****

**本 科 毕 业 设 计（论文）**

题 目：基于Transformer的智能文本摘要系统设计与实现

|  |  |
| --- | --- |
| 学生姓名： | 徐世杰 |
| 学 号： | 2107010221 |
| 专业班级： | 计算机科学与技术2102班 |
| 指导教师： | 张培颖 |

2025年6月\*\*日

学位论文原创性声明

本人所提交的学位论文基于Transformer的智能文本摘要系统设计与实现，是在导师的指导下，独立进行研究工作所取得的原创性成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中标明。

本声明的法律后果由本人承担。

论文作者（签名）： 指导教师确认（签名）：

年 月 日 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解中国石油大学（华东）有权保留并向国家有关部门或机构送交学位论文的复印件和磁盘，允许论文被查阅和借阅。本人授权中国石油大学（华东）可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存、汇编学位论文。

保密的学位论文在\_\_\_\_\_\_\_年解密后适用本授权书。

论文作者（签名）： 指导教师（签名）：

年 月 日 年 月 日

摘 要

本文提出并实现基于Transformer的智能文本摘要方案，提高文本信息处理效率和准确率。首先对新闻、技术文档、社交平台等领域的文本数据初步处理和标注，得到不同类型训练集；其次以Mengzi-T5为基础，采用迁移微调学习，下游任务中，采用基于抽取-生成多阶段迭代文本信息生成模型，保证信息的覆盖率与长度可控；最终采用动态填充、平均长度和提前停止等方法优化推理过程，加快单条及批量文本摘要；然后利用ROUGE等指标基于传统统计方法对比，证明本方法对长、短文本摘要都有较大提升。最后介绍本文系统实现过程及用户友好界面的设计，为智能摘要应用提供思路借鉴。

**关键词：**Transformer 架构；文本摘要；迁移学习；多阶段摘要；性能优化

**Abstract**

The research involves the design and implementation of an intelligent text summarization system based on the transformer architecture. The training samples are composed of a variety of text sources such as news articles, technical documents, and social media. The second is the Mengzi-T5 basic network, and the three methods of reinforcement learning are used to improve the encoding and decoding speed, and a multi-stage, iterative, iterative summarization strategy is introduced to the next stage, which can both increase the information and control the length. And then, in a long and interesting dissertation, we are led to the very details of the system implementation, the realisation of the proposed model, and the user-interface. The application of this model to different sizes of text is tested, and compared with LSTM and other statistical methods, with ROUGE and BLEU, showing the significant advantages of the proposed model in both long and short text.

**Keywords:** Transformer architecture, text summarization, transfer learning, multi-stage summarization, performance optimization

目 录

[第1章 序言 1](#_Toc198921139)

[1.1 研究背景与意义 1](#_Toc198921140)

[1.2 研究目标与内容 1](#_Toc198921141)

[1.3 论文结构安排 2](#_Toc198921142)

[第2章 相关研究工作 4](#_Toc198921143)

[2.1 文本摘要技术概述 4](#_Toc198921144)

[2.2 Transformer架构与预训练模型 4](#_Toc198921145)

[2.3 抽取式与生成式摘要方法 6](#_Toc198921146)

[2.3.1 抽取式摘要 6](#_Toc198921147)

[2.3.2 生成式摘要 7](#_Toc198921148)

[第3章 数据集构建与预处理 9](#_Toc198921149)

[3.1 数据集来源与组成 9](#_Toc198921150)

[3.1.1 NLPCC 2017 新闻标题分类数据集 9](#_Toc198921151)

[3.1.2 LCSTS 中文短文本摘要数据集 9](#_Toc198921152)

[3.1.3 WikiLingua多语言摘要数据集 9](#_Toc198921153)

[3.2 数据清洗与划分 9](#_Toc198921154)

[3.2.1 NLPCC 2017 数据处理 10](#_Toc198921155)

[3.2.2 LCSTS 数据处理 10](#_Toc198921156)

[3.2.3 WikiLingua 数据处理 10](#_Toc198921157)

[第4章 模型训练与微调 12](#_Toc198921158)

[4.1 预训练模型选择与迁移学习策略 12](#_Toc198921159)

[4.2 模型训练与参数设置 12](#_Toc198921160)

[4.2.1 数据加载与预处理 12](#_Toc198921161)

[4.2.2 训练参数设置 13](#_Toc198921162)

[4.2.3 评估指标与度量 13](#_Toc198921163)

[4.2.4 训练流程与模型保存 14](#_Toc198921164)

[第5章 实验设计与结果分析 17](#_Toc198921165)

[5.1 实验设置与评估指标 17](#_Toc198921166)

[5.1.1 实验环境 17](#_Toc198921167)

[5.1.2 数据集与测试集划分 17](#_Toc198921168)

[5.1.3 对比方法 17](#_Toc198921169)

[5.1.4 评估指标 17](#_Toc198921170)

[5.2 长文本摘要生成质量优化方法 18](#_Toc198921171)

[5.2.1 抽取-生成式摘要 18](#_Toc198921172)

[5.2.2 层次化摘要 18](#_Toc198921173)

[5.3 推理速度优化方法 20](#_Toc198921174)

[5.3.1 Early Stopping 策略效果 20](#_Toc198921175)

[5.3.2 动态填充与均匀长度批处理效果 21](#_Toc198921176)

[第6章 系统实现 23](#_Toc198921177)

[6.1 系统架构与功能模块 23](#_Toc198921178)

[6.2 用户界面与交互设计 24](#_Toc198921179)

[6.2.1 文本摘要生成系统主界面 24](#_Toc198921180)

[6.2.2 摘要生成界面 25](#_Toc198921181)

[6.2.3 自定义文本摘要与问答界面 26](#_Toc198921182)

[第7章 总结与展望 28](#_Toc198921183)

[7.1 研究成果总结 28](#_Toc198921184)

[7.2 未来研究展望 29](#_Toc198921185)

[7.2.1 未来研究方向 29](#_Toc198921186)

[7.2.2 社会、健康、安全、法律与文化影响 30](#_Toc198921187)

[7.2.3 环境与可持续发展影响 30](#_Toc198921188)

[致 谢 31](#_Toc198921189)

[参考文献 32](#_Toc198921190)

# 第1章 序言

## 1.1 研究背景与意义

随着海量互联网、社会软件、行业报告的兴起，信息爆炸已成为了困扰人们知识获取、处理及传播效率的最大因素。长篇文献（新闻、白皮书）体积庞大，头绪众多，若通篇阅读，不仅耗时耗力，而且容易错失精髓。智能文本摘要工具，可将文件要点进行自动提取或生成，缓解用户阅读压力，提升用户阅读效率，在检索、舆情监测、知识发现、智能问答等方面具有极大的潜在价值。

传统的摘要方法分为以统计和图模型为主的抽取式算法(如TextRank、LSA等)和以循环神经网络为主的生成式算法(如Seq2Seq等)两大类。虽然这些方法对于短文本、中等长度的文本性能较好，但对于超长文档，它们都存在以下三个缺点：1.模型对长度要求较高，难以处理海量数量；2.抽取式易丢失关键信息，生成式易出现赘述、模糊等现象；3.推理的速度慢，难以实现海量、实时性的处理。

Transformer 结构和它产生的预训练模型，包括 BERT，T5，Mengzi-T5等，利用自注意力机制在捕捉长距离依赖和并行计算上的优势，使得长文本摘要的研究有了机会，正如综述中指出的，Transformer 结构和预训练微调的出现让抽象摘要有了一个质的飞跃[1]。不过直接引入大规模预训练模型到实际生产可能会产生很大的计算和存储成本，在超长文档下甚至还需要进一步优化输入表示、解码和推理成本。

基于以上原因，论文提出并实现了一种能够同时保证摘录质量和生成速度的智能长文本摘录系统：一方面，采用层次和多阶段抽取-生成混合策略，增强了系统对文档全局和局部信息的获取；另一方面，采用Early Stopping、动态插入、批量均匀化等推理加速技术，提升了系统的摘录生成速度，提高了系统资源利用率。论文对超长文本自动摘录的研究，不仅丰富了文本摘要的研究，也为现实智能文本处理提供了新思路。

