive text summarization. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, 2017,0054-0062.

【8】. S. Modi R. Oza. Review on Abstractive Text Summarization Techniques (ATST) for single and multi-documents. 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, Uttar Pradesh, India, 2018, 1173-1176.

【9】. 明拓思宇，陈鸿昶.文本摘要研究进展与趋势[J].网络与信息安全学

报.2018,6:1-10.

【10】. 李舟军，范宇，吴贤杰. 面向自然语言处理的预训练技术研究综述[J]. 计算机科学.2020,3:1-16.

【11】. Luhn HP. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development,1958,2,159-165.

【12】. KUPIEC J，PEDERSEN J，CHEN F.A trainable document summarizer [C]. The 18th annual International ACM SIGIR Conference on Research and Development in Information Retrivel.1995:68-73.

【13】. THUHN T. An optimization text summarization method based on naïve bayes and topic word for single syllable language[J]. Applied Mathematical Sciences,2014,8:99-115.

【14】. SILVA G, FERREIRA R, LINSR D, et al. Automatic text document summarization based on machine learning[C].2015 ACM Symposium on Document Engineering.ACM,2015:191-194.

【15】. PAGE L. The PageRank citation ranking: Bring order to the web[J]. Stanford Digital Libraries Working Paper,1998,9:1-14.

【16】. Mihalcea R, Tarau P.TextRanking: Bringing Order into Text[C].In Proceedings of Empirical Methods in Natural Language Processing,Barcelona,Spain.2004:404-411.

【17】. Erkan G, Radev D R. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization[J]. Journal of Qiqihar Junior Teachers College,2011,22:2004.

【18】. 李娜娜. 基于 TextRank 的文本自动摘要研究[D]. 山东师范大学,2019:1-50.

【19】. 耿焕同,蔡庆生,赵鹏,等.一种基于词共现图的文档自动摘要研究[J].情报学报，2005，642:652.

【20】. 吴晓峰, 宗成庆. 一种基于 LDA 的 CRF 自动文摘方法[J].中文信息学报,2009,23:39-45.

【21】. RUSH A M, CHOPRA S, WESTON J. A Neural Attention Model for Abstractive Sentence Summarization[J]. arXiv:1509.00685.2015.

【22】. SUTSKEVIR I, VINYAIS O, Le Q V. Sequence to Sequence learning with neural networks[C]//Advances in neural information processing systems,2014:3014-3112.

【23】. LeCun Y, et al. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation,1989,1:541-551.

【24】. CHOPRA S, AULI M, RUSH A M. Abstractive sentence summarization with attentive recurrent neural networks[C]//The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,2016:93-98.

【25】. Rumelhart D E, Hinton G E, Williams R J. Learning internal representations by error propagation[R]. California Univ San Diego La Jolla Inst for Cognitive Science,1985.

【26】. NALLAPATI R, ZHOU B W, dos SANTOSCN, et al. Abstractive text summarization using sequence-to-sequence RNNs and beyond [C]//Proceedings of the20th SIGNLL Conference on Computation-al Natural Language Learning. Stroudsburg, PA: ACL, 2016:280-290.

【27】. Ashish V, Noam S, Niki P, et al, Attention is all you need [J], 2017.

【28】. Romain P, Caiming X, Richard S. A Deep Reinforced Model for Abstractive Summarization[C]//The 2017 International Conference on Learning Representations, 2017.

【29】. Hu B, Chen Q, Zhu F. LCSTS: A large scale Chinese short text summarization dataset [C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 1967-

1972.

【30】. 丁建立,李洋,王家亮. 基于双编码器的短文本自动摘要方法[J]. 计算机应用，2019,39(12):3476-3481.

【31】. 施旭涛. 基于堆叠 BiLSTM 的中文自动文本摘要研究[D].云南大学,2019:1-65.

【32】. Schuster M, Paliwal K K. Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing.1997, 45(11):2673-2681.

【33】. Deng L, Yu D. Deep learning: methods and applications[J]. Foundations and Trends® in Signal Processing, 2014,7:197-387.

【34】. MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. arXiv:1301.3781.

【35】. MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in Neural Information Processing Systems. 2013: 3111-3119.

【36】. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014: 1532-1543.

【37】. Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv:1802.05365.

【38】. Radford A, Narasimhan K, Salimans T, et al. Improving Language Understanding by Generative Pre-Training[J]. arXiv:2018.

【39】. Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv:1810.04805.

【40】. Yang Z, Dai Z, Yang Y, et al. XLNet: Generalized Autoregressive

Pretraining for Language Understanding[J]. arXiv:1906.08237.

【41】. Sun Y, Wang S, Li Y, et al. ERNIE: Enhanced Representation through Knowledge Integration[J]. arXiv:1904.09223.

【42】. Zhang Z, Han X, Liu Z, et al. ERNIE: Enhanced Language Representation with Informative Entities[J]. arXiv:1905.07129,2017.

【43】. Sun Y, Wang S, Li Y, et al. Ernie 2.0: A continual pre-training framework for language understanding[J].arXiv:1907.12412.

【44】. Zhang H, Gong Y, Yan Y, et al. Pretraining Based Natural Language Generation for Text Summarization. arXiv: 2019.

【45】. 岳一峰,黄蔚,任祥辉. 一种基于 BERT 的自动文本摘要模型构建方法[J]. 计算机与现代化,2020,293(1):63-68.

