本科生毕业论文（设计）

毕业论文题目：

基于事件结构的多文本摘要生成方法研究

部　　院　系：　人工智能学院

专　　　业：　计算机科学与技术

学　　　号：　201811210138

学 生 姓 名：　彭科钦

指 导 教 师：　王学松

指导教师职称：　高工

指导教师单位：　人工智能学院

年　　月　　日

北京师范大学本科生毕业论文（设计）诚信承诺书

本人郑重声明： 所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

本人签名： 彭科钦 2022 年 5 月 10 日

北京师范大学本科生毕业论文（设计）使用授权书

本人完全了解北京师范大学有关收集、保留和使用毕业论文（设计）的规定，即：本科生毕业论文（设计）工作的知识产权单位属北京师范大学。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电子版，允许毕业论文（设计）被查阅和借阅；学校可以公布毕业论文（设计）的全部或部分内容，可以采用影印、缩印或扫描等复制手段保存、汇编毕业论文（设计）。保密的毕业论文（设计）在解密后遵守此规定。

本论文（是、否）保密论文。

保密论文在_____年_____月解密后适用本授权书。

本人签名： 年 月 日

导师签字： 年 月 日

北京师范大学本科生毕业论文（设计）答辩记录表

<table>
<tr><td>论文题目</td><td colspan="5">基于事件结构的多文本摘要生成方法研究</td></tr>
<tr><td>学生姓名</td><td colspan="2">彭科钦</td><td>学　号</td><td colspan="2">201811210138</td></tr>
<tr><td>所在部院系</td><td colspan="2">人工智能学院</td><td>专　业</td><td colspan="2">计算机科学与技术</td></tr>
<tr><td>指导教师姓名</td><td>王学松</td><td>指导教师职称</td><td>高工</td><td>指导教师单位</td><td>人工智能学院</td></tr>
<tr><td>答辩小组成员</td><td colspan="5">张立保，郭俊奇，王慎玲</td></tr>
<tr><td>答辩过程简要记录</td><td colspan="5">论文内容概述：
本文引入了可以提取跨文档间关系的分层 Transformer 模型，用波束搜索和和长度归一优化生成的摘要质量。对长度归一参数和前馈网络激活函数进行优化实验，实验结果证明了算法的有效性。

关键问题及回答：
Q：文本摘要的自动生成是否考虑了文本的领域？
A：一个思路是针对特定文本的特定领域进行特定的训练，比如专门投入科技新闻的内容进行训练，用来生成科技新闻相关的摘要。另一个思路是加入聚类算法，从多文本当中自动提取不同事件和不同实体，并将提取到的实体加入摘要的目标和标题当中。然而由于时间仓促，没有成功实现。

答辩小组建议：
完善论文格式，规范第三章标题的命名；修改图 4；每一章的标题另起一页；可以加入一些人工评估方法；规范参考文献的格式，摘要中不需要引用文献；大章节下面可以添加简单的概述；表例写在表的上面，将“表格 1”改为“表 1”。

学生对于答辩小组老师建议的回答：
1. 图 4 没有问题，答辩的过程中由于紧张反而说错了这一点。Transformer 的 encoder 块基本结构没有 masked multi-head attention 层，只有在 decoder 块中才会加入掩码机制，后面再跟 encoder 块的基本结构。
2. 人工评估方法附加在论文中的总结与展望，作为后续的扩展方法。
3. 规范第三章标题的命名、修改图 4、标题另起一页、参考文献格式、摘要修改、表格规范等已经修改完成。辛苦老师们评阅！

记录员签字：_____

答辩小组组长签字：_____

20　年　　月　　日</td></tr>
</table>

基于事件结构的多文本摘要生成方法研究

摘 要

多文本摘要是当前计算机自然语言处理领域重要的研究问题，在舆情分析、军事情报分析领域有重要作用。然而目前的多文本摘要领域仍然存在无法学习事件结构的问题，这个问题制约了输出摘要的质量。

本文引入了一个神经网络摘要模型，可以从输入的多个文档中提取信息生成摘要。该模型从之前的分层的 Transformer 体系结构出发，利用预处理工具对中英文分别进行切分和标记后，将文档进行分段，依据标题相关度进行排序筛选。将筛选过后的段落以分层方式进行连接和编码，通过段落间的注意力机制表示跨文档关系，这种注意力机制允许提取文档结构信息，比之前将文本连接起来当成平面序列处理更高效。这样该模型可以学习文本段落之间的潜在关系，以提升输出摘要的质量。最后用 Beam Search 和长度归一优化摘要输出，使生成的语句更加顺畅。

大型摘要数据集 WikiSum 上的实验结果表明，所引入的模型能有效的解决上述问题。

关键词：多文本摘要 分层 Transformer 中文摘要

Research on Multi-Document Summary Generation Method Based on Event Structure

ABSTRACT

Multi-document summarization is an important research issue in the field of computer natural language processing, and plays an important role in public opinion analysis and military intelligence analysis. However, the problem of learning event structure still exists in the field of multi-document summarization, which restricts the quality of output summarizations.

This paper proposes a neural network summarization model, which can extract summarization from multiple input documents. Starting from the previous hierarchical Transformer architecture, the model uses Jieba and Sentencepiece to segment and mark Chinese and English respectively. Then the document is segmented, and paragraphs are sorted and filtered according to title relevance. The paragraphs after screening are linked and encoded in a hierarchical manner, and cross-document relationships are represented through the attention mechanism between paragraphs, which allows the extraction of document structure information and is more efficient than the previous processing of text linking as a flat sequence. In this way, the model can learn the underlying relationships between text paragraphs to improve the quality of the output summary. Finally, Beam Search and length normalization optimization were used to output the summary to make the generated statements more smooth.

Experimental results on WikiSum, a large abstract dataset, show that the proposed model can effectively solve the above problems.

KEY WORDS： Multi-document summarization, Hierarchical Transformer, Chinese Abstract

目录

图目录

表目录

1. 引言

1.1 研究背景与意义

文档信息自动摘要生成，特别是具有事件结构的多文档信息自动摘要生成，在自然语言处理领域，一直是一个重要的研究问题。随着互联网时代发展，信息的生成和传播、使用都带来了巨大的挑战。由于数字化时代兴起的互联网平台将用户从单一信息接收者转变成事件信息生产者的一部分，无数网络平台的用户变成生产新闻的自媒体。数量庞大的自媒体产生了浩如烟海的信息内容。

这些数量庞大的信息内容对于互联网上事件的传播和演化起到了重要的作用，如何利用相关信息，形成事件内容和传播路径的关键拓扑结构，并进一步生成事件内容的摘要描述，显得越来越重要。

本文针对基于事件的文档信息，通过事件的文档关系提取事件的基本结构，并以此促进摘要生成，形成基于事件结构的多文本摘要。相关研究和成果对互联网信息的事件分析、摘要自动生成有重要理论意义，对舆情自动化分析、竞争情报分析、科技进展分析以及军事信息分析等应用领域具有应用价值。

1.2 国内外研究现状

1.2.1 抽取式及生成式文本摘要

文本摘要按照摘要方法可以分为抽取式文本摘要、生成式文本摘要两种。抽取式文本摘要，就是从文档中抽取其中的一句、或者几句话，构成摘要。通过计算文档中句子的得分，代表重要性程度，得分越高句子越重要，依次选取得分高的若干个句子构成摘要，长度取决于压缩率。而生成式摘要方法不是利用原单词或短语组成摘要，而是从原文档中获取“主要思想”后将其表达出来。利用自然语言理解对原文档进行语法语义的分析，然后对融合信息，通过自然语言生成的技术生成新的文本摘要。

相比较而言，抽取式摘要的好处在于：简单实用，不容易完全偏离文章主旨。同时也可能有着不连贯、字数失控、主旨不明确等缺点，甚至可以说，其摘要好坏由原文决定。生成式摘要方法为了传达原文档的主要观点，可以使用原文档中的短语和语句，但总体上来说，需要自己的话来概括表达，可以较好解决对于源文档过于依赖的问题。

1.2.2 单文档及多文档摘要

摘要可以在一个或者多个文档中进行，得到单文本摘要或多文本摘要。虽然单文本摘要执行起来比较简单，但它可能不会对某个事件产生全面的摘要，因为它没有很好地利用相关的或最近的文档。相反，多文本摘要方法能从不同的文档中生成更全面、更准确的摘要，涵盖不同的视角。但由于它试图解决潜在的多样化和冗余信息，因此也相应地更加复杂。对于模型来说，保持复杂输入序列中关键的内容，同时生成连贯、无冗余、事实一致和语法可读的摘要是一项挑战。因此，多文档摘要要求模型具有更强的分析输入文档、识别和合并一致信息的能力。

