中国矿业大学计算机学院

2019 级本科生课程报告

课程时间	应用软件开发实践
报告时间	2022 年 7 月 6 日
学生姓名	胡钧耀　黄凯
学　　号	06192081　06192103
专　　业	计算机科学与技术
任课教师	张艳梅

《应用软件开发实践》评分表

序号	课程教学目标	考查方式与考查点	占比	得分
1	**目标 1**：能够采用结构化方法或面向对象方法分析系统需求。	通过学生答辩及软件验收情况，考察其知识熟练应用程度。考察撰写的报告和设计文稿与原有业务要求的贴近度，描述的清晰性、完整性、无歧义。	30%	
2	**目标 2**：综合考虑设计、测试、维护，对设计方案进行优化，开发满足系统需求和约束条件的软件系统、模块或算法流程。	通过学生答辩及软件验收和设计文档，考察学生是否开发完成了满足系统需求和约束条件的软件系统、模块或算法流程。	30%	
3	**目标 3**：熟悉软件开发过程，具有系统的工程研究与实践经历。	通过答辩，考察学生需求分析、方案设计、详细设计、编码、测试等各环节中对于软件开发和管理技术的综合应用情况。	15%	
4	**目标 4**：掌握软件需求分析、设计、编码、测试等环节的常用技术和工程开发工具。	通过答辩，考察学生在分析、设计、编码和测试过程中，对需求分析、软件设计、源代码版本管理、软件测试等计算机辅助软件工程工具的使用情况。	15%	
5	**目标 5**：理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。	通过应用软件开发综合实训环节的选题和设计文档，考察学生是否具有良好的法律意识、社会公德和社会责任感，是否理解并遵守计算机职业道德和规范。	10%	
总分				

评阅人：

日　期：

摘　要

随着移动互联网的迅速发展，越来越多的信息更加被偏向于在官方微博、朋友圈、等互联网网络平台进行发布，人们在日常生活和工作生活中被海量的文本信息包围。如何从海量文本信息中快速而准确的获取人们所需要的内容，成为了当今社会上一种非常迫切的需求。自动文本摘要技术是自然语言处理领域的一项子任务，可以将原来的长文本文档转化为简短的概要描述。

目前，常用的自动文本摘要技术有两种主流方法，一种是抽取式文本摘要，这种方法主要是基于语言学的方法，侧重于关注文本的词法、语法和语义信息等，通过对原文本的句子依据重要度算法进行重新排序，抽取出最重要的一条者几条句子组合成为一段摘要。虽然抽取句子作为摘要的方法在技术上易于实现，应用领域广泛，但生成的文摘不简洁、不连贯、内容不全面。另一种是生成式文本摘要，这种方法使用了目前最流行的深度学习方法，通过神经网络学习文本特征，自动生成摘要。本文将两种方法结合，提出了面向党建领域新闻数据的BERT+UniLM 算法模型。

本项目基于中国大学生第十一届软件杯的 A9 赛题——智能创作平台，主办方提供的新闻文本主要是面向党建领域的新闻数据。系统的实现总体架构主要包括新闻标题生成系统由数据处理模块、算法模型模块和应用服务模块，其中提出的 BERT+UniLM 算法模型是系统的核心支柱。系统算法模型在使用双向循环网络提取新闻文本特征的基础上，添加了多端自注意力机制，旨在从多层次提取文本特征信息，引入 Pre-training 预训练 +Fine-tunning 微调机制有效提高了新闻标题的质量和可读性。最终，训练好的算法模型提供 API 接口，由 Django 开发的网站进行调用，方便人们使用。在构建系统时严格遵循软件工程的思想，进行了需求分析、概要设计、详细设计、编码实现。

关键词：智能创作，BERT+UniLM，注意力机制，生成式摘要，Django 开发

Abstract

With the rapid development of mobile Internet, more and more information is released on the official microblog, circle of friends, and other Internet network platforms. People are surrounded by a large amount of text information in their daily life and work life. How to quickly and accurately obtain the content people need from the massive text information has become a very urgent need in today's society. Automatic text summarization technology is a subtask in the field of natural language processing, which can transform the original long text document into a short summary description.

At present, the commonly used automatic text summarization technology has two mainstream methods. One is extracted text summarization. This method is mainly based on linguistics, focusing on the lexical, grammatical and semantic information of the text. By reordering the sentences of the original text according to the importance algorithm, the most important sentences are extracted and combined into a summary. Although the method of extracting sentences as abstracts is easy to implement in technology and has a wide range of applications, the generated abstracts are not concise, coherent and comprehensive. The other is generative text summarization, which uses the most popular deep learning method to learn text features through neural network and automatically generate summarization. This paper combines the two methods and proposes a bert+unilm algorithm model for news data in the field of Party building.

This project is based on the A9 competition topic of the 11th software cup for Chinese College Students - intelligent creation platform. The news text provided by the sponsor is mainly news data in the field of Party construction. The overall architecture of the system mainly includes the news title generation system, which consists of data processing module, algorithm model module and application service module. The proposed bert+unilm algorithm model is the core pillar of the system. On the basis of extracting news text features using a two-way cyclic network, the system algorithm model adds a multi-end self attention mechanism to extract text feature information from multiple levels. The pre training + fine tuning mechanism is introduced to effectively improve the quality and readability of news headlines. Finally, the trained algorithm model provides an API interface that can be invoked by a Web site developed by Django, making it easy for people to use. In the construction of the system strictly follow the idea of software engineering, requirements analysis, outline design, detailed design, coding implementation.