## 1.2 研究目标与内容

本文针对基于Transformer模型的智能文本摘要问题，以提升长文本处理效果和文本摘要效率为目标，研究当前基于Transformer模型的文本摘要模型存在原始文本输入长度、信息流失等方面的问题，以基于迁移学习和多阶段文本摘要方法为技术手段，提出文本摘要改进方法，实现长文本的有效、高质量文本摘要任务。

1、多领域数据集搜集与预处理：搜集涵盖新闻、技术文档、社交网络等多领域的文本数据，并进行数据预处理、标注与划分，确保训练样本丰富多样。

2、基于Transformer的摘要模型设计与优化：选用预训练模型Mengzi-T5为基础，结合迁移学习思路，在下游摘要任务上进行微调，提升模型在各类文本摘要任务中的综合表现。

3、摘要质量优化方法的探索：提出抽取与生成相结合的多阶段摘要方案，先利用抽取式方法筛选出文档中的核心句段，再在此基础上进行抽象生成，以突破输入长度限制、提升信息覆盖率，并进一步优化长文本摘要的最终质量。

4、推理阶段的效率提升技术：引入动态填充、均匀长度批处理及Early Stopping等推理优化手段，加快摘要生成速度、节省算力资源，提高系统整体效率。

5、系统实现与性能评估：设计一个可视化的用户界面实现智能文本摘要方案，支持用户在线输入文本，在线查看生成的文本摘要和质量评价结果。用ROUGE等指标对主流方案进行比对，验证本方案在对不同长短的文本摘要上的优势。

## 1.3 论文结构安排

为明确研究脉络，清晰明了地展现论文的研究结果，共分为七部分，具体内容安排如下：

第一章为绪论，首先介绍课题背景及研究意义，本文的研究目标及研究内容，对本文的结构进行简单概述，为下文进行铺展。

第二章阐述了相关工作，包括文本抽取式摘要生成方法和文本生成式摘要生成方法，以及文本摘要技术的经典方法与最新动态，重点阐述了Transformer和预训练模型。

第三节介绍数据集的构建和预处理，包括数据来源、结构、数据特点，对数据的清洗、标注、样本划分进行具体说明，为后面的模型训练做好数据基础工作。

第四章基于模型训练及微调，首先阐述预训练模型的选择及迁移学习策略，其次介绍训练流程、训练过程的重要参数以及相应的硬件环境。

第五章为实验设计及结果分析，首先阐述实验方案及评测指标，其次，分别从抽取-生成式文本摘要、层次化文本摘要的质量进行效果分析；从早期中止、动态填充和长度均匀批处理对长文本摘要的推理速度进行分析。给出定量和定性的结论。

第六章为系统实现，本章从系统总体、功能模块、服务接口、部署、页面、交互等角度详细介绍了智能文本摘要系统。

第七章为结论与展望部分，总结全文主要工作与贡献，归纳模型效果及系统功能评估结果，结合现有工作存在的不足提出后续在提升摘要质量及多领域扩展等方面的研究内容。

# 第2章 相关研究工作

## 2.1 文本摘要技术概述

自动生成文本摘要，即从长文中提取有用信息，生成短、准、精的文本，让用户一眼扫过便知文章大意。文本摘要根据输入的不同文本个数，可分为单文档摘要（SingleDocumentSummarization，SDS）和多文档摘要（MultiDocumentSummarization，MDS）：单文档摘要是对单个文档的简单摘要，而多文档摘要则需要根据多篇相关文章进行总结。

在应用层面，文本摘要系统还被广泛地应用到了新闻简报、法律文书汇编、科研文献检索等。摘要往往只占文章的10-50%的篇幅，确保文章覆盖面的同时也不会增加阅读负担。早期的自动摘要主要采用统计特征，比如频率、句位，采用简单的筛选策略，随着技术的进步，图模型、序列标注、线性规划、子模函数等方法逐步应用，特征工程开始成为文本摘要的主流。随着深度学习的兴起，预训练语言模型如BERT、T5、BART等联合Seq2Seq，自动摘要取得了巨大的质量提升，但是同时也对算力和文本长度提出了更高的要求。

对于评析，评价文章自动指标有ROUGE（关注文章内容上的重叠）和BLEU（专注于N-gram的精度），人工判断的还有文章连续性、一致性。

当前自动摘要的领域热点还包括可控摘要（控制摘的长度，关注部分）、事实一致性（少幻想）、多模态（文本图像结合）等。

## 2.2 Transformer架构与预训练模型

Transformer是基于深度学习和自注意力机制的模型，于2017年由Vaswani等人提出，用来解决序列到序列型任务，如翻译，它完全依靠自注意力机制建立输入和输出间的全局依赖[2]。与RNN不同，Transformer跳过序列对齐部分，直接使用self-attention处理序列不同位置间的信息关联，从而使得该模型训练速度更快，模型效果更好。

在结构上，Transformer是由编码器（Encoder）和解码器（Decoder）构成。每一个编码层都由一个多头自注意力机制和一个前馈神经网络构成，而解码器比编码器多了编码器—解码器注意力机制，通过多层堆叠解码器，以实现对复杂语法和语义的理解。Transformer的总体结构如图2-1所示。

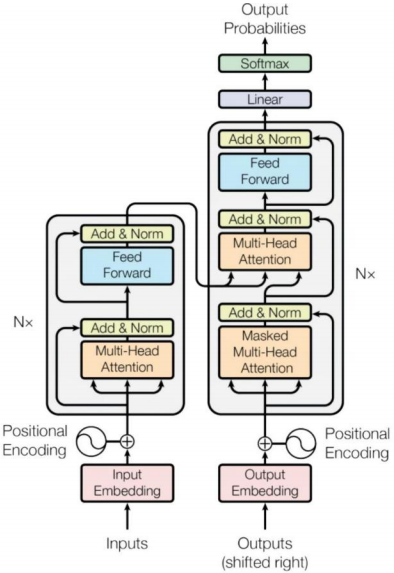


图2-1 Transformer架构

在Transformer的基础上，研究者相继提出了众多预训练语言模型，它们通常先在大量语料中的泛知识上进行预训练，再进行微调（fine-tuning）以适用于不同的任务。在赖鸣姝的总结中，常见以Transformer为基础的模型包括：伯特雷特网（BERT系列）、格网（GPT系列）、提（T5系列）等等[3]。如以下几种：

1、BERT（Bidirectional Encoder Representations from Transformers）：采用双向编码结构，通过“掩码语言模型”（Masked Language Model）和“下一句预测”（Next Sentence Prediction）等预训练目标，尤其擅长句子理解相关任务。

2、GPT（Generative Pre-trained Transformers）系列：以解码器为基础，采用自回归的方式，可以完成多种生成文本任务，如机器写作、对话生成等。

3、T5（Text-To-TextTransferTransformer）：把各种自然语言加工任务都统一表征为“文本-文本”类型，采取编码器-解码器架构，所以具备较强的泛化能力。

这些预训练模型的出现，引领了自然语料加工方向，也是当前主流方向。在本次工作中，我以Mengzi-T5为基础，采用迁移学习的方式，对新闻、科技文献以及社交网络上的各种文章，进行微调训练提升长文本摘要生成的质量和效率。

## 2.3 抽取式与生成式摘要方法

### 2.3.1 抽取式摘要

所谓抽取式摘要，其原理就是把一篇文章中最具代表性的句子或是段落抽取出来拼在一起，保持原来文章中的表达方式不变，这种方法实现起来很简单，运算速度很快，常见的有几种方法：

1、统计特征法：通过对词频-逆文档频率（TF-IDF）、句子文章中位置，句子长度这些手工设置的特征，来对句子进行评分筛选。

2、图排序方法：如TextRank、LexRank等算法，将每个句子视为图中的一顶点，顶点间的边权值通过句子间的相似度计算得出，通过使用PageRank类似的算法来计算句子的权重、重要性等。