从方法上来看，单文本摘要有很多抽取式和生成式的自动摘要模型。而多文本摘要采用的抽取式摘要方法大多数是基于图的句子/段落表示进行抽取，主要的区别在于边的权重计算方式。举例来说，基于单词的 TF-IDF 权重的余弦相似度[1]或基于话语关系[2]，以及在最后生成摘要环节中对文本段落进行排名所采用的具体算法。文献中采用了 PageRank 算法的几个变体[2]，以基于整个图递归地计算一段文章的重要性或显著性。Yasunaga 等人提出了该框架的一个神经版本[3]，其中，显著性是利用从句子嵌入和图卷积网络[4]中提取的特征来估计的，应用于表示跨文档链接的关系图。相较而言，多文档的生成式摘要的方法相对较少。一些模型基于句子融合生成摘要[5-7]。尽管神经生成式模型在单文档摘要方面战功累累[8–11]，但是将 Seq2Seq 模型扩展到多文本摘要领域就不是那么简单了。除开训练数据缺乏的难题之外，还有输入文档规模带来的计算难题。之前的解决方法包含模型转移[12,13]和或依赖于重建目标的无监督模型[14,15]。

1.2.3 神经网络多文本摘要模型

近年来，由于神经网络模型的大范围流行，自动文本摘要研究领域热度再起。同时，大规模数据集[17]（包含十万个以上文档-摘要对）推动了神经架构的开发。对于 sequence-to-sequence 模型，一些方法[8,9] 已经做出了从编码源文档到解码生成式摘要的不错的结果。

而多文本摘要，相比于单文本摘要而言，被人关注的少得多。部分原因是适合研究和学习的数据集的缺失。国际上，已经有为文档理解和会议文本分析而制作的高质量多文本摘要数据集（其中摘要是人工写成的），但是规模很小，只有数百个样例，不足以训练出强大的模型。国内的中文摘要数据集主要是来源于新浪微博的 LCSTS[18]、中文短文本摘要数据集，但是主要面向单文本摘要。在 2018 年，Liu 等人提出了一种创建大规模数据集的方法，可以用于多文本摘要的训练，数据量在数十万个样例左右。他们将维基百科的文章（尤其是首段）作为标题的问题摘要，将 Google 查询标题返回的前十个网页内容和维基百科引用的文件作为摘要的源集群。而中文的 wiki 文章也可以作为中文摘要的训练来源。

多文本摘要数据集的推出促进了多文本摘要神经网络模型的更新。2019 年，Liu 等人提出了一种构建大规模摘要数据集的方法和一个两步模型[16]，该模型首先从源文档中提取显著信息，然后使用仅解码器的体系结构(可以处理很长的序列)生成摘要。

1.3 本文工作

在本文中，我们参照 Liu[19]，建立一个基于 Transformer 架构可以有效处理多个输入文档并提取摘要的神经网络模型，让其用分层的方式处理多个文档的编码问题。这个模型通过一种注意力机制来表示跨文档关系，允许跨多个文档共享信息，而不是简单地连接文本并将它们作为平面序列提供给模型。

本文的主要工作如下：

（1）在源文档集预处理阶段，本文使用长短期记忆网络，依据与标题的相似度进行排序并筛选，得到最符合文章主旨的段落集合，再将它们送入模型中进行训练。

（2）在编码阶段，本文使用分层的 Transformer 模型对段落内信息和段落之间的信息进行提取，以收集刻画文档集所表示的事件结构。并用图结构表示段落之间的联系。

（3）在摘要生成阶段，本文使用 Beam Search 方法和长度归一方法对生成的摘要进行优化，通过提升生成摘要的得分来提升输出的摘要质量。

整个模型的处理框架如图 1 所示：

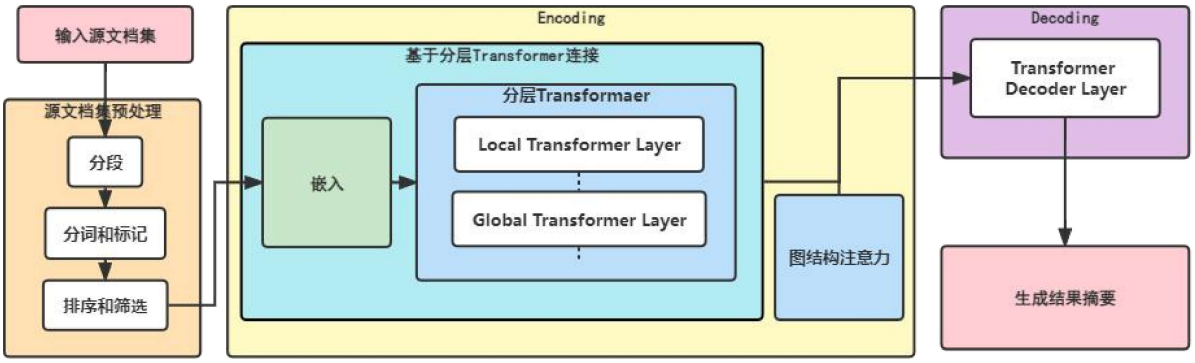

图 1 基于 Transformer 文本摘要主要工作流程

在源文档集预处理阶段，我们进行分段、分词和标记以及基于长短期记忆网络（LSTM）按照与标题的相似度对段落进行打分并排序，筛选出合适数量的段落送入模型进行训练。在编码阶段，我们使用分层的 Transformer 模型学习段落内和段落之间的上下文关系，刻画文章的结构。接着我们使用图结构来表示段落的关系，最后我们使用 Beam Search 和长度惩罚来优化输出的结果。

1.4 篇章结构

本文章节安排如下：

第一章介绍了选题背景、国内外研究现状，以及论文主要工作内容，便于对本论文的主要工作有总体了解。

第二章简要梳理了本文所使用到的理论基础，其中包括多文档摘要的大致流程、分词和标记、分段、排序和筛选、长短时记忆网络基本模型和 Transformer 基本模型。

第三章主要描述了本文所涉及的各个模型的细节和主要工作，其中包括基于长短期记忆网络模型、分层 Transformer 模型、事件结构的图表示方法、Transformer 的解码器模型和摘要优化方法。

第四章详细描述了实验设计和结果分析方法。实验设计包括选择的数据集、测试环境的配置、段落排序模型的参数配、摘要评价指标以及部分参数优化实验。结果分析包括实验结果展示分析和与其他模型的对比分析。

第五章总结了本文的主要结论以及目前模型的不足，以及对未来改进的思考。

1.5 本章小结

本章介绍了相关研究背景，和国内外研究现状，简述了本文的主要工作，最后对本文的篇章结构做出说明。

2. 相关理论基础

2.1 多文档摘要主要流程

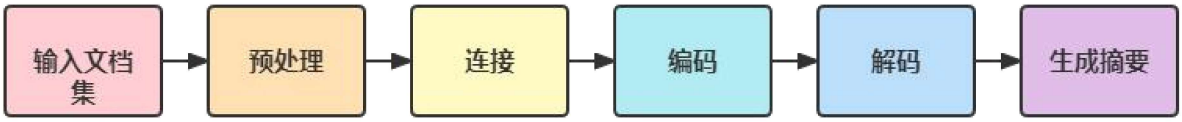

图 2 多文本摘要的处理框架

我们用图 2 表示处理多文本摘要的框架。

输入源文档集后，提取多文档摘要的第一步是对输入文档进行预处理，例如分出段落，对段落内容进行分词并且标记，并选择进入模型的输入等。第二步，对处理后的文档进行连接，多文档摘要生成模型尤其需要选择合适的连接方法来捕获跨文档关系。第三步，选择合适的基于深度学习的模型对输入进行编码，生成表示文本的向量。第四步是针对编码器生成的文本向量进行解码。最后，根据解码出的编码来将文档转换为简明且信息丰富的摘要。

2.2 分词和标记

词（word）是能独立使用的最小的音义结合体，是能够独立表达语义和内容的最基本单元。英语中的词之间往往以空格区分，但是中文的文章中不包含明显的分隔符，所以为了后续的操作，通常需要对不含分隔符的语言进行分词操作。事实上，在英文的句子中，由于词语往往具有复杂的词形变化，一般需要采取基于统计的无监督子词切分的方法，来规避数据稀疏问题和由词表太大而降低处理速度的问题。

对于以词语和意群为语言单位的文章，我们首先需要将其语言单位变换为统一的标记，语言单位可以是字、词或者短语，这取决于分词器和标记器的设置。

对英文的语料来说， google 开源的自然语言处理工具包 SentencePiece 是一个良好的分词和标记工具。它具备相当高的标记效率，并且可以基于统计来切分和标记经常出现的粒度较大的词组。但当视线集中到中文的文本任务时，由于没有天然的空格作为词语切分，Sentencepiece 切分的粒度往往局限在单个字符。针对这一点，我们使用 jieba 作为中文分词的参考，用 jieba 对训练语料和输入进行切分，让 sentencepiece 对切分好的语料进行标记，这样即能利用 jieba 切分的准确性和 sentencepiece 的高效性。