Keywords: BERT+UniLM, attention mechanism, generative summarization, Django development

目　录

一、绪论

1.1 研究背景

随着近几年互联网的发展，网络中的文本信息资源的数量呈现指数级增长。根据中国互联网协会发布的《中国互联网发展报告 2019》，截至 2018 年底，我国网页总数量已达到 2816 亿，这些网页中所包含的信息覆盖了社会生活的各个领域。然而这其中也包含了海量不规范的文本信息，例如市场上众多移动客户端中自媒体文章的“标题党”，以及互联网中大量无标题的评论、微博等，这些不规范的文本信息给人们带来了巨大的信息过载压力。如何从海量文本信息中快速而准确的获取人们所需要的内容，成为了当今社会上一种非常迫切的需求。为这些不规范的文本信息生成一条简洁、切合原文表达的标题可以缓解信息过载压力，提高工作效率并为相关部门监察社会舆论提供帮助。

实际工作中采用传统的人工总结编写标题的方式在耗费大量时间、人力成本的同时难以应对每天产生的海量不规范文本。标题生成是以文本内容作为输入，以标题作为输出的一种文本摘要任务的变体，因此通过标题生成技术可以高效、经济地解决这一问题。

自动文本摘要技术是自然语言处理领域的一项子任务，可以将原来的长文本文档转化为简短的概要描述。目前，常用的自动文本摘要技术有两种主流方法，一种是**抽取式文本摘要**，这种方法主要是基于语言学的方法，侧重于关注文本的词法、语法和语义信息等，通过对原文本的句子依据重要度算法进行重新排序，抽取出最重要的一条者几条句子组合成为一段摘要。虽然抽取句子作为摘要的方法在技术上易于实现，应用领域广泛，但生成的文摘不简洁、不连贯、内容不全面。另一种是**生成式文本摘要**，这种方法使用了目前最流行的深度学习方法，通过神经网络学习文本特征，自动生成摘要。

1.2 研究意义

1.2.1 高效提取内容

随着近几年文本信息的爆发式增长，人们每天能接触到海量的文本信息，如新闻、博客、聊天、报告、论文、微博等。从大量文本信息中提取重要的内容，已成为我们的一个迫切需求，而自动文本摘要则提供了一个高效解决方案，帮助人们从文本中快速提取出重要的内容。

1.2.2 审查文章主旨

近年来，网络新闻的“震惊体”文不对题现象层出不穷，严重影响了高效准确的信息检索以及用户体验。

例如，人民网作为世界十大报纸之一《人民日报》建设的以新闻为主的大型网上信息发布平台，每日收稿量巨大。然而，这些稿件新闻内容质量良莠不齐，其中不乏一些选取华丽标题却缺乏实质内容的“标题党”文章浪费了审稿人员大量的时间精力，如何快速的获取文章的真实主旨成为了审稿人员的重要能力。除了人工的加强审稿能力外，人工智能技术，尤其是自然语言处理技术将极大的助力新闻主旨检索。人工智能技术通过标题再生成的方法，生成最贴切新闻文章内

容的新闻标题，未来希望通过此方法减少人工排查“标题党”的时间。

1.3 文本摘要研究现状

20 世纪 50 年代，来自美国 IBM 的 Luhn（1958）首次提出使用计算机自动生成文本摘要，从此揭开了自动文本摘要研究的序幕。在此之后的几十年中，随着计算机网络时代的到来，自动文本摘要得到迅速发展和提高。1993 年 12 月，历史上第一次以自动文摘为主题的国际研讨会在德国 Wadern 召开。1995 年 9 月，国际著名期刊 Ingormation Processing and Management 出版了一期主题为“Summarizing Text”的专刊，标志着自动文本摘要的时代已经到来。自动文本摘要技术作为自然语言处理领域的研究热点，由此进入了人们关注的视野。自从 Luhn（1958）发表了第一篇有关自动文本摘要的文章以来，国外就开始了自动文本摘要研究的序幕，研究时间比较长。由于中文和其他国外语言差别比较大，在词汇量方面要多，语法和语义方面更加丰富，最根本的是中文以词为信息的载体，而不是字。因此，我国对自动文本摘要的研究起步比较晚。国内外对自动文本摘要技术的研究基本上都是从最初的文本词法、语法分析阶段，到通过特征提取的机器学习阶段，再到如今基于神经网络的深度学习模型阶段。总体上，自动文本摘要技术被划分为两种主流的方法：抽取式文本摘要方法和生成式文本摘要方法。

1.3.1 抽取式摘要研究现状

H.P.Luhn 首先提取原文每个句子中的词语，然后统计这些词语出现的频率，选取包含最多重要词语的句子作为原文的摘要，这是文本抽取式摘要最早的应用。Brandow R 等人通过特征计算得到句子权重，最后将权重最高的句子当作文本摘要。Mihalcea R 等人使用无监督的算法 TextRank，基于图排序将句子当作顶点，将句子之间的相似度当作边，通过迭代计算来得到顶点的重要性，最终将得到的重要句子作为文本摘要。

随着深度学习的发展，在抽取式摘要方面也出现了许多应用，Nallapati R 等人提出一种 SummaRuNNer 模型，模型使用基于词级别和句子级别的循环神经网络将文本摘要转换为一个二分类问题，抽取文本中的重要句子来得到最终的摘要。Isonuma 等人将抽取式摘要任务与文档分类任务相结合，让抽取类任务与文档分类任务相互促进，进一步提高模型抽取关键句子的能力。Yan Liu 等人将 BERT 用于抽取式摘要生成，对原文句子进行处理后输入 BERT 中得到每个句子向量，之后利用 Transformer 来生成需要抽取的摘要。