【46】. H. A. Chopade and M. Narvekar, "Hybrid auto text summarization using deep neural network and fuzzy logic system," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, 2017, pp. 52-56.

【47】. Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. Neural computation,9(8):1735-1780,1997.

【48】. M. S. Hasibuan, L. Nugroho and P. Santosa, "Prediction Learning Style Based on prior Knowledge for Personalized Learning," 2018 4th International Conference on Science and Technology (ICST), Yogyakarta, 2018, pp. 1-5.

【49】. H. Duan and N. Liu, "A Greedy Search Algorithm for Resolving the Lowermost C Threshold in SVM Classification," 2013 Ninth International Conference on Computational Intelligence and Security, Leshan, 2013, pp. 190-193.

【50】. S. Ortmanns, A. Eiden, H. Ney and N. Coenen, "Look-ahead techniques for fast beam search," 1997 IEEE International

Conference on Acoustics, Speech, and Signal Processing, Munich, 1997, pp. 1783-1786 vol.3.

【51】. Chin-Yew Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In Proceedings of 2003 Language Technology Conference (HLTNAACL 2003), Edmonton, Canada.

附录：

LCSTS: A Large Scale Chinese Short Text Summarization Dataset

Abstract
Automatic text summarization is widely regarded as the highly difficult problem, partially because of the lack of large text summarization data set. Due to the great challenge of constructing the large scale summaries for full text, in this paper, we introduce a large corpus of Chinese short text summarization dataset constructed from the Chinese microblogging website Sina Weibo, which is released to the public. This corpus consists of over 2 million real Chinese short texts with short summaries given by the author of each text. We also manually tagged the relevance of 10,666 short summaries with their corresponding short texts. Based on the corpus, we introduce recurrent neural network for the summary generation and achieve promising results, which not only shows the usefulness of the proposed corpus for short text summarization research, but also provides a baseline for further re- search on this topic.

1 Introduction
Nowadays, individuals or organizations can easily share or post information to the public on the social network. Take the popular Chinese microblogging website (Sina Weibo) as an example, the People's Daily, one of the media in China, posts more than tens of weibos (analogous to tweets) each day. Most of these weibos are well- written and highly informative because of the text length limitation (less than140 Chinese characters). Such data is regarded as naturally annotated web resources (Sun, 2011). If we can mine these high-quality data from these naturally annotated web resources, it will be beneficial to the research that has been hampered by the lack of data.
In the Natural Language Processing (NLP) community, automatic text summarization is a hot and difficult task. A good summarization system should understand the whole text and re-organize the information to generate coherent, informative, and significantly short summaries which convey important information of the original text (Hovy and Lin, 1998), (Martins, 2007). Most of traditional abstractive summarization methods divide the process into two phrases (Bing et al., 2015). First, key textual elements are extracted from the original text by using unsupervised methods or linguistic knowledge. And then, unclear extracted components are rewritten or paraphrased to pro- duce a concise summary of the original text by using linguistic rules or language generation techniques. Although extensive researches have been done, the linguistic quality of abstractive summary is still far from satisfactory. Recently, deep learning methods have shown potential abilities to learn representation (Hu et al., 2014; Zhou et al., 2015) and generate language (Bahdanau et al., 2014; Sutskever et al., 2014) from large scale data by utilizing GPUs. Many researchers realize that we are closer to generate abstractive summarizations by using the deep learning methods. How- ever, the publicly available

and high-quality large scale summarization data set is still very rare and not easy to be constructed manually. For example, the popular document summarization dataset DUC2, TAC3 and TREC4 have only hundreds of human written English text summarizations. The problem is even worse for Chinese. In this paper, we take one step back and focus on constructing LCSTS, the Large-scale Chinese Short Text Summarization dataset by utilizing the naturally annotated web resources on Sina Weibo. Figure 1 shows one weibo posted by the People's Daily. In order to convey the import information to the public quickly, it also writes a very informative and short summary (in the blue circle) of the news. Our goal is to mine a large scale, high-quality short text summarization dataset from these texts.

This paper makes the following contributions:

(1) We introduce a large scale Chinese short text summarization dataset. To our knowledge, it is the largest one to date;

(2) We provide standard splits for the dataset into large scale training set and human labeled test set which will be easier for benchmarking the related methods;

(3) We explore the properties of the dataset and sample 10,666 instances for manually checking and scoring the quality of the dataset;

(4) We perform recurrent neural network based encoder-decoder method on the dataset to generate summary and get promising results, which can be used as one baseline of the task.

2 Related Work

Our work is related to recent works on automatic text summarization and natural language processing based on naturally annotated web resources, which are briefly introduced as follows.