2.3 分段、排序和筛选

对于端到端的模型来说，多文本摘要的主要难点在于输入的规模不利于模型训练。由于源文档可能非常长，所以通过换行符将它们分成多个段落。

分段解决了单篇文章输入过长的问题，但模型仍然面临着段落数太多的问题。所以我们并没有直接将段落输入模型，而是先对段落进行排序，在 L 个段落中通过排序选出最好的L'个段落，再送入模型进行训练，以解决输入规模过大带来的问题。

2.4 长短时记忆网络基本模型

循环神经网络（RNN）是一种用于处理序列数据的神经网络，网络的隐藏层输出又作为其自身的输入。相较于一般的神经网络，RNN 处理序列变化的数据的能力更强。比如根据上下文内容变化某个单词的含义会有变化，RNN 就能很好地解决这个问题。

长短期记忆网络（LSTM）是 RNN 的特殊版，主要是为了解决长序列训练中的梯度消失和梯度爆炸的问题。相比普通的 RNN，LSTM 在更长的序列中表现更好。

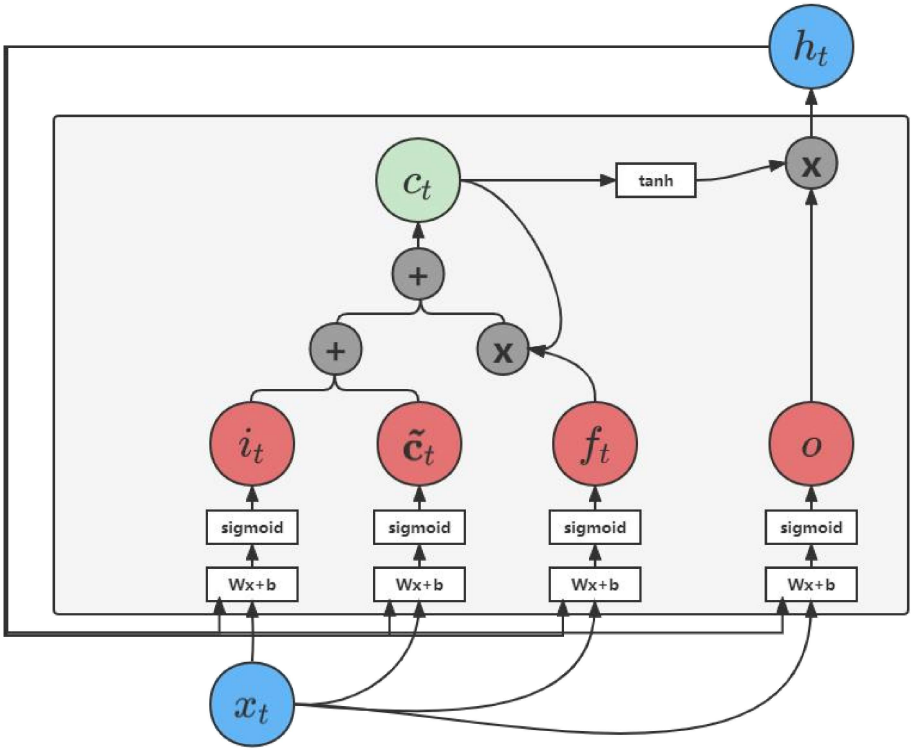

图 3　LSTM 神经元结构

图 3 [37]，表示了 LSTM 的神经元结构。主要包含 5 个基本组件，分别是：

- 单元状态（cell state），储存短期记忆和长期记忆的储存单元；
- 隐藏状态(hidden state) ，根据当前输入、先前的隐藏状态和当前单元状态信息计算得到的输出状态信息；
- 输入门（input gate），控制从当前输入流到单元状态的信息量；
- 遗忘门（forget gate），控制当前输入和先前单元状态中有多少信息流入当前单元状态；
- 输出门（output gate），控制有多少信息从当前单元状态进入隐藏状态

计算公式为：

$$\mathbf{f}_t = \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)\#(2-1)$$

$$\mathbf{o}_t = \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)\#(2-2)$$

$$\tilde{\mathbf{c}}_t = \tan\mathrm{h}\,(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c)\,\#(2-3)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t\#(2-4)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tan\mathrm{h}\,(\mathbf{c}_t)\,\#(2-5)$$

2.5 Transformer 基本模型

2.5.1 Transformer 整体结构

受注意力机制启发，当要表示序列中某一时刻的状态时，可以通过该状态与其他时刻状态之间的注意力来计算。而这种计算方式不受两个时刻之间的距离较远的限制。Transformer 模型就是其中的佼佼者，它融入了位置信息，将输入的不同映射后的结果来承担 Query、Key 和 Value 角色，并且使用多层注意力机制和多头注意力机制使得抽取不同类型的特征成为可能。Transformer 整体结构包含 encoder 部分和 decoder 部分，如图 4 所示[21]。

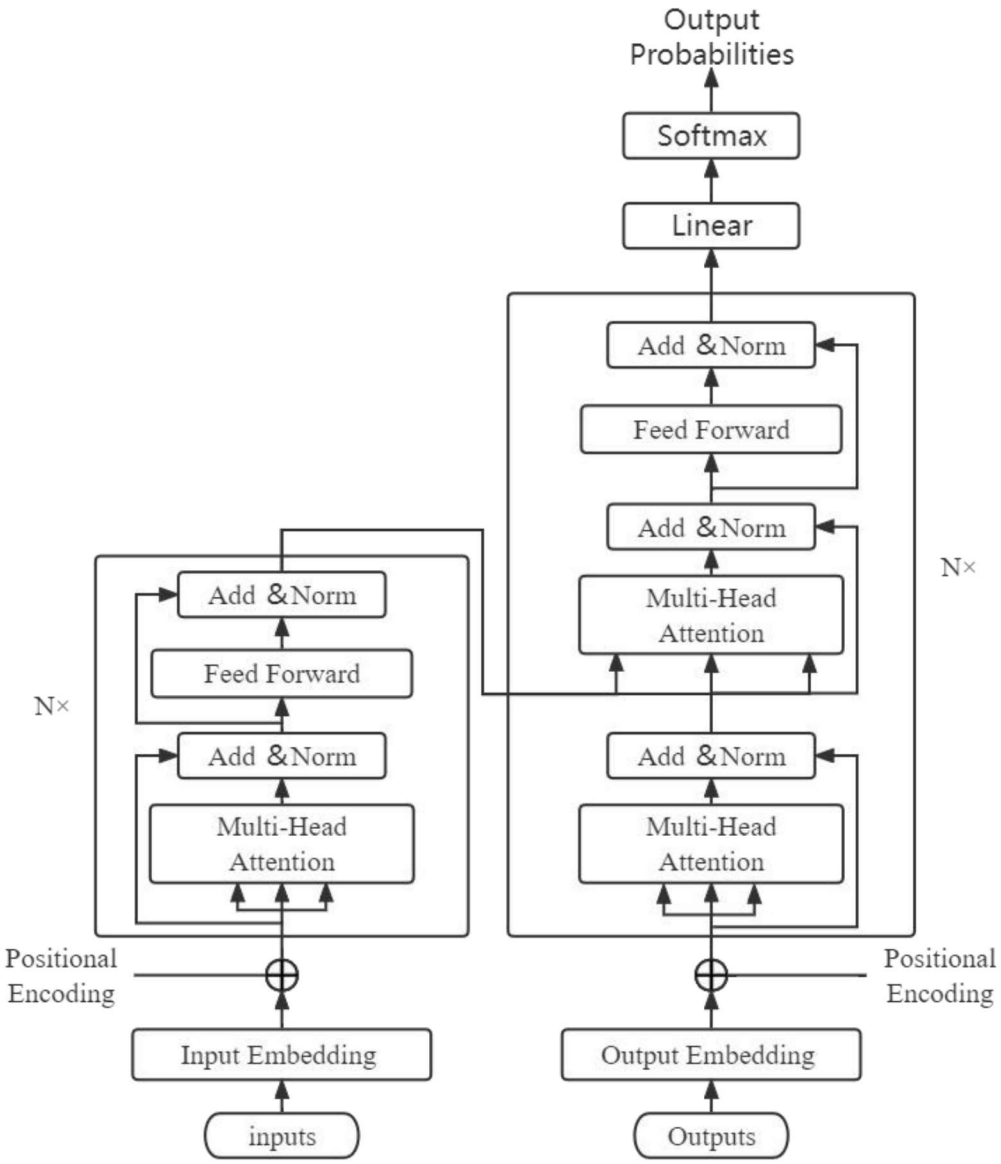

图 4　Transformer 整体结构

2.5.2　融入位置信息

输入 token 首先由词向量表示。令$w_{ij} \in \mathbb{R}^d$表示赋给t_{ij}的向量。由于 Transformer 是非循环模型，无法捕获 token 之间的位置关系，所以还需要将特殊的位置向量 pe_{ij}赋给t_{ij}，来表明 token 在输入中的位置。为了计算位置向量，我们学习 Vaswani 等人[21]，使用不同频率的正弦和余弦函数。序列中第 p 个元素的向量e_p为:

$$e_p[i] = \sin\left(p/10000^{2i/d}\right) \#(2-6)$$

$$e_p[2i+1] = \cos\left(p/10000^{2i/d}\right) \#(2-7)$$

其中$e_p[i]$表示嵌入向量的第 i 维。因为位置编码的每一个维度都对应一个正弦信号，对于任何一个固定的偏移量 o，e_{p+o}都可以表示为e_p的线性函数，这使得模型能够区分输入元素的相对位置。