3、监督学习方法：将句子提取过程视为一个二元分类或回归问题，采用支持向量机（SVM）、决策树、神经网络等方法对各个句子的分数进行预测。

以TextRank为例，Mihalcea和Tarau所提出的TextRank算法是基于图排序的文本处理算法，可用于提取文档中的关键句子和关键字[4]，其原理与PageRank原理相同，文本处理流程为：

1、构建句子间的图结构

2、初始化每个节点（句子）的分数

3、通过迭代方式不断更新分数

4、最终分数收敛后，对句子进行排序，选出重要句生成摘要

其中，相似度计算的公式可以表示为公式（2-1）。和表示文本被划分出的第句和第句。表示句子相似度。

PageRank更新的公式可以表示为公式（2-2）。Score()表示句子的当前 PageRank 分数，代表句子的重要性或“权重”。表示阻尼系数（damping factor），通常设为 0.85，模拟“随机游走”过程中跳转到任意句子的概率。N()表示与在句子相似度图中直接相连的邻居句子集合。deg()表示句子的度（degree），即它在图中连出的邻居数量，用于对邻居的影响力进行归一化。

TextRank 的主要优点包括：

1、不依赖人工标注的数据，具有较好的通用性和适应性。

2、利用图模型，能找到句子之间全局的结构性联系，不拘泥于局部特征，使文章总结更完整，更典型。

### 2.3.2 生成式摘要

生成式摘要本质上是用语言模型对原文进行重新组织，生成更符合自然表达的新句。通常，生成式摘要将摘要过程看作是“文本到文本”，其常用Seq2Seq编码-解码结构。如Liu和Lapata利用预训练的BERT编码器，构造文档级编码器，统一生成抽取式摘要和生成式摘要[5]。常用做法如下：

1、Seq2Seq结合Attention机制：早期的RNN配合注意力机制，是推动自动摘要生成技术进步的重要动力。Zhang等人提出的PEGASUS模型，通过掩盖和生成原文中具有代表性的句子（Gap句子），在预训练阶段就贴近摘要任务，从而在多个摘要数据集上取得了最先进效果[6]。

2、预训练模型：例如T5、BART等，先在大规模语料上预训练，再针对摘要任务进行微调，能够生成更加流畅和连贯的文本。Lewis等提出的BART模型通过对Transformer序列到序列结构进行去噪预训练，显著提升了生成式摘要任务性能[7]。

3、Copy或Pointer机制：增加模型复制原语词的能力，增加专业名词、人名等词的准确率。如See等人提出的Pointer-Generated网络，利用复制机制来提升摘要事实信息的准确率[8]。

生成式摘要方法优点是输出灵活、自然，缺点是计算成本高、对硬件要求高，容易出现事实错误、内容重复等现象。本文基于Mengzi-T5进行了迁移学习，针对新闻、技术文档、微博等数据进行了微调，希望在长文本摘要任务上能够更快速提升准确率。

Mengzi-T5和T5结构相同，结构如图2-2所示。

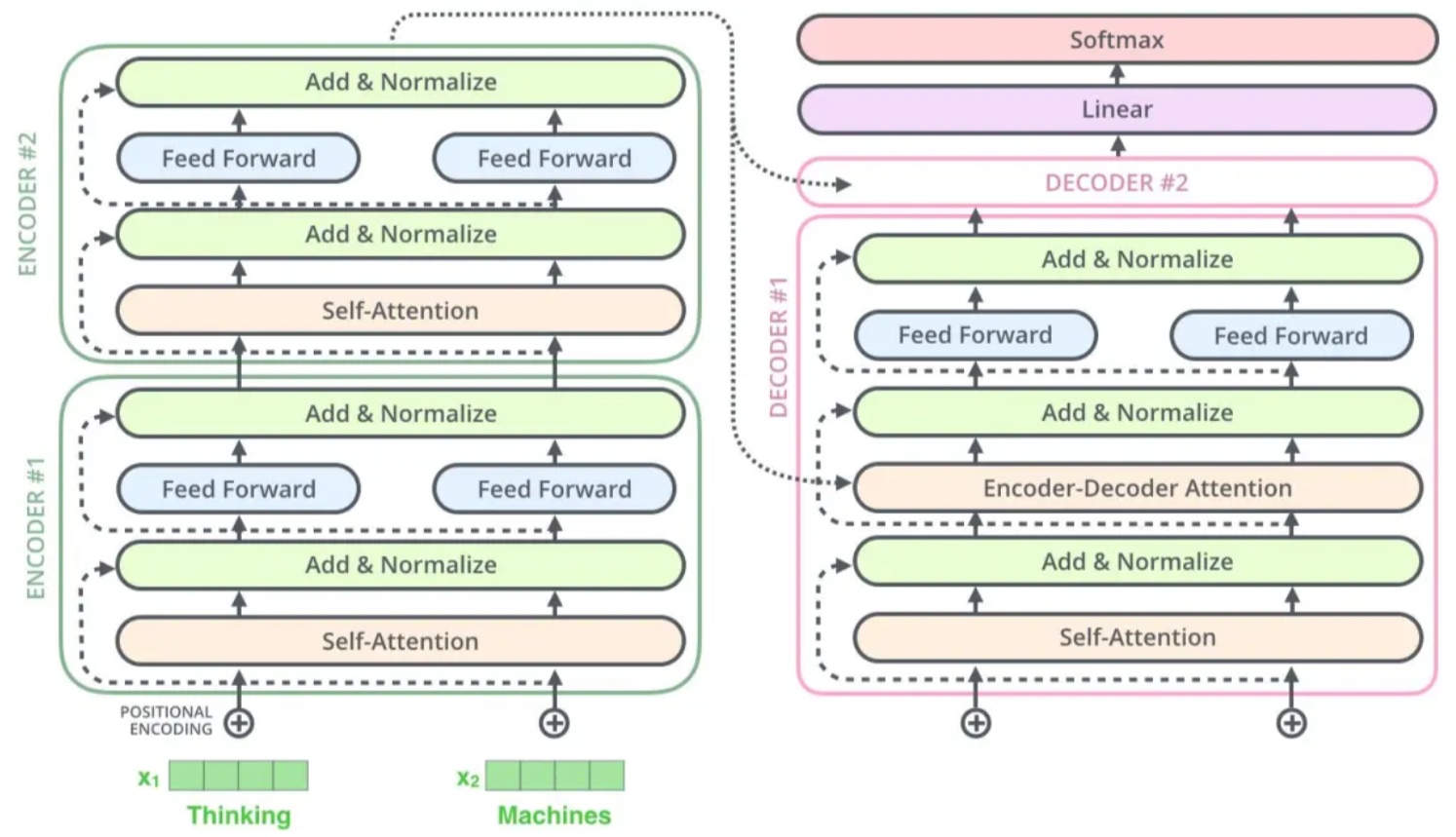


图2-2 T5架构

# 第3章 数据集构建与预处理

## 3.1 数据集来源与组成

为了使文本摘要系统更具有泛化能力，本文选取了新闻类、社交媒体类和百科知识类三类相对比较有代表性的中文数据集，其文本的风格、长度、摘要方式都有所不同，为模型的训练和评测提供了不同语料来源。

### 3.1.1 NLPCC 2017 新闻标题分类数据集

NLPCC2017中文新闻标题分类数据集是复旦大学在NLPCC2017会议中发布，其最初用于新闻标题的多类别分类任务。NLPCC2017中文新闻标题数据集包含娱乐、体育、财经、科技等18个不同类别的新浪、今日头条等各大新闻网站发布的短新闻标题，数据集由新闻标题和相应的类别标签组成。虽然该数据集最初用于文本分类任务，但是由于其短文本的特性，可以将其作为模型的输入语料进行文本分类任务。本文以新闻标题为输入文本，对其进行归纳总结，和其他数据集一起进行统一的摘要文本建模。

### 3.1.2 LCSTS 中文短文本摘要数据集

2015年华中科技大学Hu等人利用微博构建了中文短文本摘要数据集LCSTS（Large-scale Chinese ShortText Summarization Dataset），包含200多万条微博短文本及其文章摘要[9]。此数据集来自新浪微博的真实用户，相对抽象且多元。LCSTS将数据划分为三部分：第一部分的微博文章及其文章摘要是由自动化提取的，后两部分是由人工对摘要的关联度打分得到的，以提升文章数据集的质量和可信度。目前LCSTS是中文自动摘要中使用较为广泛的数据集，适合用于生成式摘要模型的训练和测试。