Automatic Text Summarization in some form has been studied since 1950. Since then, most re- searches are related to extractive summarizations by analyzing the organization of the words in the document (Nenkova and McKeown, 2011) (Luhn, 1998); Since it needs labeled data sets for supervised machine learning methods and labeling dataset is very intensive, some researches focused on the unsupervised methods (Mihalcea, 2004). The scale of existing data sets are usually very small (most of them are less than 1000). For example, DUC2002 dataset contains 567 documents and each document is provided with two 100-words human summaries. Our work is also related to the headline generation, which is a task to generate one sentence of the text it entitles. Colmenares et.al construct a 1.3 million financial news headline dataset written in English for head-line generation (Colmenares et al., 2015). However, the data set is not publicly available. Naturally Annotated Web Resources based Natural Language Processing is proposed by Sun (Sun, 2011). Naturally Annotated Web Re- sources is the data generated by users for communicative purposes such as web pages, blogs and microblogs. We can mine knowledge or useful data from these raw data by using marks generated by users unintentionally. Jure et.al track 1.6 million mainstream media sites and blogs and mine a set of novel and persistent temporal patterns in the news cycle (Leskovec et al., 2009). Sepandar et.al use the users' naturally annotated pattern 'we feel' and 'i feel' to extract

the ‘Feeling’ sentence collection which is used to collect the world’s emotions. In this work, we use the naturally annotated re- sources to construct the large scale Chinese short text summarization data to facilitate the research on text summarization.

3 Data Collection

A lot of popular Chinese media and organizations have created accounts on the Sina Weibo. They use their accounts to post news and information. These accounts are verified on the Weibo and labeled by a blue ‘V’. In order to guarantee the quality of the crawled text, we only crawl the verified organizations’ weibos which are more likely to be clean, formal and informative. There are a lot of human intervention required in each step. The process of the data collection is shown as Figure 2 and summarized as follows:

1) We first collect 50 very popular organization users as seeds. They come from the domains of politic, economic, military, movies, game and etc, such as People’s Daily, the Economic Observe press, the Ministry of National Defense and etc.

2) We then crawl fusers followed by these seed users and filter them by using human written rules such as the user must be blue verified, the number of followers is more than 1 million and etc.

3) We use the chosen users and text crawler to crawl their weibos.

4) we filter, clean and extract (short text, summary) pairs. About 100 rules are used to extract high quality pairs.

These rules are concluded by 5 peoples via carefully investigating of the raw text. We also remove those paris, whose short text length is too short (less than 80 characters) and length of summaries is out of [10,30].

4 Data Properties

The dataset consists of three parts shown as Table 1 and the length distributions of texts are shown as Figure 3.

Part I is the main content of LCSTS that contains 2,400,591 (short text, summary) pairs. These pairs can be used to train supervised learning model for summary generation.

Part II contains the 10,666 human labeled (short text, summary) pairs with the score ranges from 1 to 5 that indicates the relevance between the short text and the corresponding summary. ‘1’ denotes “the least relevant” and ‘5’ denotes “the most relevant”. For annotating this part, we recruit 5 volunteers, each pair is only labeled by one annotator. These pairs are randomly sampled from Part I and are used to analysize the distribution of pairs in the Part I. Figure 4 illustrates examples of different scores. From the examples we can see that pairs scored by 3, 4 or 5 are very relevant to the corresponding summaries. These summaries are highly informative, concise and significantly short compared to original text. We can also see that many words in the summary do not appear in the original text, which indicates the significant difference of our dataset from sentence compression datasets. The summaries of pairs scored by 1 or 2 are highly abstractive and relatively hard to conclude the summaries from the short text. They are more likely to be headlines or comments in- stead of summaries. The statistics show that the percent of score 1 and 2 is less than 20% of the data, which can

be filtered by using trained classifier.
Part III contains 1,106 pairs. For this part, 3 annotators label the same 2000 texts and we ex- tract the text with common scores. This part is independent from Part I and Part II. In this work, we use pairs scored by 3, 4 and 5 of this part as the test set for short text summary generation task.

5 Experiment
Recently, recurrent neural network (RNN) have shown powerful abilities on speech recognition (Graves et al., 2013), machine translation (Sutskever et al., 2014) and automatic dialog response (Shang et al., 2015). However, there is rare research on the automatic text summarization by using deep models. In this section, we use RNN as encoder and decoder to generate the summary of short text. We use the Part I as the training set and the subset of Part III, which is scored by 3, 4 and 5, as test set.
Two approaches are used to preprocess the data:
1) character-based method, we take the Chinese character as input, which will reduce the vocabulary size to 4,000.
2) word-based method, the text is segmented into Chinese words by using jieba5. The vocabulary is limited to 50,000. We adopt two deep architectures：
1) The local con- text is not used during decoding. We use the RNN as encoder and it's last hidden state as the input of decoder, as shown in Figure 5;
2) The context is used during decoding, following (Bahdanau et al., 2014), we use the combination of all the hidden states of encoder as input of the decoder, as shown in Figure 6. For the RNN, we adopt the gated recurrent unit (GRU) which is proposed by (Chung et al., 2015) and has been proved comparable to LSTM (Chung et al., 2014). All the parameters (including the embeddings) of the two architectures are randomly initialized and ADADELTA (Zeiler, 2012) is used to update the learning rate. After the model is trained, the beam search is used to generate the best summaries in the process of decoding and the size of beam is set to 10 in our experiment.
For evaluation, we adopt the ROUGE metrics proposed by (Lin and Hovy, 2003), which has been proved strongly correlated with human evaluations. ROUGE-1, ROUGE-2 and ROUGE-L are used. Because the standard Rouge package 6 is used for evaluating English summarization systems, we transform the Chinese words to numerical IDs to adapt to the systems. All the models are trained on the GPUs tesla M2090 for about one week. Table 2 lists the experiment results. As we can see in Figure 7, the summaries generated by RNN with context are very close to human written summaries, which indicates that if we feed enough data to the RNN encoder and decoder, it may gen- erate summary almost from scratch.
The results also show that the RNN with con- text outperforms RNN without context on both character and word based input. This result indicates that the internal hidden states of the RNN encoder can be combined to represent the context of words in summary. And also the performances of the character-based input outperform the word-based input. As shown in Figure 8, the summary generated by RNN with context by inputing the character-based short text is relatively good, while the the summary