在多文档摘要中，token t_{ij}有两个位置需要考虑，即 i(段落的排名)和 j(标记在段落中的位置)。位置向量$pe_{ij} \in \mathbb{R}^d$通过拼接表示两个位置，将其加入到词向量w_{ij}中，得到最终的输入向量x_{ij}^0:

$$pe_{ij} = [e_i; e_j] \#(2-8)$$

$$x_{ij}^0 = w_{ij} + pe_{ij} \#(2-9)$$

2.5.3 Encoder 块基本结构

图 4 左边部分表示了 Transformer encoder 块的基本结构，它由两个子层组成，分别为多头自注意力机制和前馈神经网络层公式如下：

$$h = \mathrm{Norm}\left(x^{l-1} + \mathrm{MultiHeadAttention}\left(x^{l-1}\right)\right) \#(2-10)$$

$$x^l = \mathrm{Norm}\left(h + \mathrm{FeedForwardNetwork}\left(h\right)\right) \#(2-11)$$

其中 Norm 为 Ba 等提出的层归一化[22]; MultiHeadAttention 是 Vaswani 等人引入的多头自注意力机制，它设置多组映射矩阵，将产生的多个输出向量拼接。拼接后的输出向量经过线性映射回到对应维度向量。这就允许每个输入向量关注具有不同注意分布的其他不同向量; FeedForwardNetwork 是一个以 ReLU 为隐藏激活函数的两层前馈网络。

2.5.4 Decoder 块基本结构

Vaswani 等人提出的 Transformer decoder 层设计，用于从上游 encoder 层输出的文本向量中解码出 token，由三个子层组成，分别为遮盖的多头自注意力、多头注意力和前馈神经网络层。如图 4 右半部分所示。

2.5.4.1 遮盖的多头自注意力

第一个子层是遮盖的多头自注意力机制，为了防止在训练过程中的 decoder“偷看”要预测的内容。可以视为模型考虑之前的输出来计算当前的输出。

这里的 mask 是一个下三角矩阵，对角线以及对角线左下都是 1，其余都是 0。

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \#(2-12)\#$$

当 mask 不为空，计算 attention 时需要将矩阵中的 0 替换为$-1e9$，替换后，此位置的值经过 Softmax 计算后近似为 0。这样就可以将未来的信息掩盖。多头自注意力的计算和 Transformer encoder 中的一致。

2.5.4.2 多头自注意力

第二个子层是多头自注意力机制，结构 encoder 中的多头注意力机制类似。输入 encoder 输出的向量和上一个输出的结果。Decoder 在这里用 encoder 的输出信息来计算当前应该输出什么。与 Encoder 中的计算不同在于，此时的 Q、K、V 向量中，只有 Q 是 Decoder 的属性，而 K、V 来源于 Encoder。通过这个方法，Decoder 可以捕捉 Encoder 的输出信息。

2.6 本章小结

本章主要综述了本文多文本摘要模型的相关理论基础。主要包括多文档摘要的大致流程、分词和标记、分段排序和筛选、长短时记忆网络基本模型和 Transformer 基本模型。接下来的章节将会对本文的多文本摘要模型的各部分详细实现细节加以介绍。

3. 基于事件结构的多文本摘要模型设计

3.1 基于长短期记忆网络的排序和筛选

输入标题 $T = \{w_{t1}, ..., w_{tm}\}$和一段文本 $P = \{w_{p1}, ..., w_{pn}\}$，通过 LSTM 来编码获得标题和段落中每个词向量：

$$\{u_{t1}, \cdots, u_{tm}\} = \text{lstm}_t(\{w_{t1}, \cdots, w_{tm}\}) \#(3-1)$$

$$\{u_{p1}, \cdots, u_{pn}\} = \text{lstm}_p(\{w_{p1}, \cdots, w_{pn}\}) \#(3-2)$$

其中w_{ti}, w_{pi}是 T 和 P 中的 token 的词向量，而u_{ti}, u_{pi}是应用 LSTM 后每个 token 的更新向量。

然后对标题的 LSTM 表示进行压缩，在标题向量上使用一个最大池化操作，来获得一个固定长度的表示$\hat{u}_t$：

$$\hat{u}_t = \text{maxpool}(\{u_{t1}, \cdots, u_{tm}\}) \#(3-3)$$

将标题$\hat{u}_t$与段落的每一个 token 的向量u_{pi}进行拼接映射来获得一个融合了标题和文段内容的表示：

$$p_i = \tanh\left(W_1\left(\left[u_{pi}; \hat{u}_t\right]\right)\right) \#(3-4)$$

在通过压缩和非线性映射得到匹配分数 s

$$\hat{p} = \text{maxpool}(\{p_1, \cdots, p_n\}) \#(3-5)$$

$$s = \text{sigmoid}\left(W_2(\hat{p})\right) \#(3-6)$$

其中，s 为分数，表征了输入文段中的内容与目标标题之间的内容相似程度，高低表示是否要用段落 P 来生成摘要。

然后对所有的段落$\{P_1, ...P_L\}$都依照上述步骤计算相似度分数$\{s_1, ...s_L\}$，最后通过对相似度分数进行排序来选择最优的 TopK 个段落用于之后的生成。

模型的训练目标是最小化每个段落计算得到的相似度分数s_i和真实分数y_i之间的交叉熵损失，y_i是对段落P_i和黄金摘要 D 之间的 $\text{ROUGE}-2$ 召回率分数，表示段落与黄金摘要的相关性。在测试中，根据模型预测分数对输入段落进行排序，生成序列$R_1, ...R_L$。前L_0段$R_1, ...R_{L0}$选为第二生成阶段的输入。

3.2 分层 Transformer 文本编码模型

该分层模型由多个本地和全局的 Transformer 层构成，这些层可以自由叠加，而每一层的输出都作为下一层的输入。其中本地 Transformer 层用于学习段落内的上下文关系，而全局的 Transformer 层用于学习段落之间的上下文关系。

3.2.1 多文本摘要的输入

对于多文本摘要而言，作为输入的下标必须包含段落信息和位置信息。令t_{ij}表示表示第 i 个段落中的第 j 个 token; 该模型接受每个 token 向量x_{ij}^{0}作为起始输入。对于第 l 层 transformer，输入为x_{ij}^{l-1},输出为x_{ij}^{l}。

3.2.2 用于学习段落内上下文信息的本地层

本地 Transformer 层主要用于为段落内的 token 编码上下文信息。结构相同于 Transformer 基本结构，如图 5 所示。

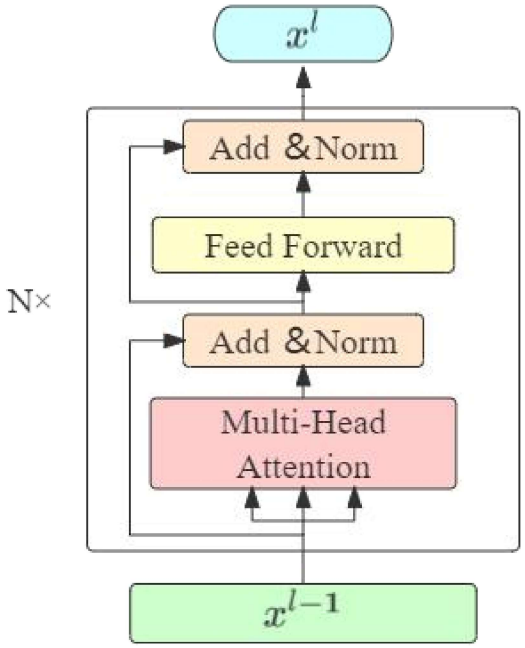

图 5 Transformer encoder 块基本结构

本地 Transformer 层的输入是每一个标记的词向量，经过多头自注意力机制（图中 Multi-Head Attention）后计算得到每个词向量与段落内其他词向量的关系，然后通过前馈