1.3.2 生成式摘要研究现状

基于神经网络的自动文摘被越来越多地应用在文本摘要中。首先是 Sutskever 等人提出的序列到序列（seq2seq）模型，由一个编码器和一个解码器组成。接着 Rush 等人将注意力机制与 seq2seq 模型结合，解码器通过解码状态与编码状态的注意力分数从编码器中提取信息。Hu 等人创建了一个大规模的中文摘要数据集 LCSTS，并且分别以词为单位和以字为单位对数据进行处理，然后使用结合注意力机制的 seq2seq 模型在两种处理方式上进行了实验对比，为中文摘要模型的发展奠定了基础。Chopra 等人使用循环神经网络（Recurrent Neural Network）代替神经网络语言模型（Neural Network Language Model，NNLM），结合注意力机制进一步提高了生成摘要的质量，这个模型之后被当作基础模型。Lopyrev 等人使用长短期记忆网络（Long Short-Term Memory，LSTM）代替 RNN，缓解了网络

对之前信息的遗忘，但是模型依然存在词汇表限制（out-of-vocabulary，OOV）问题。Vinyals 等人提出 PointerNetwork，通过指针开关决定在解码时刻从原文中复制一个词，还是根据词汇表生成一个词，缓解了 OOV 问题。Lin 等人提出一种门控卷积单元对文本进行全局编码，门控神经单元由 CNN 和 self-attention 组成，使用 CNN 来提取 n-gram 特征，使用 self-attention 来学习每个时间步信息与全局信息之间的关系。Paulus 等人引入了强化学习进行文本摘要生成，考虑到了将 ROUGE 评价指标引入到训练中，但由于 ROUGE 指标不可导，因此，使用强化学习将 ROUGE 指标加入到模型的训练中，对最后摘要的生成起到了很好的作用。Li 等人引入了句子级别的重复机制来解决生成摘要中重复信息过多的问题。Cao 等人提出了使用参考摘要来指导模型生成摘要，将参考摘要作为外部知识来辅助模型训练。徐如阳等人提出一种自注意力卷积门控单元编码方法，通过卷积单元来挖掘局部信息，多头注意力机制学习上下文信息，最后通过强化学习对 ROUGE 指标进行优化，取得了很好的效果。

1.4 论文研究的主要内容

本论文旨在使用自然语言处理领域的自动文本摘要技术实现**智能创作平台系统**。系统在服务器端自动处理纷乱复杂的新闻文本数据，然后利用深度学习算法对处理好的新闻文本数据实行智能化标题、摘要、关键词生成，并在前端进行反馈，为“智能创作平台”的使用者提供便利服务。本课题根据项目需求进行分析，对深度学习做了深入的技术解剖。最终，提出了基于 transformer 模型，采用 Bert+UniLM 作为系统的整体架构，并将整体框架详细的划分为多个子模块。为了实现一款生成质量优质、文本内容流畅的智能创作平台系统，本论文主要研究的内容如下：

1. 新闻数据处理相关技术。文本的数据格式是整个系统的基础，本文在将中文数据转化为计算机识别的数据时使用了 word2vec 词向量转换技术。同时，在数据输入系统前还使用基于图模型的文本排序算法 TextRank 算法，用来预处理过长的新闻文本，解决算法模型因无法输入过长文本导致生成新闻标题质量不佳的问题。

2. 提出一种利用 BERT 进行内容标题生成的方法。由于文本为长文本，使用深度学习方法对长文本进行处理时存在编码器对长文本理解不完全以及对模型训练硬件要求过高的问题，因此本文首先将输入文本每个句子前后分别添加特殊符号［CLS］与［SEP］，将处理过后的数据输入到 BERT 中，利用 BERT 得到［CLS］位置的向量作为句子向量，由于 BERT 内部使用多头注意力机制，因此可以更好地得到句子的具体上下文信息，得到更加精确的句子向量，相比其他方法得到更高的 ROUGE 分数值。最终将抽取的内容作为之后标题生成的数据集用于模型训练与测试。

3. 实际环境中自动文本摘要系统的实现与部署。本课题在需求分析和算法模型构建的基础上，还进行了系统的实现与部署。使用 Python 对党建新闻做数据处理后，通过以深度学习框架 keras 架构实现的算法模型进行训练和预测，将训练好的模型部署在后台服务器端。前端使用 Django 框架搭建，通过 API 格式调用后端反馈数据。系统最终部署在购买的服务器中。

1.5 论文的组织结构

本文主要分为七个部分，具体的内容组织结构如下：

第 1 章 绪论。本项目基于自动文本摘要技术，主要介绍了本课题的研究背景及意义，根据自动文本摘要技术两种主流实现方式，详细分析了国内外的研究现状，最后介绍了本课题的主要研究内容以及论文的组织结构。

第 2 章 需求分析。主要介绍了项目背景以及设计目标。然后根据系统实现目标，从功能性和非功能性两个方面对系统进行了详细的需求分析。经过对智能创作品平台系统的需求与分析进行详细的介绍，为下一步系统的概要设计与详细设计提供了明确的目标和清晰的规划。