generated by RNN with context on word-based input contains many UNKs. This may attribute to that the segmentation may lead to many UNKs in the vocabulary and text such as the person name and organization name. For example, “愿 景 光 电 子” is a company name which is not in the vocabulary of word-based RNN, the RNN summarizer has to use the UNKs to replace the “愿景光电子” in the process of decoding.

6 Conclusion and Future Work

We constructed a large-scale Chinese short text summarization dataset and performed RNN-based methods on it, which achieved some promising results. This is just a start of deep models on this task and there is much room for improvement. We take the whole short text as one sequence, this may not be very reasonable, because most of short texts contain several sentences. A hierarchical RNN (Li et al., 2015) is one possible direction. The rare word problem is also very important for the generation of the summaries, especially when the input is word-based instead of character-based. It is also a hot topic in the neural generative models such as neural translation machine (NTM) (Luong et al., 2014), which can benefit to this task. We also plan to construct a large document summarization data set by using naturally annotated web resources.

LCSTS：大规模中文短文本摘要数据集

摘要
自动文本摘要被广泛认为是一个非常困难的问题，部分原因是缺少大型文本摘要数据集。鉴于构建全文本的大规模摘要所面临的巨大挑战，在本文中，我们提出介绍了一个由中国微博网站--新浪微博构建的大型中文短文本摘要数据集，并已向公众发布。该语料库包含超过 200 万条实际的中文短文本，并由每个文本的作者给出了简短的摘要。我们还手动将 10,666 条摘要和其相应的简短文本标记了相关性程度标签。在语料库的基础上，我们引入了循环神经网络进行摘要生成并取得了可喜的结果，这不仅表明了本文提供的语料对于短文本摘要研究的有用性，而且还为该主题的进一步研究提供了基准结果。

1 引言

如今，个人或组织可以在社交网络上轻松地将信息共享或发布给公众。以中国流行的微博网站（新浪微博）为例，中国的媒体之一《人民日报》每天发布数十条微博（类似于推文）。由于文本长度的限制（少于 140 个中文字符），大多数这些 weibos 都写得很好并且含义丰富。此类数据被视为具有自然注释的 Web 资源（Sun，2011）。如果我们可以从这些自然注释的 Web 资源中挖掘这些高质量的数据，则对那些因缺乏数据而受阻的研究将是有益的。

在自然语言处理（NLP）社区中，自动文本摘要是一项艰巨的任务。一个好的摘要系统应该能够理解全文并重新组织信息，以产生连贯，内容丰富且简短的摘要，这些摘要应传达原始文本的重要信息（Hovy and Lin，1998）（Martins，2007）。大多数传统的摘要总结方法将分为两个流程（Bing et al，2015）。首先，通过使用无监督方法或语言知识从原始文本中提取关键文本元素。然后，通过使用语义信息或自然语言生成技术，对提取的不明确的成分进行重写或改写，以提供原始文本的简洁摘要。尽管已经进行了广泛的研究，但摘要概述的语言质量仍然远远不能令人满意。最近，深度学习方法显示了潜在的能力，它可以利用 GPU 计算，从大规模数据中学习表示（Hu et al，2014； Zhou et al，2015）并生成语言（Bahdanau et al，2014； Sutskever et al，2014）。许多研究人员意识到，我们更接近通过使用深度学习方法来做生成式摘要研究。但是，公开可用的高质量大规模汇总数据集仍然非常罕见，而且不容易手动构建。例如，流行的文档摘要数据集 DUC2，TAC3 和 TREC4 仅具有数百种人工书面英文文本摘要。对于中国人来说，这个问题更加严重。在本文中，我们向后退一步，着重于利用新浪微博上的自然注释网络资源来构建 LCSTS，即大规模中文短文本摘要数据集。图 1 显示了《人民日报》发布的一个微博。为了将导入信息快速传达给公众，它还撰写了一个非常有用且简短的新闻摘要（蓝色圆圈）。我们的目标是从这些文本中挖掘出大规模，高质量的短文本摘要数据集。