神经网络（图中 Feed Forward）向前传播。

3.2.3 用于段落结构学习的全局层

全局 Transformer 层用于跨多个段落交换信息，学习段落间的上下文信息，刻画段落间结构。如图 6 所示，不同的颜色表示不同的注意力头。

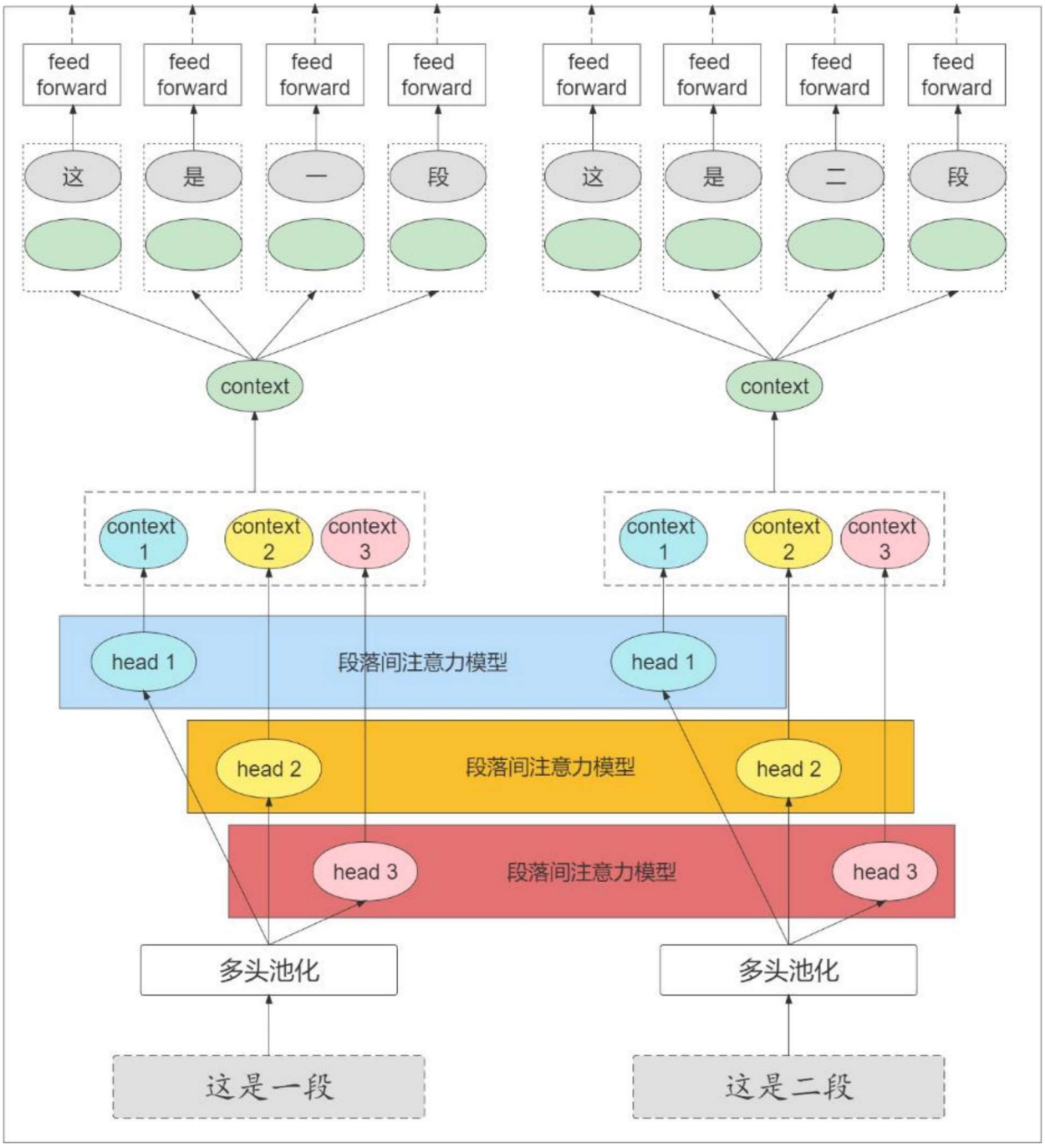

图 6 全局 Transformer 层

3.2.3.1 多头池化方法

我们首先对每个段落应用一个多头池化操作。不同的头将以不同的注意力权重对段落进行编码。然后对于每个头，通过段落间的自注意力机制，每个段落就可以从其他段落交互信息，生成这个段落的上下文向量，从而从段落间捕捉上下文信息：

令 $x_{ij}^{l-1} \in \mathbb{R}^d$ 作为 token $\mathrm{t_{ij}}$ 的上一层 Transformer 层的输出向量，同时作为当前 transformer 层的输入。对于每一段$\mathrm{R_i}$，都有头$z \in \{1, \cdots, n_{head}\}$，我们首先计算注意力分数$\mathrm{a_{ij}^z}$和 value 分数$b_{ij}^z$。然后，对于每个头在段落内所有的 token，都计算一个头在 token 中的注意力概率分布$\mathrm{\hat{a}_{ij}^z}$:

$$a_{ij}^z = W_a^z x_{ij}^{l-1} \#(3-7)$$

$$b_{ij}^z = W_b^z x_{ij}^{l-1} \#(3-8)$$

$$\hat{a}_{ij}^z = \exp\left(a_{ij}^z\right) / \sum_{j=1}^{n} \exp\left(a_{ij}^z\right) \#(3-9)$$

其中 $\mathrm{W_a^z \in R^{1*d}}$和$\mathrm{W_b^z \in R^{d_{head}*d}}$ 是权重. $\mathrm{d_{head} = d/n_{head}}$是注意力头的维度，n 是$\mathrm{R_i}$中 token 的数量。

接下来，我们通过线性变换和层归一化来应用加权求和，以得到该段落的每个头对应的向量表示 $\mathrm{head_i^z}$：

$$\mathrm{head}_i^z = \mathrm{LayerNor\,m}\left(W_c^z \sum_{j=1}^{n} (\widehat{a})_{ij}^z b_{ij}^z\right) \#(3-10)$$

其中$W_c^z \in \mathbb{R}^{d_{\mathrm{head}}*d_{\mathrm{head}}}$是权重。

该模型可以灵活地结合多个头，每个段落有多个注意力分布，从而聚焦于输入的不同观点。

3.2.3.2 段落间注意力模型

获得各个段落的多头表示后，我们使用段落间注意力机制对各个头中不同段落的向量进行交互。与自注意力机制类似，段落间注意力通过计算注意分布来让每一段注意到其他段落:

$$q_i^z = W_q^z h_{ead}^z \#(3-11)$$

$$k_i^z = W_k^z head_i^z \#(3-12)$$

$$v_i^z = W_v^z head_i^z \#(3-13)$$

$$\text{context } t_i^z = \sum_{i=1}^{m} \frac{\exp\left(q_i^{zT} k_{i'}^z\right)}{\sum_{o=1}^{m} \exp\left(q_i^{zT} k_o^z\right)} v_{i'}^z \#(3-14)$$

其中，q_i^z, k_i^z，$v_i^z \in \mathbb{R}^{d_{\text{head}} * d_{\text{head}}}$ 是由head_i^z线性变换的 query、key 和 value 向量[21]; $\text{context } t_i^z \in \mathbb{R}^{d_{\text{head}}}$ 表示通过对所有段落的自注意力操作生成的上下文向量。m 是输入段落的数目。

3.2.3.3 前馈神经网络

接下来，通过一个前馈神经网络，我们用上下文信息来更新 token 表示。我们首先通过连接所有上下文向量来融合各个头的上下文表示，并对权重$W_c \in \mathbb{R}^{d*d}$应用一个线性变换:

$$c_i = W_c\left[\text{context}_i^1; \cdots; \text{context}_i^{n_{\text{head}}}\right] \#(3-15)$$

然后我们用c_i来更新每个输入段落的每个 token 的表示。将c_i加到每个输入 token 向量x_{ij}^{l-1}中，并将其输入到一个以 ReLU 为激活函数，顶部为高速层归一化的两层前馈网络中:

$$g_{ij} = W_{o2}\text{ReLU}\left(W_{o1}\left(x_{ij}^{l-1} + c_i\right)\right) \#(3-16)$$

$$x_{ij}^l = \text{LayerNorm}\left(g_{ij} + x_{ij}^{l-1}\right) \#(3-17)$$

其中$W_{o1} \in \mathbb{R}^{d_{ff}*d}$ 和 $W_{o2} \in \mathbb{R}^{d*d_{ff}}$是权重，$d_{ff}$是后面隐藏的前馈层的大小。

这样，经过 local 和 global 的 attention 就能让每个字符都经过一次分层的上下文更新，R_i段中的每个 token 都可以以一种分层的、高效的方式从其他段落收集信息。

3.3 事件结构的图表示

段落间注意力机制可以看作是让模型学习输入段落集合绘制一张隐式图结构表示。尽管之前的工作已经表明，相似的隐式结构表示对下游的 NLP 任务是有益的[23-26]，在多文档摘要方面的许多工作都利用了显式图表示，每个图表示都关注摘要任务的不同方面。分层 transformer 的一个优点在于可以很容易地将图合并到模型外部，以生成更好的摘要。

我们构造两张图，第一张图旨在捕捉词汇关系：节点为段落，边的权值是段落与段落

之间的余弦 tf-idf 相似度分数。如图 7 左边所示。第二张图旨在捕捉话语关系[2]：它在段落上建立一个近似语篇图(ADG) [3]，以段落为结点，段落之间的边是通过计算段落与段落之间共现词的数量和转折词的数量来绘制的。如图 7 右边所示。