第 3 章 概要设计。本章主要介绍了智能创作品平台系统的概要设计。然后根据系统的需求分析，确定了系统的架构设计，即自底向上分别为数据处理层、算法模型层和应用服务层，并详细说明了每个层级需要完成的任务。接着，根据系统架构设计，进行了系统的功能模块设计，给出了系统的业务流程图。最后给出了系统的数据库设计，为下一步的详细设计以及编码实现提供了依据。

第 4 章 详细设计。详细设计是系统编码实现的重要依据，本章从四个角度规划了系统的实现：用户登陆注册、文章表增删改查、首页数据可视化、文章标题生成。通过绘制出系统的控制流程图、时序图以及算法流程图，严格遵循软件工程的思想，给出了系统的详细设计描述。

第 5 章 算法模型与原理。首先介绍了算法模型的建模思路，接着描述了抽取式算法 TextRank 的流程与特点。模型部分，从 seq2seq 开始介绍，解释了 RNN+LSTM 的模型思想，接着介绍了目前应用广泛的 tansformer 模型,并将 RNN 与 Transformer 进行了比较。之后提出了项目应用的模型：BERT+UniLM 模型。最后应用 ROUGE 以及 BLEU 评价指标对模型的效果进行评价。

第 6 章 智能创作系统的编码与实现。从用户管理、文章管理、用户主页、AI 模型生成四个角度，讲解了系统的前端页面代码、后端 Django 代码以及与对数据库进行增删改查的代码，并展示了对应的前端页面的图片。

第 7 章 总结与未来展望。主要对本文所研究内容进行总结以及未来的发展提出想法。

1.6 本章小结

本章主要介绍了项目的研究背景及意义，以及国内外对自动文本摘要技术的研究现状。同时，本章还概述了项目的主要研究内容，以及整篇论文的组织结构。本章对项目进行了系统性疏理，为后面模型算法的选择和系统实现打下了基础。

二、需求分析

2.1 项目背景及设计目标

2.1.1 项目背景

目前，标题和摘要主要由人工完成，人工理解整个输入文本后，为输入文本确定对应的文本标题和摘要。由于人工处理的方式需要经历人工文本理解，受主观性影响较大，可能会由于人的主观意识导致对文本类型误判，因此使得生成的文本标题和摘要不够准确，并且人工处理的方式效率较低。

随着互联网技术和大数据技术的快速发展，越来越多的研究机构和项目平台提供使用深度学习技术的在线系统服务，比如中科院计算所的在线 NLPIR 中文分词系统、清华大学的社会媒体关键词抽取演示系统、哈尔滨工业大学的 KGB 知识图谱系统和东北大学的小牛翻译开放平台等。

2.1.2 设计目标

本项目基于第十一届“中国软件杯”的 A9 赛题——智能创作平台，配合赛题给定高质量训练数据，使用深度学习算法生成契合文章主题内容和有吸引力的优质标题和摘要，并开发出一套智能创作 WEB 系统，帮助人类提升写作的效率和质量。

本系统采用了时下日渐成熟的深度学习技术，根据自然语言处理领域下的自动文本摘要任务，并成功的应用在时评网的新闻数据上。人们可以通过门户网站调用训练好的算法模型，对长文本的新闻生成标题、摘要、关键词。本系统的设计目标有

（1）配合主办方提供的高质量训练数据，保证系统的算法模型有足够的数据进行充分训练，避免算法模型的欠拟合导致新闻标题信息不准确、不完整。

（2）构建 Bert + UniLM 算法模型对训练集进行微调训练。

（3）构建算法模型的 API 接口，通过网站直接调用，方便人们操作使用，满足人机交互简易性的需求。

2.2 系统功能性需求分析

通过对项目背景的研究和对项目的需求进行分析，智能创作平台系统主要包含数据处理的功能需求、算法模型的功能需求和应用服务的功能需求，具体的需求分析如下：

2.2.1 数据处理的功能需求

数据处理部分主要负责提供算法模型训练所需要的标准数据，是整个系统的基础所在。党建新闻标题系统的目标是为用户提供党建新闻的标题生成功能，因此需要党建领域的新闻数据集。使用爬虫在人民日报网站上爬取党建新闻，对得到的源新闻文本数据进行人工处理，并构建党建新闻语料库。数据处理主要包括以下三个基本功能：数据收集、数据处理和 TextRank 算法预抽取。数据收集功能负责获取人民日报的党建新闻数据，数据清理功能负责对源新闻数据进行中文分词，过滤标点符号、特殊符号，去除常见词、停用词等一系列文本清理。TextRank 算法预抽取负责对长度过长的党建新闻数据做新闻重要句的预抽取，重新组成符

合要求的新数据。最终把处理后的新闻数据组成党建新闻语料库，供上层算法模型层训练测试使用。

2.2.2 算法模型的功能需求

算法模型是整个智能创作平台系统的核心组成部分，主要包括以下四个基本功能：新闻文本特征提取、关键信息指向、关键摘要抽取和新闻标题生成功能。

2.2.3 应用服务的功能需求

智能创作平台系统主要是通过网站的门户界面提供应用服务，对用户更加人性化。应用服务部分主要包括创建门户网站和构建算法模型的 API 接口两部分。门户网站的创建应该包含用户输入文本框、功能提交和生成结果展示等基础的用户前端交互服务；构建算法模型的 API 接口应该包含服务器后端捕获用户输入数据、预测结果传递到门户网站的表单等基础的前端与后端交互服务。

2.2.4 需求分析用例图

如图 2.2-1 所示。系统功能主要在于满足用户的文章发布获取以及进行文章的标题、摘要、关键词的自动提取，对于管理员和用户还要能对用户信息进行修改，还要能对数据库中的文章进行增删改查，最重要的是用户可以调用标题、摘要、关键词的自动生成的 API。