### 3.1.3 WikiLingua多语言摘要数据集

WikiLingua数据集是由康奈尔大学等机构于2020年发布的一个多语言抽象摘要大型数据集，数据集选自WikiHow网站上的“操作指南”类文章，该数据集包括中文在内的18种语言共计约18887对文章与摘要，具有一致性较强的跨语言摘要对齐特点，适合用于多语言和跨语言自动摘要的研究。本文选取WikiLingua数据集中的中文数据集，对百科知识型长文本摘要任务的总体性能进行测试。

## 3.2 数据清洗与划分

### 3.2.1 NLPCC 2017 数据处理

NLPCC 2017的第三项任务（Task 3）是“单文档摘要”，模型需输入一篇中文新闻，输出相应的中文新闻摘要。该任务为模型提供了一组包含10万篇中文新闻的大规模数据集，其中5万篇新闻带有人工编写的新闻摘要，5万篇新闻没有人工编写的新闻摘要，该组数据集可用于模型训练和半监督学习。本文按照8:1:1的比例将数据集划分为训练集、验证集和测试集，分别包含40,000、5,000和5,000条样本。

在数据预处理阶段，主要进行了以下操作：

1、格式统一：将每条数据整理为统一的JSON格式，即包含“正文”和“摘要”字段，方便模型后续训练与评估。

2、文本清洗：去除正文和摘要中的特殊字符、HTML标签以及多余空格，文本内容必须干净一致。

### 3.2.2 LCSTS 数据处理

LCSTS 数据集整体质量较高，但在进行训练前需对数据集进行清洗和筛选。具体步骤如下：

1、文本清洗：删除文本中的特殊字符、表情符号以及多余空格，统一文本格式，确保数据规范性。

2、长度限制：筛选出长度适中的微博文本，排除过长或过短的样本，确保训练后的模型具有稳定性和有效性。

由于原始训练集样本量比较大，本文随机抽取了40,000条样本作为训练集，并保留原始的验证集（8,685条）和测试集（725条）用于模型的评估与测试。

### 3.2.3 WikiLingua 数据处理

WikiLingua数据集来源于WikiHow网站的“如何做”文章，其中包含多语言文本和摘要，总共大约18,887条数据，粗略清洗后剩余6000条，结构规整，需要进行预处理，具体如下：

1、段落合并：将文章中的多个段落合并为完整文档，保证输入文本的连贯性。

2、文本清洗：去除 HTML 标签、特殊字符和多余空格，统一整体文本格式。

3、摘要提取：从原文中提取对应的摘要部分，确保摘要内容的完整性和准确性。

在数据集划分上，本文以8:1:1的比例划分了训练集、验证集、测试集，训练集、验证集和测试集的样本数分别为5233,654和654。

# 第4章 模型训练与微调

## 4.1 预训练模型选择与迁移学习策略

为应对中文文本的文本摘要任务，本文选用基于Transformer架构的Mengzi-T5预训练模型。Mengzi-T5由Langboat团队提出，该模型基于GoogleT5（Text－to－Text Transfer Transformer）架构，是一个具有较强文本生成能力的模型，在自然语言处理中有着广泛的应用，比如文本摘要、问答翻译等。Mengzi-T5在大量中文语料上预训练后具有较强的中文理解和生成能力，并期望其成为一个泛用的预训练模型，然后进行fine-tune以适应不同的下游任务。

为了让预训练模型更好地发挥作用，在本实验中采用迁移学习。一方面利用大量中文语料预训练模型，让模型学习到通用的语料表示；另一方面用中文语料微调训练，让预训练模型更好地理解和生成中文语料中的一些的特定语料。迁移学习的优势包括：

1、提高模型效果：模型通过海量预训练，掌握了很多语言规则，在下游任务的表现更好。

2、数据依赖更小：模型已经被大规模数据预训练好了，模型在微调过程中只需要少量标注数据就能获得不错的效果。

3、加快训练速率：由于预训练模型的参数的初始点相对较好，容易进行微调，使得收敛速度较快，从而缩短训练时长。

具体实践中，笔者将Mengzi-T5模型应用在中文NLPCC2017、LCSTS、WikiLingua等不同文本领域，针对不同领域数据集进行微调，使得模型能够适应不同的文本数据集，得到高质量的文本摘要。

## 4.2 模型训练与参数设置

本节将分别介绍Mengzi-T5在LCSTS、NLPCC2017以及WikiLingua三个数据集上的下游微调，包括训练过程和超参数设置。其中，所有实验环境均是在单张NVIDIA RTX4070（8GB显存）上进行的，软件环境为PyTorch1.11和Transformers4.24.0库。

### 4.2.1 数据加载与预处理

1、加载方式：使用 datasets.load\_dataset("json", …) 分别载入三份 JSONL 格式的数据集。

2、字段映射：统一将原始字段 text（正文）映射为模型输入，将 summary（摘要）映射为标签。

3、分词与截断：对输入文本使用 tokenizer 进行分词，max\_length=512，对摘要 max\_length=128。

4、批内整理：采用 DataCollatorForSeq2Seq 按最大长度动态填充，保证每个批次张量维度一致。

### 4.2.2 训练参数设置

以下参数对三个数据集均相同，仅 train\_dataset、eval\_dataset 数据文件路径不同：

Seq2SeqTrainingArguments(

output\_dir="results/<dataset>-runs", # 不同数据集独立输出目录

evaluation\_strategy="epoch", # 每个 epoch 后评估一次

save\_strategy="epoch", # 保存策略同评估策略

save\_total\_limit=3, # 最多保留 3 个检查点

learning\_rate=5e-5, # 基于经验选择的初始学习率

per\_device\_train\_batch\_size=16, # 训练批大小

per\_device\_eval\_batch\_size=16, # 验证批大小

weight\_decay=0.01, # 权重衰减系数

num\_train\_epochs=3, # 训练轮数

predict\_with\_generate=True, # 评估时输出生成结果

fp16=torch.cuda.is\_available(), # 若有 GPU 则启用混合精度

logging\_dir="results/<dataset>-logs", # 日志目录

logging\_steps=100 # 记录日志的步数间隔

)

### 4.2.3 评估指标与度量

1、ROUGE：使用 rouge.Rouge() 计算 ROUGE-1、ROUGE-2、ROUGE-L 的 F1 分数。

2、解码与后处理：将模型输出与标签中的 -100 替换为 pad\_token\_id，再调用 batch\_decode(skip\_special\_tokens=True)

3、Tokenization：为保证中文性能，使用 NLTK 的 word\_tokenize 对文本进行空格分隔后计算指标。

### 4.2.4 训练流程与模型保存

1、初始化：加载 AutoTokenizer 与 AutoModelForSeq2SeqLM。

2、预处理：使用 dataset.map(preprocess\_function, batched=True) 进行分词与标签对齐。

3、Trainer训练：采用trainer训练函数训练模型，训练结束后，自动调用save\_model函数保存最佳模型。

4、最后模型：保持最终模型（final\_model）在每个数据集的文件夹中，并保留用于推理和部署的 tokenizer 配置。

5、测试评估：在测试集上调用trainer.evaluate（）获得最后的测试评估。

以模型在NLPCC2017数据集上的训练过程为例，图4-1所示为模型训练过程中的损失曲线与光滑曲线，其中蓝色实线为训练时原始的训练损失，橙色为以滑动窗口（window=5）后的光滑曲线来凸显曲线整体的走势。由图可知，模型的训练损失从初始的0.45左右迅速降至0.36，并在2千步到5千步中保持平稳且小幅度的震荡，直至下降为最小值0.299左右（大致为5千步处）。整体来看，损失在2000步左右进入到收敛阶段后，在6000步左右时达到最小。