本文做出以下贡献：

（1）引入了大规模的中文短文本摘要数据集。据我们所知，它是迄今为止最大的一个；

（2）我们将数据集的标准划分分为大规模训练集和人工标记的测试集，这将更易于对相关方法进行基准测试；

（3）探索数据集的属性，并抽样 10,666 个实例，以手动检查和评分数据集的质

量；
（4）我们对数据集执行基于循环神经网络的编码器/解码器方法，以生成摘要并获得有对应的实验结果，可以将其用作后续研究任务的一个基准。

2 相关工作

我们的工作与基于自动注释的 Web 资源的自动文本摘要和自然语言处理的最新工作有关，以下将进行简要介绍。
自 1950 年以来，人们一直在研究某种形式的自动文本摘要。此后，大多数搜索都通过分析文档中单词的组织方式与提取摘要相关内容（Nenkova and McKeown，2011）（Luhn，1998）。由于它需要标记数据集来进行有监督的机器学习方法，并且标记数据集非常密集，因此一些研究集中在无监督的方法上（Mihalcea，2004）。现有数据集的规模通常很小（大多数小于 1000）。例如，DUC2002 数据集包含 567 个文档，并且每个文档都提供了两个 100 字的人工摘要。我们的工作还与标题生成有关，标题生成是要为文本生成一个句子的任务。 Colmenares 等构建了 130 万个用英文编写的金融新闻头条数据集，以生成标题（Colmenares et al，2015）。但是，该数据集不是公开可用的。
Sun（Sun，2011）提出了基于自然注释 Web 资源的自然语言处理方法。带有自然注释的 Web 资源是用户出于交流目的而生成的数据，例如网页，博客和微博客。我们可以通过无意中使用用户生成的标记来从这些原始数据中挖掘知识或有用数据。 Jure 等人追踪了 160 万个主流媒体站点和博客，并在新闻周期中挖掘出一系列新颖而持久的时间模式（Leskovec et al，2009）。 Sepandar 等人使用用户自然注释的模式“我们感觉”和“我感觉”来提取“感觉”句子集合，该句子集合用于收集情感数据。在这项工作中，我们使用自然注释资源来构建大规模的中文短文本摘要数据，以促进文本摘要的研究。

3 数据收集

许多中国流行媒体和组织都在新浪微博上创建了帐户。他们使用自己的帐户发布新闻和信息。这些帐户已在微博上验证，并带有蓝色的“ V”标记。为了确保所抓取的文本的质量，我们仅对经过验证的组织的 weibos 进行爬取，而 weibos 更有可能是整洁，正式且信息丰富的。每个步骤都需要大量的人工干预。数据收集的过程如图 2 所示，总结如下：
1）我们首先收集了 50 个非常受欢迎的组织用户作为种子。它们来自政治，经济，军事，电影，游戏等领域，例如《人民日报》，《经济观察》新闻社，国防部等。
2）然后，我们开始爬取数据，然后过滤这些种子用户并通过使用人工编写的规则对其进行过滤，例如必须对用户进行蓝色验证，关注者的数量超过 100 万，等等。
3）我们使用选择的用户和文本搜寻器来搜寻其 weibos 内容。
4）我们过滤，清理和提取（短文本，摘要）对。大约有 100 条规则用于提取高质量数据对。这些规则由 5 个人通过仔细研究原始文本得出结论。我们还删除了短文本长度太短（少于 80 个字符）且摘要长度超出[10,30]的数据。

4 数据属性

数据集由表 1 所示的三部分组成，文本的长度分布如图 3 所示。

第一部分是 LCSTS 的主要内容，其中包含 2,400,591（短文本，摘要）对。这些对可用于训练用于摘要生成的监督学习模型。
第二部分包含 10,666 人标记的（简短文本，摘要）数据对，其分数范围为 1 到 5，表明该简短文本与相应摘要之间的相关性。“ 1”表示“最不相关”，“ 5”表示“最相关”。为了对此部分进行注释，我们招募了 5 名志愿者，每对仅由一名注释者标记。这些对是从第 I 部分中随机抽取的，用于分析第 I 部分中对的分布。图 4 说明了不同分数的示例。从示例中我们可以看到，得分为 3、4 或 5 的对与相应的摘要非常相关。与原始文本相比，这些摘要内容丰富，简明且简短。我们还可以看到摘要中的许多单词没有出现在原始文本中，这表明我们的数据集与句子压缩数据集的显着差异。得分为 1 或 2 的对的摘要是高度抽象的，并且很难从短文本中得出摘要。它们更可能是标题或评论，而不是摘要。统计数据显示，得分 1 和 2 的百分比不到数据的 20％，可以使用训练有素的分类器对其进行过滤。
第三部分包含 1,106 对。在这部分中，有 3 个注释器标记相同的 2000 文本，我们以共同的分数提取文本。此部分独立于第一部分和第二部分。在这项工作中，我们使用该部分的 3、4 和 5 得分对作为简短文本摘要生成任务的测试集。