其角色主要为：游客、用户、管理员、用户信息数据库、文章信息数据库、算法模型。相关功能主要有：对用户信息进行修改，完成注册登录功能；对文章进行增删改查、对数据集进行增删改查；智能生成标题、摘要、关键词。

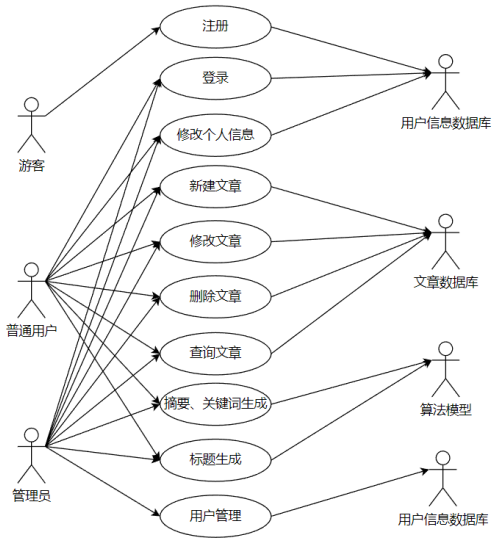

图 2.2-1 智能创作平台系统用例图

2.2.5 顶层数据流图

根据系统的需求分析用例图，画出顶层数据流图如图 2.2-2 所示：

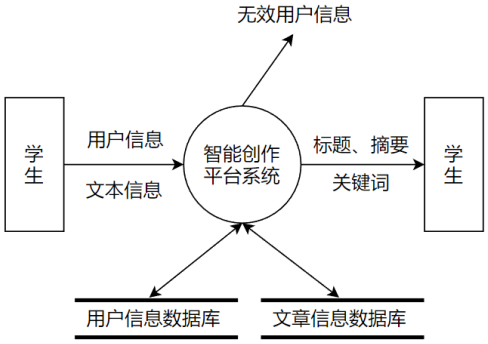

图 2.2-2 智能创作平台顶层数据流图

2.2.6 第二层数据流图

根据顶层数据流图，做出第二层数据流图如图 2.2-3 所示：

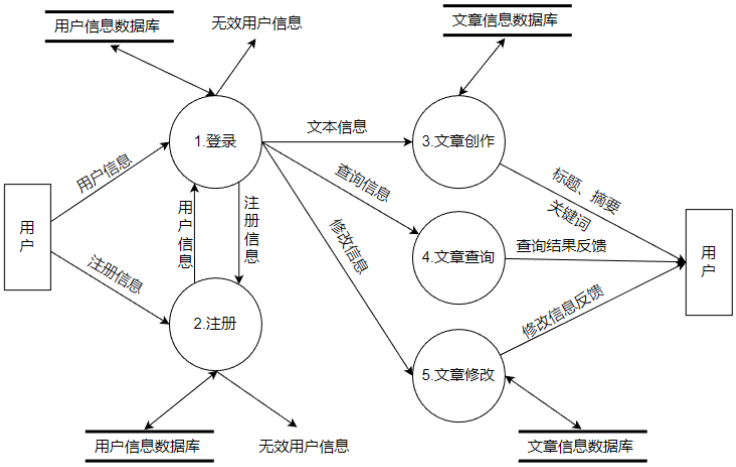

图 2.2-3 智能创作系统第二层数据流图

2.2.7 第三层数据流图

根据第二层数据流图，画出第三层数据流图如下（以登录与文章创作为例）：

用户登录第三层数据流图如图 2.2-4 所示：

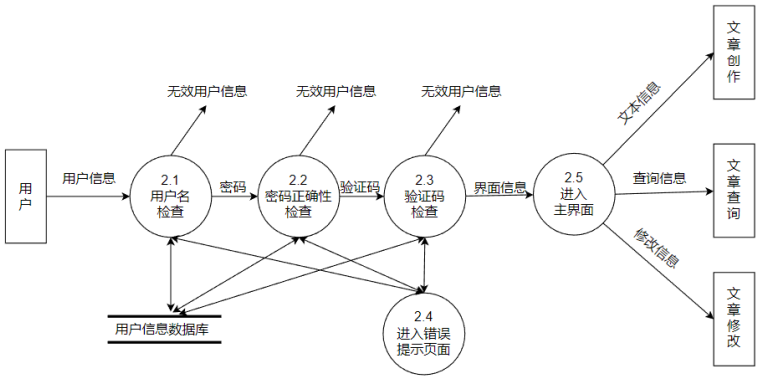

图 2.2-4 用户登录第三层数据流图

文章创作第三层数据流图如图 2.2-5 所示：

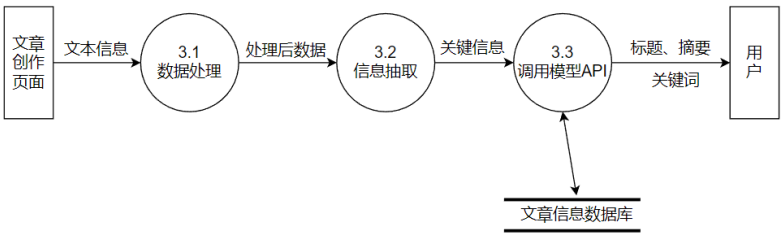

图 2.2-5 文章创作第三层数据流图

2.3 系统非功能性需求分析

智能创作品平台系统在必要的功能性需求基础上，还需要考虑到非功能性需求。非功能性需求是保证系统能够平稳运行的一些因素分析，主要包括实用性需求分析、高内聚低耦合性需求分析、易用性需求分析、可扩充性需求分析和系统性能需求分析等。智能创作品平台系统主要的非功能性需求分析具体如下。