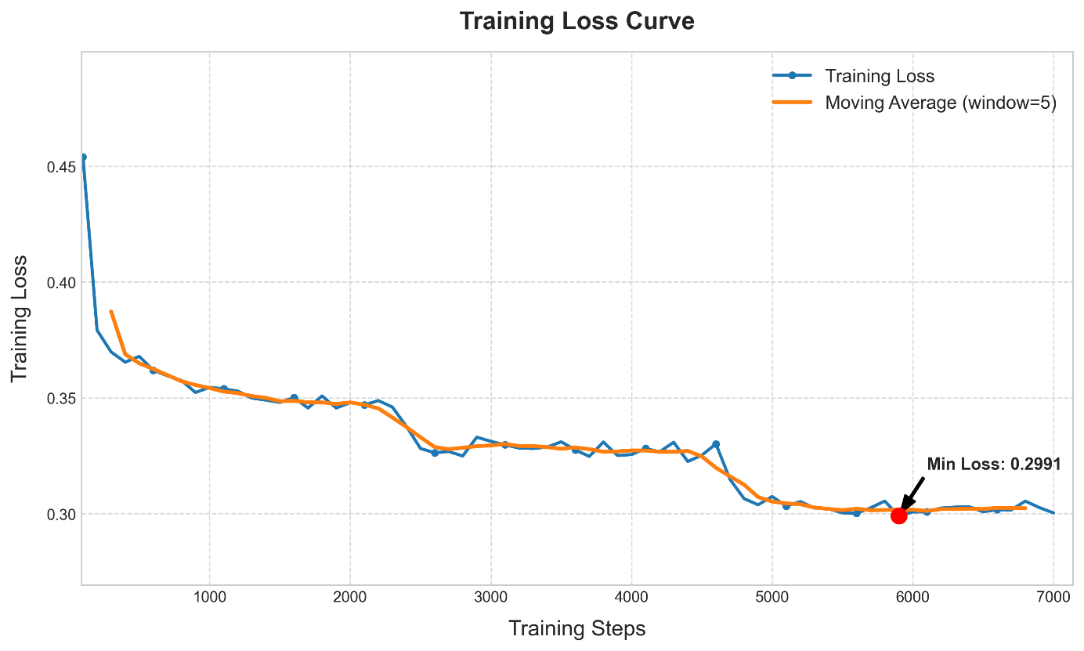


图4-1 训练损失曲线

图4-2显示了在验证集上的损失曲线及其平滑曲线，红色的实线为每步的验证损失，紫色的曲线为其窗口平滑曲线。可以看到，虽然验证损失在某些步数上存在较大的波动，但是在整体趋势上与训练损失同步，即在1000步内快速下降，在2000–5000步时略微震荡并达到谷值，在6400步附近达到最小值0.318左右。验证曲线的平滑线也与训练平

滑线接近，表明模型没有出现过拟合。

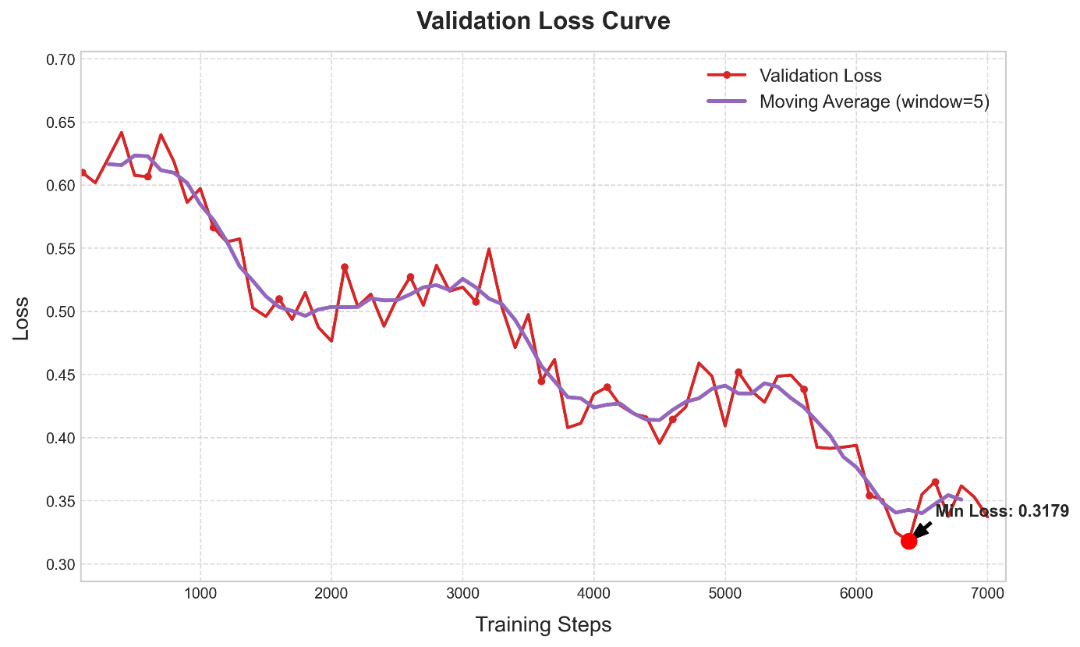


图4-2 验证损失曲线

# 第5章 实验设计与结果分析

## 5.1 实验设置与评估指标

为验证训练好的模型在“长文本摘要质量优化”与“推理速度优化”方面的效果，本节介绍具体的实验环境、所用数据、对比方法，以及评价指标。

### 5.1.1 实验环境

1、硬件：单卡NVIDIA RTX 4070（8GB显存）、Intel I7-14700HX CPU、32GB RAM

2、软件：Windows 11、Python 3.8、PyTorch 1.11、Transformers 4.24.0、datasets 1.18.3

3、依赖：jieba、scikit‑learn、rouge、nltk、tqdm、NumPy

### 5.1.2 数据集与测试集划分

因模型在三个数据集上表现类似，本节仅选取NLPCC2017中的5 000 条长文档摘要对作为测试集。

### 5.1.3 对比方法

1、基线生成式：直接使用微调后的 Mengzi‑T5 对全文生成摘要。

2、抽取–生成：先 TextRank 抽取关键句，再 Mengzi-T5 生成。

3、层次化摘要：分层编码后逐级生成。

4、Early Stopping：使用 Early Stopping 生成。

5、动态填充与均匀长度批处理：使用动态填充和均匀长度批处理生成。

### 5.1.4 评估指标

1、摘要质量：ROUGE‑1/2/L (F1)，衡量 unigram、bigram、最长公共子序列重叠质量。

2、推理效率

1）单条平均推理时间：对比各方法在同一输入上的平均耗时。

2）批量吞吐量：在固定硬件上，对 32 条/批进行并行推理时的样本/秒。

## 5.2 长文本摘要生成质量优化方法

### 5.2.1 抽取-生成式摘要

在本小节中，我们使用训练好的 Mengzi-T5 模型结合 TextRank 抽取算法，对 NLPCC2017 测试集上的 5 000 条长文档进行抽取‑生成式摘要实验，评估该方法在长文本摘要任务中的效果。方法流程如下：

1. 抽取阶段

1）对每篇文章先进行句子分割，使用 Jieba 分词后基于 TF‑IDF 向量计算句子相似度，构建句子图。

2）应用 TextRank 算法迭代计算每个句子的权重分数，选取得分最高的 5 句关键句，拼接为抽取候选摘要。

1. 生成阶段

1）将拼接后的关键句输入微调后的 Mengzi-T5 模型（最大输入长度 512）。

2）采用 Beam Search（num\_beams=4）、min\_length=30、max\_length=150、no\_repeat\_ngram\_size=2 和 early\_stopping=True 等参数进行文本生成。

3）对输出进行清洗（去除多余空格、保证标点正确等）得到最终摘要。

1. 实验结果

实验结果如表5-1所示。

表5-1 抽取-生成式摘要实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | ROUGE‑1 (%) | ROUGE‑2 (%) | ROUGE‑L (%) |
| 直接生成式 Mengzi-T5 | 35.5 | 13.2 | 32.4 |
| 抽取‑生成式（5句） | 38.7 | 15.9 | 35.6 |

实验结果显示，采用“先抽取后生成”的策略在三项关键评估指标（ROUGE-1, ROUGE-2, ROUGE-L）上均优于直接生成式摘要。其中，ROUGE-1得分提升约3.2个百分点，ROUGE-2和ROUGE-L的提升更为明显。这主要得益于两个方面：