5 实验

最近，循环神经网络（RNN）在语音识别（Graves et al，2013），机器翻译（Sutskever et al，2014）和自动对话响应（Shang et al，2015）方面显示出强大的作用 。但是，关于使用深层模型进行自动文本摘要的研究很少。在本节中，我们使用 RNN 作为编码器和解码器来生成短文本摘要。我们使用第一部分作为训练集，使用第三部分的子集（由 3、4 和 5 评分）作为测试集。
有两种方法可以对数据进行预处理：
1）基于字符的方法，我们将汉字作为输入，这将使字符的大小减小到 4,000。
2）基于单词的方法，使用 jieba5 将文本分割成中文单词。词汇限制为 50,000。
我们采用两种深层架构：
1）解码期间不使用上下文信息。我们使用 RNN 作为编码器，并将其最后一个隐藏状态用作解码器的输入，如图 5 所示；
2）在解码过程中使用上下文信息，随后（Bahdanau et al，2014）我们使用编码器将所有隐藏状态的组合作为解码器的输入，如图 6 所示。对于 RNN，我们采用（Chung et al，2015）提出的门控循环单元（GRU）已被证明可与 LSTM 媲美（Chung et al，2014）。两种架构的所有参数（包括嵌入）均被随机初始化，并使用 ADADELTA（Zeiler，2012）更新学习率。训练模型后，在搜索过程中使用集束搜索生成最佳摘要，并且在我们的实验中将束大小设置为 10。
为了进行评估，我们采用了（Lin and Hovy，2003）提出的 ROUGE 度量标准，该度量标准已被证明与人类评估高度相关。使用 ROUGE-1，ROUGE-2 和 ROUGE-L。因为标准的 Rouge 软件包 6 用于评估英语汇总系统，所以我们将中文单词转换为数字 ID 以适应系统。所有模型都在 tesla M2090 GPU 上进行了大约一周的训练。表 2 列出了实验结果。如图 7 所示，RNN 生成的带有上下文的摘要非常接近人类编写的摘要，这表明如果我们将足够的数据提供给 RNN 编码器和解码器，则几乎可以从零开始生成摘要。
结果还显示，在基于字符和单词的输入上，带有上下文的 RNN 优于没有上下文的 RNN。该结果表明，可以将 RNN 编码器的内部隐藏状态组合起来以概括地表

示单词的上下文。而且，基于字符的输入的性能也优于基于单词的输入。如图 8 所示，RNN 通过输入基于字符的短文本在上下文中生成的摘要相对较好，而 RNN 在基于单词的输入中通过上下文生成的摘要包含许多 UNK。这可能归因于该细分可能导致词汇和文本中的许多 UNK，例如人员姓名和组织名称。例如，“愿景光电子”是一个不在基于单词的 RNN 词汇表中的公司名称，RNN 摘要器必须在解码过程中使用 UNK 代替“愿景光电子”。

6 结论与展望

我们构建了一个大规模的中文短文本摘要数据集，并在其上执行了基于 RNN 的方法，取得了不错的结果。这只是针对此任务的深度学习模型的开始，还有很大的改进空间。我们将整个短文本作为一个序列，这可能不是很合理，因为大多数短文本包含几个句子。分层的 RNN（Li et al，2015）是一个可能的方向。稀有单词问题对于生成摘要也非常重要，尤其是当输入是基于单词而不是基于字符的输入时。这在神经生成模型（例如神经翻译机（NTM））中也是一个热门话题（Luong et al，2014），这可以从这项任务中受益。我们还计划通过使用自然注释的 Web 资源来构建大型文档摘要数据集。