2.3.1 实用性需求分析

利用深度学习技术构建算法模型并搭建用户 Web 平台，为用户提供新闻标题自动生成功能，使得人们能够通过移动终端设备快速的、简洁的、随时随地的获取文本中的主要内容。

2.3.2 高内聚低耦合性需求分析

内聚性是指系统内部功能关系聚集、关联的程度，耦合性是指系统结构中各个功能模块之间相互联系的紧密程度。如果不同功能模块之间联系紧密，那么在提出新的系统功能需求时就会出现“牵一发而动全身”的局面，一处更改，处处更改，导致系统代码的可维护性特别差。本文设计的智能创作品平台系统各个功能模块层相对独立，每个功能模块负责相对独立的功能，功能模块内部元素之间的联系保持紧密结合。功能模块之间通过接口交互，而不依赖其他功能模块。系统的高内聚低耦合性有效提高了代码的重复利用率。

2.3.3 易用性需求分析

易用性需求分析要求系统操作简单，用户无需参考复杂的文档，通过系统网站界面即可实现所需要的任务功能。本系统的开发过程中遵循高内聚低耦合原则，用户无需操作后端的算法模型源码，只在前端页面进行操作就可以完成系统功能。系统网站界面的模块化，输入框、输出框和操作按钮简单明了，十分符合人机交互原理。

2.3.4 可扩充性需求分析

可扩充性指当提出新的功能需求时，系统扩充功能简易方便。本系统开发过程中遵循高内聚低耦合原则，各功能模块分别根据其独立的功能创建接口。在扩充新功能时，只需在相应的功能模块实现后定义好接口供其他模块调用，在扩充过程中不会影响到其他功能模块的正常运行。

2.3.5 系统性能需求分析

系统性能性需求是指系统能够满足用户任务需求的同时，还能够稳定、高效的保持正常运行。用户正常使用系统的功能时，系统能够及时做出反馈。

2.4 本章小结

本章主要介绍了智能创作品平台系统的项目背景以及设计目标。然后根据系统实现目标，从功能性和非功能性两个方面对系统进行了详细的需求分析。在功能性需求分析确定了系统的三个主要需求：数据处理的功能需求、算法模型的功能需求和应用服务的功能需求。非功能性需求分析中主要分析了系统开发时保证能够安稳运行的五个主要需求：实用性需求分析、高内聚低耦合性需求分析、易用性需求分析、可扩充性需求分析和系统性能需求分析。经过对智能创作品平台系统的需求与分析进行详细的介绍，为下一步系统的概要设计与详细设计提供了明确的目标和清晰的规划。

三、概要设计

3.1 系统架构设计

本文旨在实现基于自动文本摘要的智能创作品平台系统，为上层人员直接提供新闻文本的标题、摘要、关键词。系统使用 TextRank 算法对原始新闻数据做抽取预处理，系统的核心模型利用了流行的深度学习方法，基于多端注意力机制构建了 Bert+UniLM 算法。并将算法模型部署在服务器端，由 Web 界面直接调用后台接口，用户人员可以通过门户网站方便使用。智能创作平台系统由数据处理层、算法模型层和应用服务层三部分组成，系统总体架构如图 3.1-1 所示。

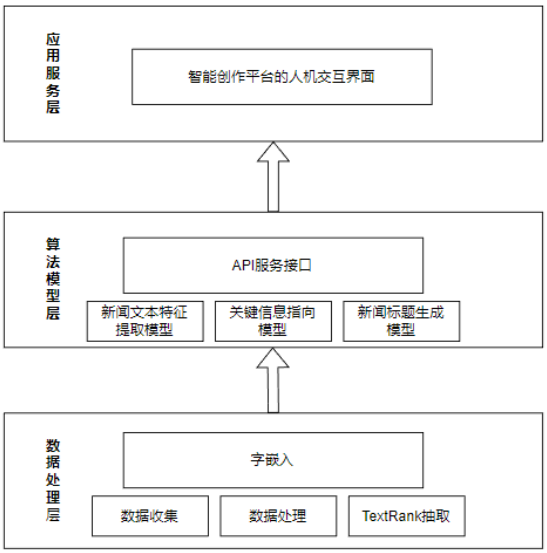

图 3.1-1 智能创作平台系统总体架构图

整个系统架构自下往顶的结构共三层：

数据处理层：位于系统架构的最下面一层，是整个系统的基础。数据处理层包括数据收集、数据处理和使用 TextRank 算法对长文本类型的党建新闻预抽取三部分，主要负责新闻数据的预处理功能，包括中文分词，过滤标点符号、特殊符号，去除常见词、停用词等，是连接算法模型层的输入接口。

算法模型层：位于中间层，是整个系统的核心模块。算法模型层的主要功能是自动文本摘要算法的选择与实现，包括新闻文本特征提取模块、关键信息指向模块和新闻标题生成模块三部分。算法模型层根据组合算法的功能又可以看作是三个阶段，即新闻文本特征阶段、关键信息重组阶段和新闻标题、摘要生成阶段。