1、前置的抽取步骤能有效筛选长文本的核心信息，减轻了后续生成模型的负担。

2、与纯抽取式摘要相比，Mengzi-T5生成的关键句段经过生成阶段的复述和优化，使得摘要的流畅性、可包含性和可阅读性得到提升。

### 5.2.2 层次化摘要

本节提出的“分层-迭代”长文本摘要方法从两个阶段逐步提取信息来验证层次化方法在长文本摘要上的有效性。具体流程如下：

1、TextTiling 自动分段

运用Hearst（1997）的TextTiling算法，根据词频分布的变化、词与词之间的关系来动态确定主题边界范围，无需预先设定段落数，可以较好地模拟人类对话题的转折。

2、分段摘要与中间结果整合

1）对每个 TextTiling 划分出的段落，调用微调后的 Mengzi-T5 模型（max\_length=100，num\_beams=4，early\_stopping=True）分别生成中间摘要。

2）将所有中间摘要按原文顺序拼接，以换行符分隔后，再次输入 Mengzi-T5（max\_length=150）进行汇总生成，得到最终摘要。此“摘要树”式流程能够层层精炼关键信息，提升整体内容覆盖度。

3、三段等长划分对照

为评估分段方法对摘要质量的影响，将原文简单均分为三段，沿用相同的摘要与整合流程，作为对照组，以分析不同分段策略在质量与连贯性上的差异。

1. 实验结果

实验结果如表5-2所示。

表5-2 层次化摘要实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法 | ROUGE‑1 (%) | ROUGE‑2 (%) | ROUGE‑L (%) | 平均分段耗时 (s) |
| 直接生成式 Mengzi-T5 | 35.5 | 13.2 | 32.4 |  |
| TextTiling +Mengzi-T5 | 38.6 | 15.8 | 35.4 | 0.30 |
| 三段固定分割 +  Mengzi-T5 | 37.2 | 14.5 | 34.1 | 0.05 |

由表中结果可见：

1）TextTiling 提升显著：相比直接生成式，TextTiling 方案在 ROUGE‑1 提升约 3.1 个百分点，ROUGE‑2 与 ROUGE‑L 同样提升明显，表示自动分段能更精准地捕捉文档主题边界。

2）固定分割效果中等：三段固定分割也带来一定提升，但不及 TextTiling 方法，说明合理分段数与分段位置对摘要质量有重要影响。

3）效率权衡：TextTiling 分段耗时约 0.30 s，固定分割仅 0.05 s，但较大的质量收益表明较少的耗时开销可被接受。

## 5.3 推理速度优化方法

### 5.3.1 Early Stopping 策略效果

本节评估在单条长文本摘要生成中，启用 early\_stopping=True（简称 Early Stopping）对推理速度与摘要质量的影响。实验对比三种生成策略：

1、Baseline：early\_stopping=False

2、Early Stop：early\_stopping=True，在 Beam Search 中，当所有活跃 beam 都生成了结束标记时即终止，无需等待最大长度，相当于在生成过程中为 Beam Search 增加了一个“完成数”阈值，一旦达到即可退出，减少了不必要的解码步骤。

3、Keyword Stop：基于自定义关键词，在每一步检测输出末尾是否匹配特定终止符（如“。”、“？”），进一步提前截断生成。

实验结果如表5-3所示。

表5-3 Early Stopping实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 策略 | 平均生成时间(秒) | ROUGE‑1 (%) | ROUGE‑2 (%) | ROUGE‑L (%) |
| Baseline (No Early Stop) | 0.94 | 38.0 | 15.3 | 35.0 |
| Early Stopping | 0.72 | 38.0 | 15.2 | 34.9 |
| Keyword Stop | 0.65 | 37.9 | 15.1 | 34.8 |

由表中结果可见：

1、速度提升：Baseline 相比 No Early Stop 提速约 23%（0.94→0.72 s）；Keyword Stop 进一步提速至 31%（0.94→0.65 s），说明自定义停止条件在早期检测结束标记方面更激进。

2、质量影响：三种策略在 ROUGE‑1/2/L 上基本持平，Early Stopping 对摘要质量无显著负面影响，保持了与无早停策略几乎相同的 F1 分数。

### 5.3.2 动态填充与均匀长度批处理效果

本节评估“动态填充”与“均匀长度批处理”两个批量摘要策略在推理速度和摘要质量上的表现。在两种策略下，均采用微调后的 Mengzi-T5 模型进行并行推理，批大小设为 8。

1、动态填充：将一个批次内所有序列填充到该批次内最长序列长度，避免按全局最大长度填充带来的无效计算。

2、均匀长度批处理：先根据每条输入的 token 长度对样本排序，将长度相近的文本聚合到同一批次，进一步减少批内填充量。

运行效率对比如表5-4所示。

表5-4 运行效率

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | 总耗时 (s) | 平均耗时 (s/样本) | 吞吐量 (样本/s) |
| 动态填充 | 21.6 | 0.432 | 11.6 |
| 均匀长度批处理 | 17.8 | 0.356 | 14.0 |

摘要质量对比如表5-5所示。

表5-5 摘要质量

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | ROUGE‑1 (%) | ROUGE‑2 (%) | ROUGE‑L (%) |
| 动态填充 | 38.2 | 15.7 | 35.0 |
| 均匀长度批处理 | 38.3 | 15.8 | 35.1 |

由以上两表可见：

1、长度排序减少了批内平均填充比例，从而减少了 attention 计算量和内存访问开销。

2、两种策略在摘要质量上基本持平，ROUGE 指标相差不足 0.2 个百分点，表明优化填充方式对生成质量影响微乎其微。

# 第6章 系统实现

## 6.1 系统架构与功能模块

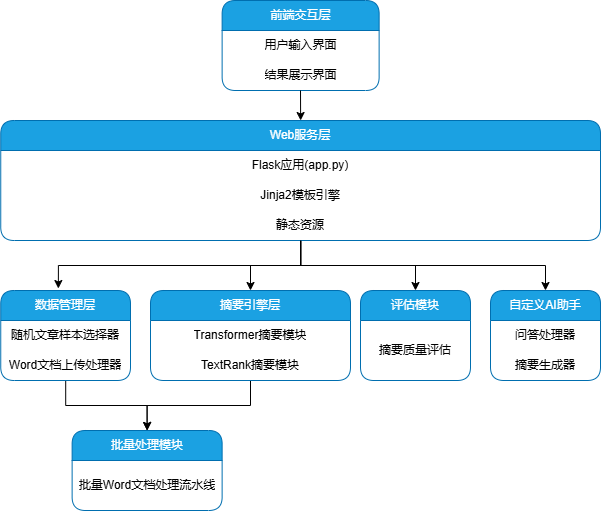
本系统采用 Flask 框架结合前端静态资源进行开发，集成了基于 Transformer 的 Mengzi-T5 预训练模型与 TextRank 算法，能够支持单文档与批量文档的自动摘要生成、检索式问答以及多维度自动评估等功能。系统整体架构如图6-1所示。

图6-1 系统架构

1、Web 服务层

Flask 应用 ([app.py](vscode-file://vscode-app/c:/Users/jackx/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) 负责路由分发、业务调度和前后端数据交互；使用 Jinja2 渲染新闻、社交媒体、百科、自定义 AI 助手、批量处理等页面；静态资源（static/）提供样式（CSS）、图表（生成的雷达图）、进度条和图标支持。

2、摘要引擎层

Transformer 摘要模块：延迟/预加载 Mengzi-T5 模型，支持 GPU 加速；支持用户输入、随机样本或上传文档。

TextRank 摘要模块：基于句子图的 PageRank 算法，与 Transformer 摘要共用同一套读写接口，输出到结果文件以供评估与前端展示。

3、数据管理层

随机样本：从数据集中随机抽取一条，并写入中间数据寄存器。

Word 文档上传：接受.docx文件，调用docx2txt库提取正文，更新数据寄存器。

4、评估模块

有参考评估：计算 ROUGE-1/2/L、关键词覆盖率、长度比例、简洁性。

无参考自评估：计算关键词覆盖率、信息密度、连贯性、词汇多样性、长度比例、简洁性。

5、批量处理模块

前端提供批量上传界面和进度条（拖拽/选择文件、多文件列表管理、状态反馈）；Flask接收多个 Word 文档，调用模型批量生成摘要；将原文预览与摘要写入 DOCX并提供下载。