论文代码文件

```
#加载 Bert 预训练模型、模型参数、模型权重和词表等文件
pretrained_path = '/content/drive/My Drive/Undergraduate-paper/chinese_L-12_H-768_A-12'
config_path = os.path.join(pretrained_path, 'bert_config.json')
checkpoint_path = os.path.join(pretrained_path, 'bert_model.ckpt')
vocab_path = os.path.join(pretrained_path, 'vocab.txt')
#配置 Bert 模型
model = load_trained_model_from_checkpoint(config_path, checkpoint_path)
# model.summary(line_length=120)
#开始数据预处理
#  文件读取
def read_text(filename):
    file = open(filename, mode='rt', encoding='utf-8')
    text = file.read()
    file.close()
    return text
#  文本分割成句
def to_lines(text):
    sents = text.strip().split('\n')
    sents = [i.split('\t') for i in sents]
    return sents
#  移除标点符号
def remove_punctuation(line):
    #英文标点符号
    # line = line.translate(string.maketrans("",""), string.punctuation)
    line = re.sub(r'[^\w\s]',",line)
    #中文标点符号
    re_punctuation ="[\s+\.\!\/_,$%^*(+\"\']+-|[+——！，。？?、~@#￥%……&*：”；《》.·～「」""（）]"
    line = re.sub(re_punctuation, "", line)
    return line
#清洗数据
def clear(data_path):
    data=read_text(data_path)
    data=to_lines(data)
    for i in range(len(data)):
        s=str(data[i][0])
        s=s.lower()
        data[i]=remove_punctuation(s)
        # print(data[i])
    return data
#构建特征提取函数
def extract(word):
    tokenizer = Tokenizer(token_dict)
    indices, segments = tokenizer.encode(first=word, max_len=512)
    predicts = model.predict([np.array([indices]), np.array([segments])])[0]
    return predicts[1]
#提取 LCSTS 数据集的词向量特征
word_feature={}
for word,i in tqdm(char2id.items()):
    word_feature[word]=extract(word)
embedding_matrix = np.zeros((len(chars)+4, 768))
#根据提取的词向量特征配置词嵌入层
for word,i in char2id.items():
    embedding_matrix[i]=np.asarray(word_feature[word],dtype='float32')
```

```
# 构造数据生成器
def data_generator(content,title):
    X,Y = [],[]
    while True:
        for a in range(len(content)):
            X.append(str2id(content[a]))
            Y.append(str2id(title[a], start_end=True))
            if len(X) == batch_size:
                X = np.array(padding(X))
                Y = np.array(padding(Y))
                yield [X,Y], None
                X,Y = [],[]
# 输出一个词表大小的向量，来标记该词是否在文章出现过
def to_one_hot(x):
    x, x_mask = x
    x = K.cast(x, 'int32')
    x = K.one_hot(x, len(chars)+4)
    x = K.sum(x_mask * x, 1, keepdims=True)
    x = K.cast(K.greater(x, 0.5), 'float32')
    return x
# 构造缩放平移变换层（Scale and shift）
class ScaleShift(Layer):
    def __init__(self, **kwargs):
        super(ScaleShift, self).__init__(**kwargs)
    def build(self, input_shape):
        kernel_shape = (1,) * (len(input_shape)-1) + (input_shape[-1],)
        self.log_scale = self.add_weight(name='log_scale',
                                         shape=kernel_shape,
                                         initializer='zeros')
        self.shift = self.add_weight(name='shift',
                                     shape=kernel_shape,
                                     initializer='zeros')
    def call(self, inputs):
        x_outs = K.exp(self.log_scale) * inputs + self.shift
        return x_outs
# 封装双向 RNN，允许传入 mask，保证对齐
class OurBidirectional(OurLayer):
    def __init__(self, layer, **args):
        super(OurBidirectional, self).__init__(**args)
        self.forward_layer = layer.__class__.from_config(layer.get_config())
        self.backward_layer = layer.__class__.from_config(layer.get_config())
        self.forward_layer.name = 'forward_' + self.forward_layer.name
        self.backward_layer.name = 'backward_' + self.backward_layer.name
    def reverse_sequence(self, x, mask):
        """这里的 mask.shape 是[batch_size, seq_len, 1]
        """
        seq_len = K.round(K.sum(mask, 1)[:, 0])
        seq_len = K.cast(seq_len, 'int32')
        return tf.reverse_sequence(x, seq_len, seq_dim=1)
    def call(self, inputs):
        x, mask = inputs
        x_forward = self.reuse(self.forward_layer, x)
        x_backward = self.reverse_sequence(x, mask)
        x_backward = self.reuse(self.backward_layer, x_backward)
        x_backward = self.reverse_sequence(x_backward, mask)
```

```
            x = K.concatenate([x_forward, x_backward], -1)
            if K.ndim(x) == 3:
                return x * mask
            else:
                return x
        def compute_output_shape(self, input_shape):
            return input_shape[0][:-1] + (self.forward_layer.units * 2,)
# 构造多头注意力机制
class Attention(OurLayer):
        def __init__(self, heads, size_per_head, key_size=None,
                     mask_right=False, **kwargs):
            super(Attention, self).__init__(**kwargs)
            self.heads = heads
            self.size_per_head = size_per_head
            self.out_dim = heads * size_per_head
            self.key_size = key_size if key_size else size_per_head
            self.mask_right = mask_right
        def build(self, input_shape):
            super(Attention, self).build(input_shape)
            self.q_dense = Dense(self.key_size * self.heads, use_bias=False)
            self.k_dense = Dense(self.key_size * self.heads, use_bias=False)
            self.v_dense = Dense(self.out_dim, use_bias=False)
        def mask(self, x, mask, mode='mul'):
            if mask is None:
                return x
            else:
                for _ in range(K.ndim(x) - K.ndim(mask)):
                    mask = K.expand_dims(mask, K.ndim(mask))
                if mode == 'mul':
                    return x * mask
                else:
                    return x - (1 - mask) * 1e10