应用服务层：位于最上一层，是人与系统算法模型进行交互的界面。算法模型经过党建新闻数据的多轮训练后，使用 Django 部署在服务器端。为了增强人机交互体验，设计门户网站界面，通过 Django 传递的 API 接口调用服务器端的模型。人们只需要在系统门户网站的文本框输入长文本类型的党建新闻，就可以得到算法模型生成的新闻标题。

3.2 功能模块设计

根据党建新闻标题生成系统的总体架构分析可知，系统由数据处理层、算法模型层和应用服务层三部分组成。数据处理层收集并清理数据，做模型输入前的一些预处理工作；算法模型层对数据处理层的输入使用多端注意力机制做特征提取，使用指针网络指向新闻文本中的重要信息，融入到生成的新闻摘要中，保证系统生成流畅、精简的新闻标题，是整个系统设计的核心模块；应用服务层将训练好的模型部署在设计好的 Web 门户网站，方便人们直接网页操作。系统详细功能设计图如图 3.2-1 所示。

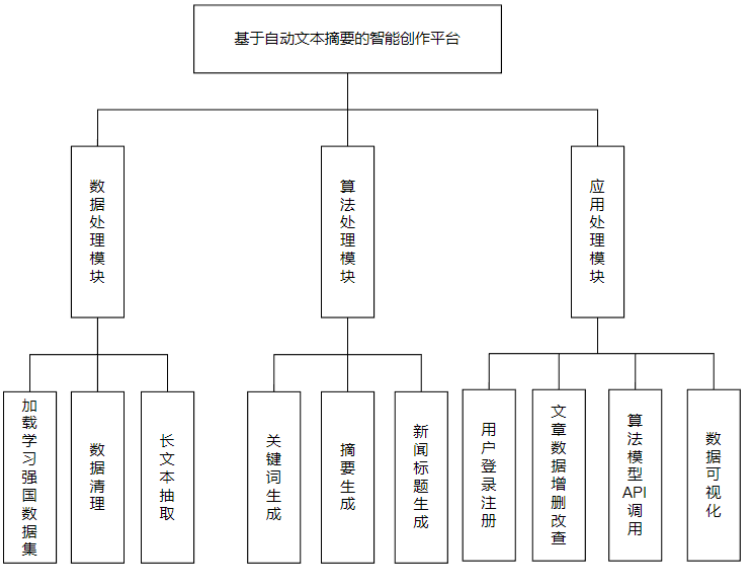

图 3.2-1 智能创作系统详细功能设计图

系统流程图如图 3.2-2 所示。

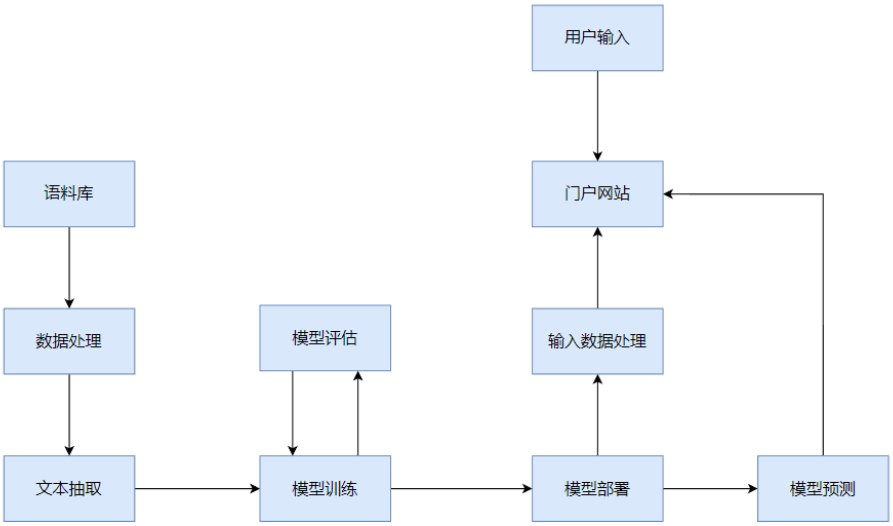

图 3.2-2 智能创作系统系统流程图

3.3 数据库设计

3.3.1 用户数据库

Auth 模块是 Django 自带的用户认证模块：我们在开发一个网站的时候，无可避免的需要设计实现网站的用户系统。此时我们需要实现包括用户注册、用户登录、用户认证、注销、修改密码等功能。它内置了强大的用户认证系统——auth，它默认使用 auth_user 表来存储用户数据。auth_user 表的设计如表 3.3-1 所示。

表 3.3-1 用户数据库 auth_user 设计

字段列表	字段描述	主键	类型	长度	说明
id	id	是	int	11	非空
password	密码		varchar	126	非空
last_login	最后登录时间		datetime	6	
is_superuser	超级管理权限		tinyint	1	非空
username	用户名		varchar	150	非空
first_name	名		varchar	150	非空
last_name	姓		varchar	150	非空
email	邮箱		varchar	254	非空
is_staff	管理员权限		tinyint	1	非空
is_active	登陆权限		tinyint	1	非空
date_joined	注册时间		datetime	6	非空

3.3.2 文章数据库

Django 可以根据自身需要，创建新的数据库，例如在创建 writing 应用后，需要一个存储文章数据的数据库，在相应 model 文件中编写 Article 类，然后使用数据库迁移命令，即可创建相应的数据库 writing_arcitle，数据库还支持多次更新多次迁移，这为拓展应用功能，后期产品迭代提供了很好的基础。我们的文章数据库设计如表 3.3-2 所示。