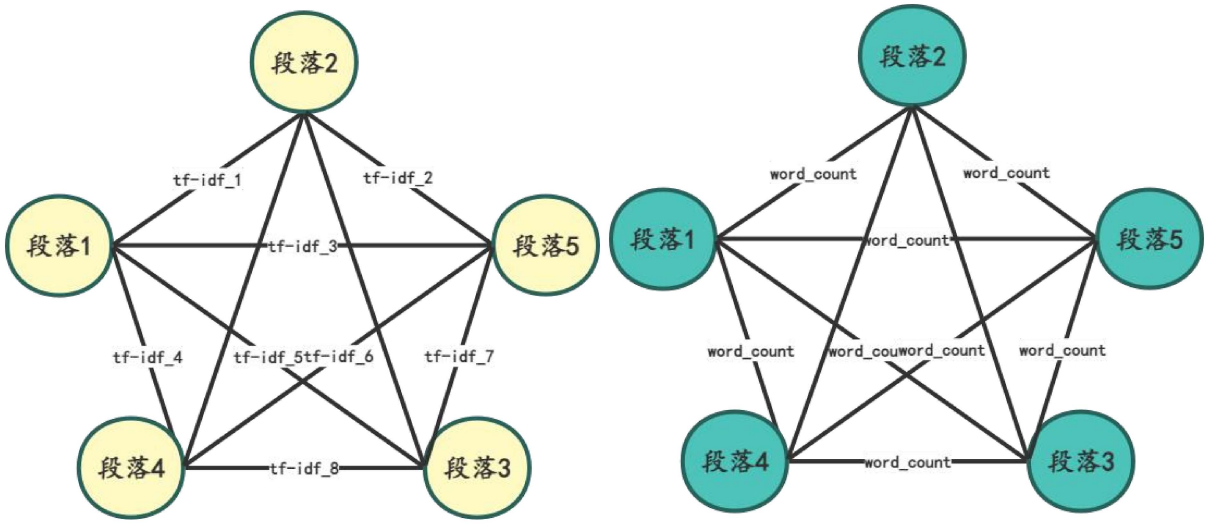

图 7 词汇关系图（左）和话语关系图（右）

这样的图引入到模型中表示为一个矩阵 G，其中$G_{ii'}$是连接段落 i 和 i'的边的权值。然后，在计算 context 向量时替换掉每个头学习到的 q、k、v。计算这个头部上下文向量的公式修改为:

$$\text{context}_i^{z'} = \sum_{i'=1}^{m} \frac{G_{ii'}}{\sum_{o=1}^{m} G_{io}} v_{i'}^{z'} \#(3-18)$$

3.4 Transformer 解码器模型

我们的解码器遵循基本的 Transformer decoder 模块，主要用于为文本向量解码出 token。与普通 Transformer decoder 层相同，由三个子层组成，包括遮盖的多头自注意力、多头注意力机制和前馈神经网络层，如图 8 所示。

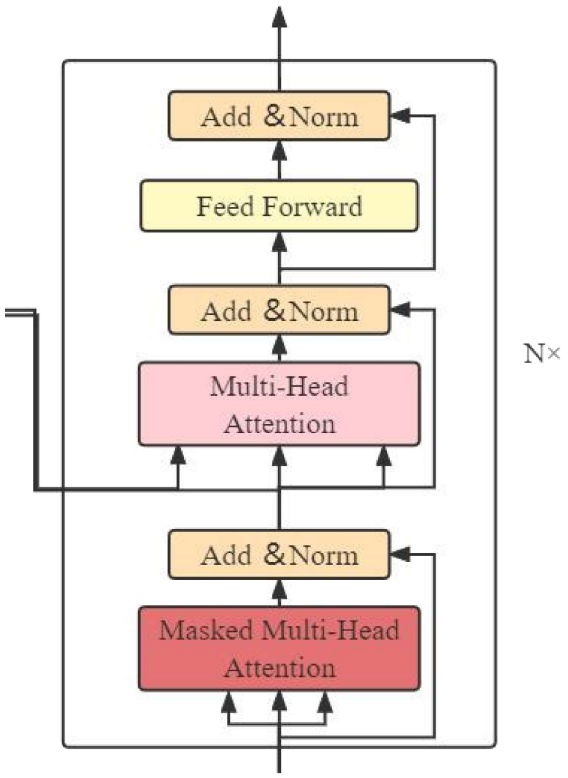

图 8 Transformer decoder 块基本结构

解码器的输入是编码器的输出以及上一个预测出的词语，经过遮盖的多头自注意力机制（图中 Masked Multi-Head Attention）后输入多头自注意力机制（图中 Multi-Head Attention），得到的结果通过前馈神经网络（图中 Feed Forward）向前传播。

3.5 摘要优化方法

3.5.1 优化摘要生成的 Beam search 方法

在本文中，我们使用 Beam Search 来使得生成的文章更加通顺。

当输出的序列为 $Y = (y_1, y_2, ..., y_m)$，概率分布建模如下：

$$P(Y|X) = P(y_1|X)P(y_2|y_1, X)...P(y_m|y_{1...m-1}|X)\#(3-19)$$

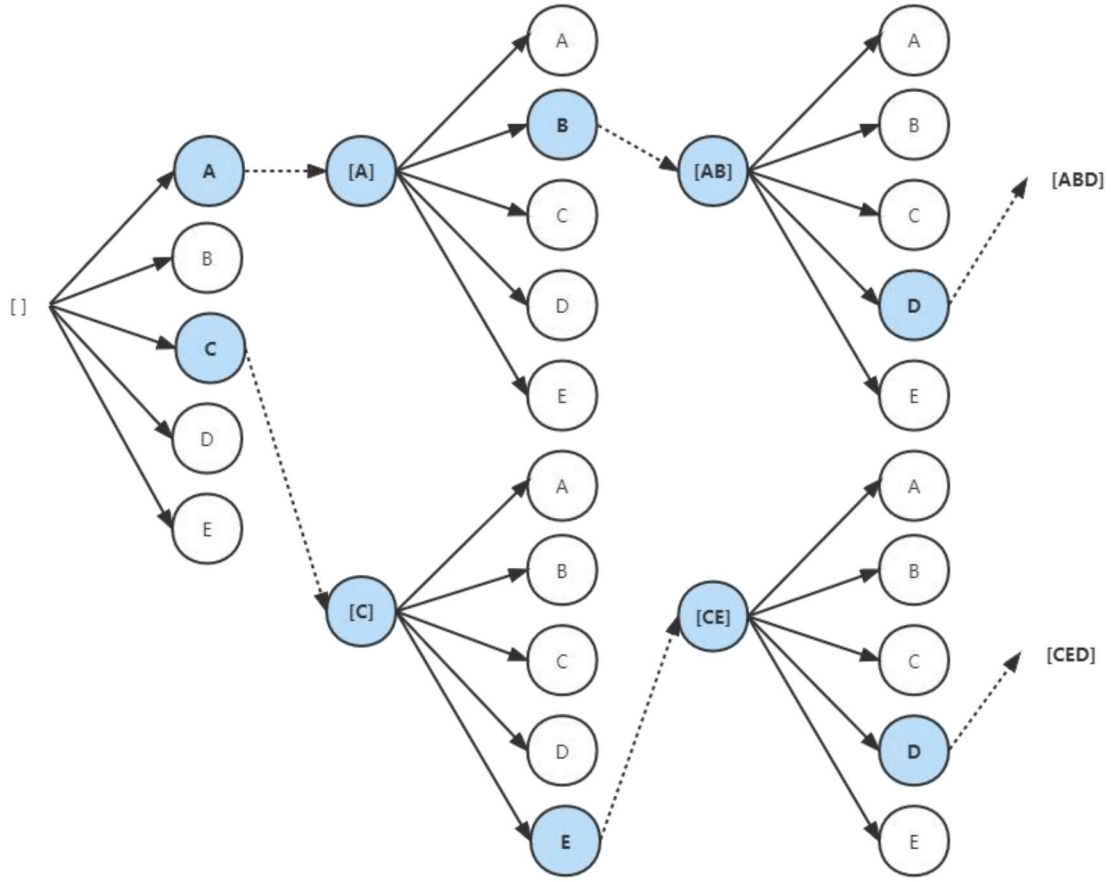

图 9 Beam search 原理图（beam size = 2）

在 decode 时对于不同的选词方案，由于序列太长，穷举会造成过大的开销，而贪心算法不一定能找到全局最优解。而折中方案就是 Beam Search，如图 9 所示，每一步解码仅仅保留前 K 个可能结果，带入下一步解码得到K^2个 结果最后保留概率乘积最大的前 K 个结果。最终选择整体概率最高的序列作为输出。