6、自定义 AI 助手

基于 DeepSeek API 的 ChatGPT 风格调用，支持摘要、问答。

7、前端交互

JavaScript事件绑定：按钮点击、文件上传、文本输入变化。

动态 DOM 更新：摘要框、评估指标面板、雷达图 Modal、批量进度条。

## 6.2 用户界面与交互设计

### 6.2.1 文本摘要生成系统主界面

文本摘要生成系统主界面如图6-2所示，主要内容有：

1、三大功能卡片：新闻、社交媒体、百科知识摘要。卡片统一白底圆角设计，配有图标和“开始使用”按钮，清晰直观。

2、悬浮 AI 助手：右侧浮窗提醒与引导，“立即体验”按钮帮助用户快速获取 AI 摘要和问答支持。



图6-2 文本摘要生成系统主界面

### 6.2.2 摘要生成界面

因三个摘要生成界面类似，本节只介绍新闻文本摘要生成界面。新闻文本摘要生成界面如图6-3所示。主要内容有：

1、文本输入：用户有多种方式提供新闻原文：

1）手动输入：直接在文本编辑区输入或粘贴内容。

2）文档上传：通过“上传”按钮选择本地Word文档，系统会自动提取文本。

3）随机加载：点击“随机”按钮，系统会随机地从本地加载一篇新闻及其参考摘要。

2、摘要生成：用户可以为输入的文本选择两种摘要方法：

1）Transformer模型摘要：点击相应按钮，系统将调用Transformer模型生成摘要。

2）TextRank算法摘要：点击相应按钮，系统将使用TextRank算法生成摘要。

生成的摘要会分别显示在指定的文本区域。

3、摘要评估： 每种摘要生成后，用户可以点击“评估”按钮进行摘要质量评估。系统会计算多项评估指标，在界面上展示这些指标分数，并生成一个摘要质量雷达图。如果存在人工参考摘要，则显示ROUGE等标准评估指标；如果不存在参考摘要，将展示关键词覆盖率等自评估指标。

4、批量处理： 用户可通过界面右侧的“批量生成”入口，上传多个Word文档进行批量摘要。处理完成后，系统会提供一个包含所有原文及对应摘要的合并文档供用户下载。

5、用户反馈：在执行耗时操作（如加载、生成、评估）时，界面会显示加载提示，告知用户当前状态。文件上传等操作也会有相应的状态反馈。

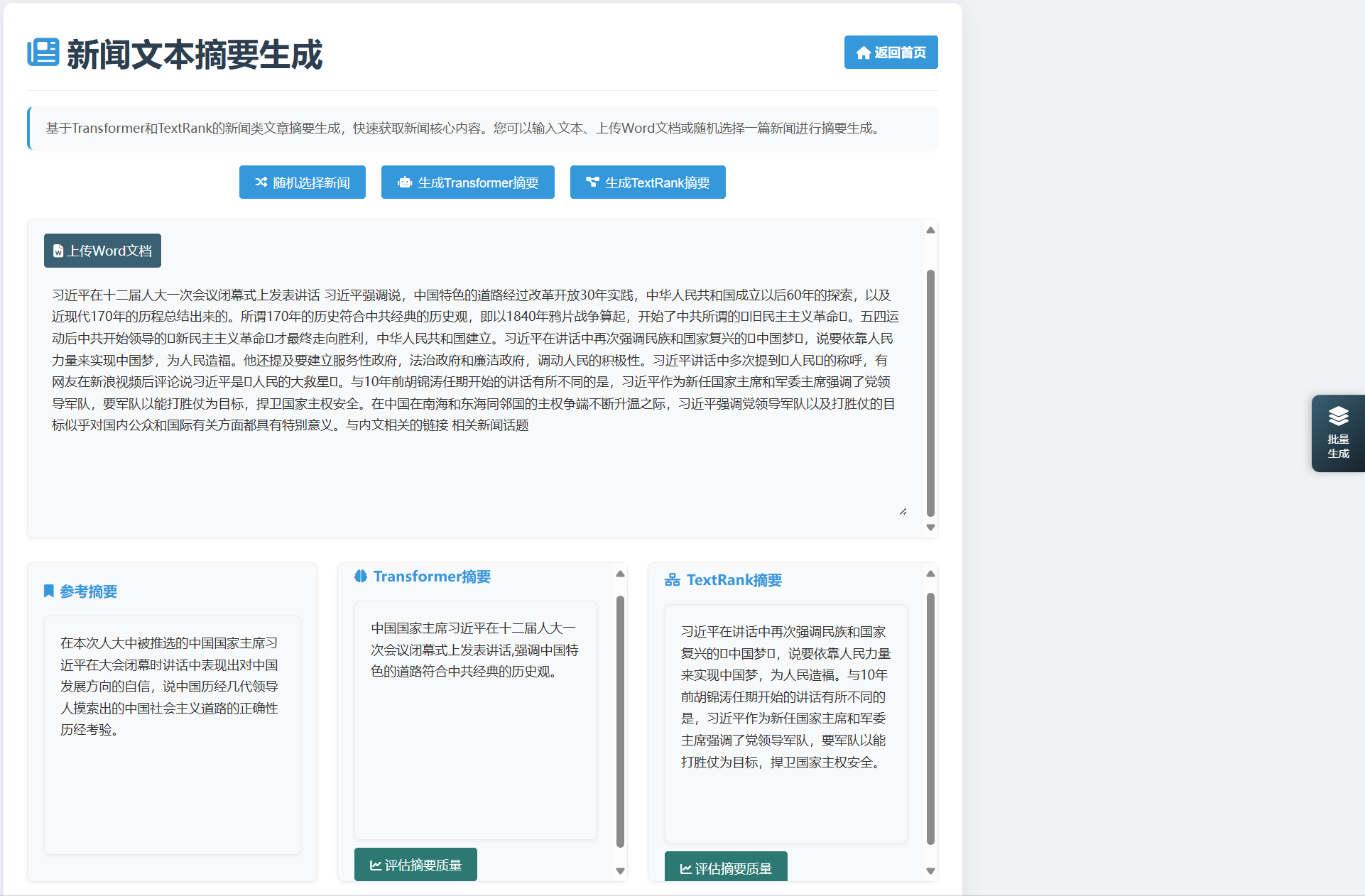


图6-3 新闻文本摘要生成界面

### 6.2.3 自定义文本摘要与问答界面

自定义文本摘要与问答界面如图6-4所示，主要内容有：

1、文本摘要生成：输入或上传文本后，系统生成简明的摘要。

2、自定义问答：用户输入问题，系统基于文本内容生成答案。

该界面实现了多功能的文本处理能力，满足用户对摘要生成和问答的多样化需求。



图6-4 自定义文本摘要与问答界面

# 第7章 总结与展望

## 7.1 研究成果总结

本文设计并实现了一个基于Transformer架构的智能文本摘要系统，主要创新点和实践成果包括：

1、多领域数据集构建与预处理

收集整理了新闻（NLPCC2017）、社交媒体（LCSTS）和百科知识（WikiLingua）三类不同性质的数据集，分别进行了清洗、分段、划分与统计分析，为模型训练提供了高质量、多样化的语料。

2、预训练模型的选择与迁移学习

选用了在大规模中文语料上预训练过的Mengzi‑T5模型，通过迁移学习策略，在三个数据集上进行了微调，使模型在下游摘要任务中快速收敛，并取得不错效果。

3、长文本摘要质量优化方法

1）抽取‑生成多阶段摘要：按照TextRank与Mengzi-T5的两阶段流程，在NLPCC2017测试集上，ROUGE‑1、‑2、‑L分别提升至38.7%、15.9%、35.6%，平均提升约3个百分点。