        def call(self, inputs):
            q, k, v = inputs[:3]
            v_mask, q_mask = None, None
            if len(inputs) > 3:
                v_mask = inputs[3]
                if len(inputs) > 4:
                    q_mask = inputs[4]
            # 线性变换
            qw = self.reuse(self.q_dense, q)
            kw = self.reuse(self.k_dense, k)
            vw = self.reuse(self.v_dense, v)
            # 形状变换
            qw = K.reshape(qw, (-1, K.shape(qw)[1], self.heads, self.key_size))
            kw = K.reshape(kw, (-1, K.shape(kw)[1], self.heads, self.key_size))
            vw = K.reshape(vw, (-1, K.shape(vw)[1], self.heads, self.size_per_head))
            # 维度置换
            qw = K.permute_dimensions(qw, (0, 2, 1, 3))
            kw = K.permute_dimensions(kw, (0, 2, 1, 3))
            vw = K.permute_dimensions(vw, (0, 2, 1, 3))
            # Attention
            a = tf.einsum('ijkl,ijml->ijkm', qw, kw) / self.key_size**0.5
            a = K.permute_dimensions(a, (0, 3, 2, 1))
            a = self.mask(a, v_mask, 'add')
```

```
            a = K.permute_dimensions(a, (0, 3, 2, 1))
            if self.mask_right:
                ones = K.ones_like(a[:1, :1])
                mask = (ones - K.tf.matrix_band_part(ones, -1, 0)) * 1e10
                a = a - mask
            a = K.softmax(a)
            # 完成输出
            o = tf.einsum('ijkl,ijlm->ijkm', a, vw)
            o = K.permute_dimensions(o, (0, 2, 1, 3))
            o = K.reshape(o, (-1, K.shape(o)[1], self.out_dim))
            o = self.mask(o, q_mask, 'mul')
            return o
        def compute_output_shape(self, input_shape):
            return (input_shape[0][0], input_shape[0][1], self.out_dim)
# 搭建 seq2seq 模型
x_in = Input(shape=(None,))
y_in = Input(shape=(None,))
x, y = x_in, y_in
x_mask = Lambda(lambda x: K.cast(K.greater(K.expand_dims(x, 2), 0), 'float32'))(x)
y_mask = Lambda(lambda x: K.cast(K.greater(K.expand_dims(x, 2), 0), 'float32'))(y)
x_one_hot = Lambda(to_one_hot)([x, x_mask])
x_prior = ScaleShift()(x_one_hot) # 学习输出的先验分布（标题的字词很可能在文章出现过）
#Bert 特征提取层
embedding =
Embedding(len(chars)+4,char_size,weights=[embedding_matrix],trainable=False)
x = embedding(x)
y = embedding(y)
# encoder，双层 BiLSTM
# x = LayerNormalization()(x)
x = OurBidirectional(CuDNNLSTM(z_dim // 2, return_sequences=True))([x, x_mask])
# x = LayerNormalization()(x)
x = OurBidirectional(CuDNNLSTM(z_dim // 2, return_sequences=True))([x, x_mask])
x_max = Lambda(seq_maxpool)([x, x_mask])
# decoder，双层单向 LSTM
y = SelfModulatedLayerNormalization(z_dim // 4)([y, x_max])
y = CuDNNLSTM(z_dim, return_sequences=True)(y)
y = SelfModulatedLayerNormalization(z_dim // 4)([y, x_max])
y = CuDNNLSTM(z_dim, return_sequences=True)(y)
y = SelfModulatedLayerNormalization(z_dim // 4)([y, x_max])
# Attention 交互
xy = Attention(8, 16)([y, x, x, x_mask])
xy = Concatenate()([y, xy])
# 输出分类
xy = Dense(char_size)(xy)
xy = LeakyReLU(0.2)(xy)
xy = Dense(len(chars)+4)(xy)
xy = Lambda(lambda x: (x[0]+x[1])/2)([xy, x_prior]) # 与先验结果平均
xy = Activation('softmax')(xy)
# 交叉熵作为 loss，但 mask 掉 padding 部分
cross_entropy = K.sparse_categorical_crossentropy(y_in[:, 1:], xy[:, :-1])
cross_entropy = K.sum(cross_entropy * y_mask[:, 1:, 0]) / K.sum(y_mask[:, 1:, 0])
model = Model([x_in, y_in], xy)
model.add_loss(cross_entropy)
model.compile(optimizer=Adam(1e-3)) # 编译模型
```

```
model.summary()#打印输出模型结构图表
#集束搜索解码:每次只保留 topk 个最优候选结果；如果 topk=1，那么就是贪心搜索
def gen_sent(s, topk=10, maxlen=30):
    s=s.lower()
    s=remove_punctuation(s)
    xid = np.array([str2id(s)] * topk) # 输入转 id
    yid = np.array([[2]] * topk) # 解码均以<start>开头，这里<start>的 id 为 2
    scores = [0] * topk # 候选答案分数
    for i in range(maxlen): # 强制要求输出不超过 maxlen 字
        proba = model.predict([xid, yid])[:, i, 3:] # 直接忽略<padding>、<unk>、<start>
        log_proba = np.log(proba + 1e-6) # 取对数，方便计算
        arg_topk = log_proba.argsort(axis=1)[:,-topk:] # 每一项选出 topk
        _yid = [] # 暂存的候选目标序列
        _scores = [] # 暂存的候选目标序列得分
        if i == 0:
            for j in range(topk):
                _yid.append(list(yid[j]) + [arg_topk[0][j]+3])
                _scores.append(scores[j] + log_proba[0][arg_topk[0][j]])
        else:
            for j in range(topk):
                for k in range(topk): # 遍历 topk*topk 的组合
                    _yid.append(list(yid[j]) + [arg_topk[j][k]+3])
                    _scores.append(scores[j] + log_proba[j][arg_topk[j][k]])
            _arg_topk = np.argsort(_scores)[-topk:] # 从中选出新的 topk
            _yid = [_yid[k] for k in _arg_topk]
            _scores = [_scores[k] for k in _arg_topk]
        yid = np.array(_yid)
        scores = np.array(_scores)
        best_one = np.argmax(scores)
        if yid[best_one][-1] == 3:
            return id2str(yid[best_one])
    # 如果 maxlen 字都找不到<end>，直接返回
    return id2str(yid[np.argmax(scores)])
#开始训练模型
history=model.fit_generator(data_generator(content,title), steps_per_epoch=40000,
epochs=20, verbose=1, validation_data=data_generator(valid_C,valid_T),
validation_steps=200, callbacks=[evaluator] )
#训练好的模型预测生成测试集文本的摘要内容
RR=[]
for i in tqdm(test_C):
  a=gen_sent(i)
  RR.append(a)
#计算生成摘要的 Rouge 评价分数
rouge = RougeCalculator(stopwords=True, lang="zh")
rouge_1, rouge_2, rouge_l =[],[],[]
for i in range(len(test_C)):
  a=rouge.rouge_n(summary=RR[i], references=test_T[i], n=1)
  rouge_1.append(a)
  b=rouge.rouge_n(summary=RR[i], references=test_T[i],   n=2)
  rouge_2.append(b)
  c=rouge.rouge_l(summary=RR[i], references=test_T[i])
  rouge_l.append(c)
#打印输出 Rouge 分数
print(np.mean(rouge_1), np.mean(rouge_2), np.mean(rouge_l))
```