3.5.2 控制文本生成长度的归一方法

由于随着输出序列长度增加，整体概率会不断降低，这不利于生成较长的句子。所以我们遵循 Wu[27]，加上一个评分函数，根据覆盖率归一来对输出进行评分，评分函数 $s(Y,X)$ 为

$$s(Y,X) = \log\left(P(Y|X)\right)/lp(Y)\#(3-20)$$

$$lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}}\#(3-21)$$

式子中p_{ij}为第 j 个目标词y_j对第 i 个源词x_i的注意概率。参数 α 控制着长度归一化的强

度。

这样，根据整体概率和评分进行选择，我们就能生成最合适的摘要。

3.6 本章小结

本章详细介绍了本文多文本摘要模型的细节处理，包括基于长短记忆网络的段落排序方法、用于学习文本之间和段落内的分层 Transformer 模型、对事件结构的图表示方法以及 Transformer 解码模型以及摘要优化的方法。下一个章节将会详细描述本文所设计的实验以及结果分析。

4. 实验设计与结果分析

4.1 实验设计

4.1.1 数据集选择

我们使用 brightmart 提供的 wiki2019zh[28]数据集，包含 104 万个词条(1,043,224 条;)同时模仿 Liu 等人提供的脚本，爬取参考文段。我们筛选了 10 段正文以上的文章作为训练来源，共有 58000 篇左右。每个输入有（除摘要之外）十段以上的维基正文，每段平均 70.1 个 token，目标摘要平均长度为 139.4 个 token。我们将数据集分割为 50000 个训练集，4000 个验证集和 4000 个测试集。

在排序和生成总结的阶段，用 sentencepiece 对原段落和目标摘要进行编码和解码，词汇表由 32000 个词组成，源和目标摘要通用。

4.1.2 测试环境配置

在所有的生成式模型中，我们在所有线性层之前应用 dropout(概率为 0.1~~);~~);也使用平滑因子 0.1 的标签平滑[31]。训练是传统的最大似然估计的序列到序列的方式。优化器为 Adam，学习速率为 2， $\beta 1 = 0.9$， $\beta 2 = 0.998$；我们还在前 8000 步中应用了学习速率热身，并衰减。所有基于 Transformer 的型号都有 256 个隐藏单元;所有层的前馈隐藏大小为 1024。所有模型都在 4 张 GPU (NVIDIA TITAN)上训练了 10 万步。使用梯度积累来是所有模型训练时间近似一致。根据验证集的性能，选择 5 个最佳检查点，并报告测试集的平均结果。

解码过程中，我们用 beam search（beam size = 5）和 length penalty（$\alpha = 0.4$）~~,~~，直到序列结束的 token。

4.1.3 段落排序与筛选

为了训练回归模型，我们计算了每个段落相对于目标摘要的 ROUGE-2 recall[29]，并以此作为基准得分。Lstm 的参数设置为：隐藏层大小：256； dropout（在所有线性层前使用）概率：0.2；经查阅资料，Adagrad[30]学习速率设置为：0.15。

我们将我们的排序模型与 Li[16]等人提出的方法进行比较，Liu 等人使用每个段落与文

章标题之间的 tf-idf 余弦相似度对输入段落进行排序。我们分别从排序器和基于相似度的方法生成的有序段落集中选取前L'段。我们将这些段落连接起来，并根据黄金目标文本计算它们的 ROUGE-L recall。结果如表 1 所示。我们可以看到，我们的排序器有效地提取了相关段落，并为下游的摘要任务生成了更有信息的输入。

表 1 基于 tf-idf 相似度和 LSTM 回归得到的段落的召回率

排序方法	评价指标：ROUGE-L 召回率			
	$l' = 5$	$l' = 10$	$l' = 20$	$l' = 40$
Tf-idf 相似度	24.86	32.43	40.87	49.49
LSTM 回归（本文方法）	39.38	46.74	53.84	60.42

4.1.4 摘要评价指标

我们使用 ROUGE[29]来评估摘要质量。ROUGE(Recall-Oriented Understudy for Gisting Evaluation)是一种自动评价摘要的方法，将生成的摘要与参考摘要进行比较，统计二者之间重复的 n 元语法、词序列和词对的数目，来计算摘要质量的分值，以衡量生成的摘要与参考之间的相似度。其中：

$$\text{Recall} = \frac{\text{重复词数}}{\text{原文中总词数}} \#(4-1)$$

$$\text{Precision} = \frac{\text{重复词数}}{\text{生成摘要总词数}} \#(4-2)$$

$$ROUGE-N = \frac{\sum_{S\in \text{Reference Summaries}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S\in \text{Reference Summaries}} \sum_{gram_n \in S} \text{Count}(\text{gram}_n)} \#(4-3)$$

其中，-n 代表 n-gram 的长度，$\text{Count}_{\text{match}}(\text{gram}_n)$表示同时出现在一篇候选摘要(candidate summary)和参考摘要(reference summary)的 n-gram 个数，$\text{Count}(\text{gram}_n)$ 表示参考摘要中 n-gram 的个数。分母即为参考摘要集合中所有 n-gram 的个数。

对于多个参考文档，选择得分最高的参考摘要作为参考。

ROUGE-L 使用 LCS 测量最长的匹配序列，优点是不需要连续匹配，能反映句子级词

语间的顺序关系且不需人工定义 n-gram 的值。

$$R_{lcs} = \frac{LCS(X,Y)}{m} \#(4-4)$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \#(4-5)$$

$$F_{lcs} = \frac{(1+\beta^2 R_{lcs} P_{lcs})}{R_{lcs} + \beta^2 P_{lcs}} \#(4-6)$$

我们使用单词和双词重叠(ROUGE-1 和 ROUGE-2)来评估摘要信息是否完整，最长公共子序列(ROUGE-L)来评估摘要是否流畅。

4.1.5 参数优化

4.1.5.1 控制生成摘要长度归一的 alpha

由公式(3-21)可知，alpha 控制着生成摘要的长度，而 Google 在原论文[27]中提到 alpha 取值范围是[0, 1]，同时推荐范围是[0.6,0.7]。我们令 alpha = 0.4, 0.5, 0.6, 0.7，分别计算生成结果的 rouge 值有没有提升。

4.1.5.2 前馈神经网络的激活函数 ReLU

在经典的 Transformer 模型中，作者使用了 ReLU 函数作为前馈神经网络的激活函数。相比于最早期的 sigmoid 和 tanh 函数，ReLU 可以避免梯度消失的问题，使得大规模深度神经网络训练成为可能。但是 ReLU 也有自身的缺点：当出现极端数据时，权重和偏置的值可能会爆发式地增大，进而导致梯度爆炸。所以不得不采取一些干预方法来防止梯度爆炸现象发生（比如人工设置阈值）。

相比而言，GELU 是一个相对“新鲜”的激活函数。相当有名的预训练模型 bert 就用 GELU 作为激活函数。GELU 在激活中引入了随机正则的思想，是一种对神经元输入的概率描述，直观上更符合自然的认识，同时实验效果要比 Relu 与 ELU 都要好[33]。

我们令神经网络的激活函数分别为 ReLU 和 GELU，进行对比，看生成摘要的 rouge 值有没有提升。

4.2 结果与分析

4.2.1 实验结果

表 2 参数优化对于分层 Transformer 性能的影响

Beam Search 长度归一	ReLU			GELU		
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
α = 0.4	40.82	25.99	35.08	41.31	26.28	35.38
α = 0.5	40.82	26.09	35.09	41.32	26.36	35.38
α = 0.6	40.83	26.16	35.06	41.33	26.42	35.36
α = 0.7	40.83	25.97	35.05	41.29	26.38	35.33

表 2 总结了对于 Transformer 特定参数调节对模型带来的影响。首先是 beam search 中 alpha 值的调整，这关系到输出句子长度的设定。其次是前馈神经网络中激活函数选用 ReLU 和 GELU 的调整，相较而言，GELU 的曲线更加平滑，与 ReLU 相比，最大的优势是能避免梯度爆炸的问题。

从表中可以看出，当激活函数为 ReLU 的情况下，alpha 最好的取值是 0.6，大于 0.6 会造成 ROUGE-2，ROUGE-L 成绩的下降。

当激活函数为 GELU 的情况下，alpha 最好的取值是 0.6，大于 0.6 会造成 ROUGE-2，ROUGE-L 成绩的下降。

当 alpha 值不变的情况下，GELU 的训练效果比 ReLU 更好，ROUGE-1 平均提高 0.4875，ROUGE-2 平均提高 0.3075， ROUGE-L 平均提高 0.2925。

4.2.2 对比分析

我们将这个分层的 Transformer 模型与几个 baseline 方法进行比较：

Lead：是连接标题和排序段落的简单 baseline，并提取前 k 个 token;我们设 k 为基准目标的长度。

T-DMCA 是 Liu 等人(2018) [16]的最佳性能模型，也是具有内存压缩注意力机制的 Transformer decoder 的一种缩写;他们只用了 Transformer decoder 并利用卷积层压缩了自注

意力中的 key 和 value。该模型有 5 层，它的隐藏大小是 512，前馈隐藏大小是 2048。标题和排序段落被连接和截断为 3000 个符号。

我们的模型是分层 Transformer。模型体系结构是一个 7 层的网络(底部有 5 个 local Transformer 层，顶部有 2 个 global Transformer 层)。该模型将标题和$L^{'}=24$段作为输入，以生成目标摘要，这将导致每个实例大约有 1600 个输入 token。