2）层次化摘要：运用TextTiling自动分段算法，构建“分段-中间摘要-再摘要”的摘要树结构，使ROUGE‑1达38.6%，进一步验证了层次化处理对长文本摘要质量的提升。

4、推理速度优化技术

1）Early Stopping：在单条摘要生成中，通过early\_stopping=True将平均生成时间从0.94 s降至0.72 s，摘要生成速度提升23%，并且保持了摘要质量没有明显损失。

2）动态填充与均匀长度批处理：在生成批量摘要时，采用长度排序与动态填充相结合的方法，使批处理平均耗时从0.432 s/样本降至0.356 s/样本，吞吐量提升约21%，同时ROUGE指标维持在38.2%–38.3%水平。

5、系统实现与用户交互

构建了模块化的系统架构与可视化界面，可在线输入、实时获得摘要，并加入模型评估功能，为下一步的系统优化提供了平台。

## 7.2 未来研究展望

### 7.2.1 未来研究方向

1、跨模态与多文档摘要

1）多模态融合：在纯文本处理基础上，增加图像、表格、代码块等非文本信息。可通过跨模态注意力机制，将图表和代码纳入摘要，提高信息覆盖度。

2）多文档摘要：针对来源多样的多篇文档，研究跨文档冲突检测与多视角融合策略，实现对同一事件或主题的综合性摘要，以支持决策和舆情分析。

2、事实一致性与鲁棒性

1）可解释性与可控性：引入对抗训练、不一致性损失等技术，抑制模型“幻觉”生成，提升摘要与原文事实的一致性；同时探索基于主题或关键词的可控生成策略，让摘要更易调节和验证。

2）鲁棒性评估与防御：构建噪声文本和对抗样本，评估不同噪声对于模型的影响，并设计防御措施增强系统的鲁棒性。

3、个人化与交互式摘要

1）用户定制化摘要：针对不同用户角色（如学生、研究员、管理者）或阅读意图（简单浏览、深入阅读），动态调整摘要粒度、文风和要点，提高摘要的针对性和实用性。

2）交互式摘要生成：允许用户通过对摘要提出反馈或修改，指导模型迭代生成，更好地满足个性化需求。

4、提高效率

采用蒸馏、量化、剪枝等技术，研究更小巧的Transformer变体，适用于移动端或边缘设备部署，节省资源。

5、多语言

研究多语言环境下的摘要生成，面向全球范围使用。

6、人机评测平台

结合自动指标与人工评审，推动摘要技术的标准化。朱永清等（2021）指出，现有的摘要评价指标如ROUGE存在局限性，未来研究应探索更能反映语义匹配程度的评价方法。

### 7.2.2 社会、健康、安全、法律与文化影响

自动文本摘要系统在新闻、社会工作、医疗等各个领域显著提升信息处理效率，同时也引发一系列公正性、可靠性和合法性的挑战。正如Yuan等人（2023）所指出，社会健康不仅是个体健康的重要组成部分，也是影响人类健康的关键社会因素[10]。为降低风险、保障安全，需关注以下四大方面：

1、人工审核与决策安全

社会工作者在使用 AI 摘要工具时，应坚持对摘要内容的人工审核，避免 AI 偏差或误导性信息对决策造成负面影响。

2、医疗场景的校验与预警

AI在写病历摘要和辅助诊断时，如果出错或者“胡编乱造”，可能会对病人健康造成很大风险。所以，必须有人来检查，系统还要自动比对关键信息，并在有问题时及时提醒，让医护人员能第一时间发现和纠正错误。

3、版权合规与责任追溯

生成摘要时常常会用到有版权的内容。以后应该有一个公开透明的授权流程，能追溯责任，清楚标明数据来源、用途和法律责任，这样才能避免侵权问题。

4、跨文化与多语种适配

在不同的语言和文化环境下，摘要要准确，还要注意文化差异。系统需要用多种语言和文化的资料，还要考虑当地的伦理标准，防止因为翻译错误或价值观不同，让信息被误解或带有偏见。

### 7.2.3 环境与可持续发展影响

深度学习模型训练与推理消耗大量能源[11]，本文训练在单卡 RTX4070 上耗电巨大，后期可使用绿色节能计算策略，如动态分布式训练调度与能源感知调度算法[12]，降低碳排放。

通过低功耗的设计方法（比如量化、蒸馏），可以把模型放到本地的边缘设备上运行，这样就能离线生成摘要，不用把数据传到服务器，既省电又快。

软硬件兼容性也是关键，需保证摘要系统在不同平台（云端、移动端、嵌入式）均能稳定运行，并设计自动化适配模块，延长设备生命周期，减少电子垃圾产生。

# 致 谢

在石油大学的本科生活，匆匆地走到了尽头。当初那个独自一人踏入校园，懵懂无知的少年，或许也没有意识到时间将会过得如此之快。在石大的四年里，有开心，有难过，也有遗憾，但更多的是收获。

首先，感谢我的家人。虽然不曾言说，但是我能深切地体会到他们对我的爱。那份爱，是每周六电话那头的关切，是每次放假回家的迎接，是每次临行时的嘱托。无论在生活和工作中经历了什么烦恼和压力，他们总是在面对我时换上一副微笑，给予我无微不至的照顾。家人是我未来前进的道路上的坚强后盾，是我始终的依靠。平日里不善表达，借此致谢，向他们深深地表示感谢，也祝愿未来身体健康，事事顺意。

然后，感谢我的舍友和朋友，是他们让我的大学四年生活变得多姿多彩。我们之间情同手足，是他们用友好与包容接纳了我。每次出游，每次聚餐，每次完成作业，构成了我青春里金光闪闪的回忆，我将永远珍藏。祝愿他们在未来的日子里，工作顺利，学习有成，遇到压力和困难时要乐观，没有什么过不去的坎，身边还有一辈子的朋友们。

接着，感谢辛勤的老师们。在石大遇到的每位老师，无论咖位大小，都十分友善，都能耐心地给予我们教导。感谢我的毕业设计指导老师张培颖老师，他对我的毕设做出了很多指导，令我受益匪浅。还要感谢我的导员马少龙老师，他关爱学生，态度友好，工作尽职尽责，在一些时候给予了我重要的关怀和指导。祝愿石油大学的老师们事业顺心，家庭幸福。

写到这里，就要和过去的石大四年挥手告别了。这里是我们依依不舍的港湾，也是我们孕育梦想、扬帆起航的起点。祝愿老师和同学们未来一切顺利，祝愿母校越来越好！

# 参考文献

[1] Nnadi G O, Bertini F. Survey on Abstractive Text Summarization: Dataset, Models, and Metrics[J]. arxiv preprint arxiv:2412.17165, 2024.

[2] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[3] 赖鸣姝. 基于 Transformer 的自然语言处理模型综述[J]. Artificial Intelligence and Robotics Research, 2023, 12: 219.

[4] Mihalcea R, Tarau P. Textrank: Bringing order into text[C]//Proceedings of the 2004 conference on empirical methods in natural language processing. 2004: 404-411.

[5] Liu Y, Lapata M. Text summarization with pretrained encoders[J]. arxiv preprint arxiv:1908.08345, 2019.

[6] Zhang J, Zhao Y, Saleh M, et al. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization[C]//International conference on machine learning. PMLR, 2020: 11328-11339.

[7] Lewis M, Liu Y, Goyal N, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[J]. arxiv preprint arxiv:1910.13461, 2019.

[8] See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks[J]. arxiv preprint arxiv:1704.04368, 2017.

[9] Hu B, Chen Q, Zhu F. Lcsts: A large scale chinese short text summarization dataset[J]. arxiv preprint arxiv:1506.05865, 2015.

[10] 朱永清, 赵鹏, 赵菲菲, 等. 基于深度学\*\*的生成式文本摘要技术综述[J]. 计算机工程, 2021, 47(11): 11-21.

[11] Strubell, E., Ganesh, A., & McCallum, A. Energy and Policy Considerations for Deep Learning in NLP[C]// Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 3645-3650.

[12] Schwartz, R., Dodge, J., Smith, N. A., et al. Green AI[J]. Communications of the ACM, 2020, 63(12): 54-63.