表 3 与其他 baseline 方法的对比结果

方法	ROUGE-1	ROUGE-2	ROUGE-L
Lead	38.22	16.85	26.89
T-DMCA	40.77	25.60	34.90
本文使用的模型	41.33	26.42	35.36

表 3 总结了我们和其他模型的对比。表中包括了 Lead 和 T-DMCA 和我们所使用的模型产生的对比结果。可以看到，我们的模型生成的摘要得分优于 Lead 和 T-DMCA 模型。

4.3 本章小结

本章主要对本文多文本摘要模型部分参数优化进行实验设计和结果分析，并且与其他 baseline 方法进行对比。首先对于参数优化部分，本文分别对长度归一控制参数 α 和前馈神经网络 ReLU 进行优化，经过 WIKISUM 数据集的验证，有效提升了模型生成的摘要质量。与此同时与其他 baseline 方法进行对比，同样经过 WIKISUM 数据集的验证，摘要得分优于其他摘要模型。

5. 总结与展望

5.1 本文总结

本文针对多文本摘要中无法提取事件结构的问题，运用分段排序方法和分层 Transformer 方法，对多文档摘要问题进行研究。主要涉及方面的工作：基于长短时记忆网络的段落排序模型、分层 Transformer 的编码模型和摘要优化方法。经过参数优化实验和对比实验表明，本文提出的多文本摘要模型表现比其它摘要模型更好，对提升多文本摘要质量有一定现实意义。

5.2 系统局限以及未来展望

虽然本文引入的多文本摘要模型目前能取得不错的结果，但还有很多可以提升的方面：

首先，现有的评估标准并不能很好的评估摘要的质量。目前的摘要指标大多是基于召回率，而对摘要的关键信息保留程度以及语言流畅性不能很好的衡量。其次，大规模多文本摘要数据集的缺失仍然是不可忽视的障碍。虽然 WIKISUM 数据集可以作为多文本摘要的替代，但是人工标注的多文本摘要拥有更好的训练价值。最后，分层的 Transformer 虽然可以学习到段落内和段落间的信息，但是其巨大的开销和大量的参数使得模型训练成为困难，如何改进 Transformer 结构使其有更好的训练速度仍然是需要研究的课题。

6. 致谢

本科毕业设计历时 6 个月，收货良多。在这六个月中，我经历了很多之前没有经历过的挑战、挫折和磨难，也完成了前所未有的学习、积累和实践。首先要感谢我的导师——王学松老师。从课题的提出到收集资料到开始实验到实验大体完成，不论我遇到什么困难，不管遭遇什么挫折和挑战，王老师都在支持和鼓励我，为我指明方向。每次给老师汇报进度都让我受益匪浅，很多在我看来艰深复杂的话题在老师的料理下显得结构分明、条理清晰。同时，老师还在忙碌的工作之中抽出空闲时间帮我检查、修改我的开题报告，此期间老师带给我的分析问题的思路更是宝贵的精神财富，触类旁通，一通百通。更重要的是，王老师带着我一点点打开了 NLP 和深度学习的一扇大门，让我一窥里面的精彩和缤纷。师父领入门之后，我也将继续踏上在这个领域的修行之路。

还要感谢的是我的室友曾睿哲同学，在我还不熟悉代码的时候，面对大量的代码和结构常常感到非常棘手。他教会了我如何理解 torch 领域的常见思路，并且提供了很多学习和参考的资料，帮我快速上手相关代码。

感谢 Liu Yang 对我的指导和提点。他的工作对我的参考价值非常大，当我看到他的 github 仓库时，清晰的思路快速帮我理解我应该干什么。同时，当我遇到问题和麻烦发邮件向他请教时，他真的能及时回复并且耐心解答我的问题。

最后还要感谢我的父母，二十余载的养育之恩，学业上的无私支持，待我学成而归，定将报以汪洋。

7. 参考文献

[1] ERKAN G, RADEV D R. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization[J/OL]. Journal of Artificial Intelligence Research, 2004, 22: 457-479.

[2] CHRISTENSEN J, MAUSAM, SODERLAND S, etc. Towards Coherent Multi-Document Summarization[C/OL]//Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Atlanta, Georgia: Association for Computational Linguistics, 2013: 1163-1173[2022-04-03].

[3] YASUNAGA M, ZHANG R, MEELU K, etc. Graph-based Neural Multi-Document Summarization[J/OL].

[4] KIPF T N, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[J/OL].

[5] BARZILAY R, MCKEOWN K R. Sentence Fusion for Multidocument News Summarization[J/OL]. Computational Linguistics, 2005, 31(3): 297-328.

[6] FILIPPOVA K, STRUBE M. Sentence fusion via dependency graph compression[C/OL]//Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08. Honolulu, Hawaii: Association for Computational Linguistics, 2008: 177[2022-04-23].

[7] BING L, LI P, LIAO Y, etc. Abstractive Multi-Document Summarization via Phrase Selection and Merging[J/OL].

[8] SEE A, LIU P J, MANNING C D. Get To The Point: Summarization with Pointer-Generator Networks[J/OL].

[9] PAULUS R, XIONG C, SOCHER R. A Deep Reinforced Model for Abstractive Summarization[J/OL].

[10] GEHRMANN S, DENG Y, RUSH A M. Bottom-Up Abstractive Summarization[J/OL].

[11] CELIKYILMAZ A, BOSSELUT A, HE X, etc. Deep Communicating Agents for Abstractive Summarization[J/OL].

[12] ZHANG J, TAN J, WAN X. Adapting Neural Single-Document Summarization Model for Abstractive Multi-Document Summarization: A Pilot Study[C/OL]//Proceedings of the 11th International Conference on Natural Language Generation. Tilburg University, The Netherlands: Association for Computational Linguistics, 2018: 381-390[2022-04-23].

[13] LEBANOFF L, LIU F. Automatic Detection of Vague Words and Sentences in Privacy Policies[J/OL].

[14] MA S, DENG Z H, YANG Y. An Unsupervised Multi-Document Summarization Framework Based on Neural Document Model[C/OL]//Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee, 2016: 1514-1523[2022-04-23].

[15] CHU E, LIU P J. Unsupervised Neural Multi-Document Abstractive Summarization of Reviews[J/OL]. 2018[2022-04-23].

[16] LIU P J, SALEH M, POT E, etc. Generating Wikipedia by Summarizing Long Sequences[J/OL].

[17] GRUSKY M, NAAMAN M, ARTZI Y. Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies[C/OL]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018: 708-719[2022-04-23].

[18] HU B, CHEN Q, ZHU F. LCSTS: A Large Scale Chinese Short Text Summarization Dataset[J/OL].

[19] LIU Y, LAPATA M. Hierarchical Transformers for Multi-Document Summarization[J/OL].

[20] KUDO T, RICHARDSON J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing[J/OL].

[21] VASWANI A, SHAZEER N, PARMAR N, etc. Attention Is All You Need[J/OL].

[22] BA J L, KIROS J R, HINTON G E. Layer Normalization[J/OL].

[23] LIU Y, LAPATA M. Learning Structured Text Representations[J/OL]. Transactions of the Association for Computational Linguistics, 2018, 6: 63-75.

[24] KIM Y, DENTON C, HOANG L, etc. Structured Attention Networks[J/OL].

[25] Do latent tree learning models identify meaningful structure in sentences? | Transactions of the Association for Computational Linguistics | MIT Press[EB/OL]. [2022-04-23].

[26] NICULAE V, MARTINS A F T, CARDIE C. Towards Dynamic Computation Graphs via Sparse Latent Structure[J/OL].

[27] WU Y, SCHUSTER M, CHEN Z, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation[J/OL].

[28] BRIGHTMART. brightmart/nlp_chinese_corpus[M/OL]. 2022[2022-04-23].

[29] LIN C Y. ROUGE: A Package for Automatic Evaluation of Summaries[J]. 8.

[30] DUCHI J, HAZAN E, SINGER Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization[J]. 39.

[31] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the Inception Architecture for Computer Vision[C/OL]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, 2016: 2818-2826[2022-04-23].

[32] KINGMA D P, BA J. Adam: A Method for Stochastic Optimization[J/OL].

[33] DEVLIN J, CHANG M W, LEE K, etc. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J/OL].

[34] CLARKE J, LAPATA M. Discourse Constraints for Document Compression[J/OL]. Computational Linguistics, 2010, 36(3): 411-441.

[35] NARAYAN S, COHEN S B, LAPATA M. Ranking Sentences for Extractive Summarization with Reinforcement Learning[J/OL].

[36] Best-Worst Scaling: Theory, Methods and Applications - Jordan J. Louviere, Terry N. Flynn, A. A. J. Marley - Google 图书[EB/OL]. [2022-04-23].

[37] Hochreiter S , Schmidhuber J . Long Short-Term Memory[J]. Neural Computation, 1997, 9(8):1735-1780.