表 3.3-2 文章数据库 writing_arcitle 设计

字段列表	字段描述	主键	类型	长度	说明
id	id	是	bigint	20	非空
title	文章名		varchar	50	非空
abstract	摘要		longtext	0	非空
body	正文		longtext	0	非空
author	作者		varchar	10	非空
create_time	创建时间		datetime	6	非空
update_time	最后更新时间		datetime	6	非空
category	分类		varchar	5	非空
keyword	关键词		varchar	40	非空

3.3.3 其他数据库

Django 的 Auth 模块为我们提供了其他的与用户认证、数据库连接有关的数据库，如和 session 数据存储有关的 django_session 表、和数据库迁移更新记录有关的 django_migrations 表、存储所有的关系表的 django_content_type 表，但因为是模块自带的数据库，并不是软件主要设计内容，在此不详细展开说明。

3.4 本章小结

本章主要介绍了智能创作品平台系统的概要设计。然后根据系统的需求分析，确定了系统的架构设计，即自底向上分别为数据处理层、算法模型层和应用服务层，并详细说明了每个层级需要完成的任务。接着，根据系统架构设计，进行了系统的功能模块设计，给出了系统的业务流程图。最后给出了系统的数据库设计，为下一步的详细设计以及编码实现提供了依据。

四、详细设计

4.1 用户登陆注册

在用户登录部分，沿用实验三的设计，主要完成进入管理系统的安全性、鲁棒性。在逻辑判断用户角色身份时，其权限控制流程如图 4.1-1 所示。

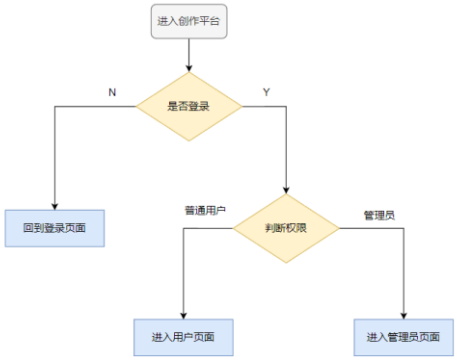

图 4.1-1 用户角色权限控制流程

在用户注册部分，沿用实验二的设计，主要完成注册合法性的判定。其合法性控制流程如图 4.1-2 所示。

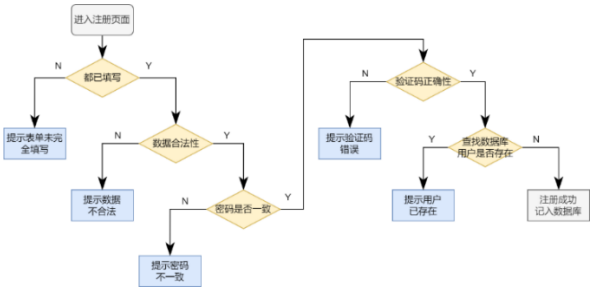

图 4.1-2 用户注册合法性控制流程

在用户登录部分，沿用实验二的设计，主要完成进入登录合法性的判定。其合法性控制流程如图 4.1-3 所示。

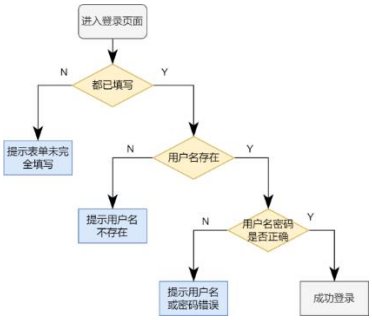

图 4.1-3 用户登录合法性控制流程

4.2 文章表增删改查

文章表部分的主体逻辑也是对于数据表的增删改查，在总体逻辑上与实验三相近，重点需要关注其相互之间的调用协作关系，其增、删、改、查四种操作的时序协作图如下图所示。

新建文章时序协作图如图 4.2-1 所示。

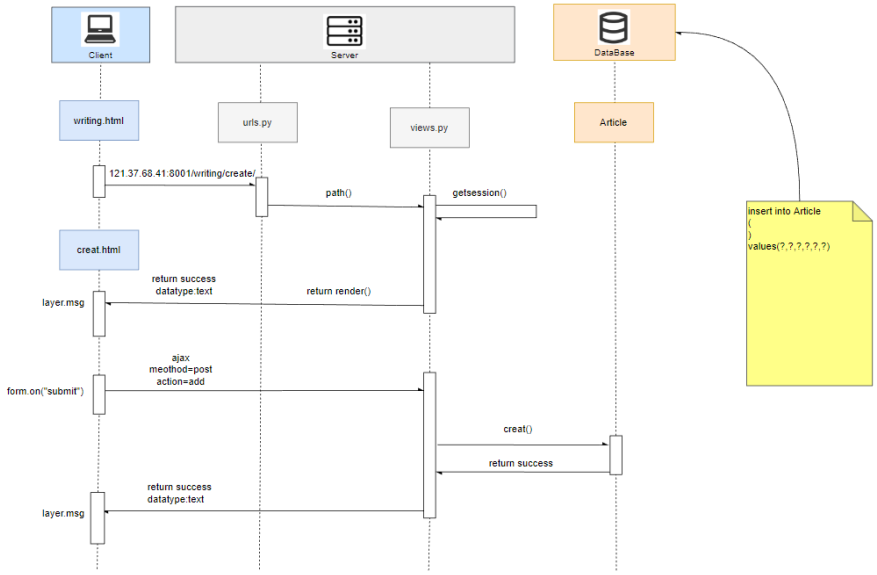

图 4.2-1 新建文章时序协作图

删除文章时序协作图如图 4.2-2 所示。

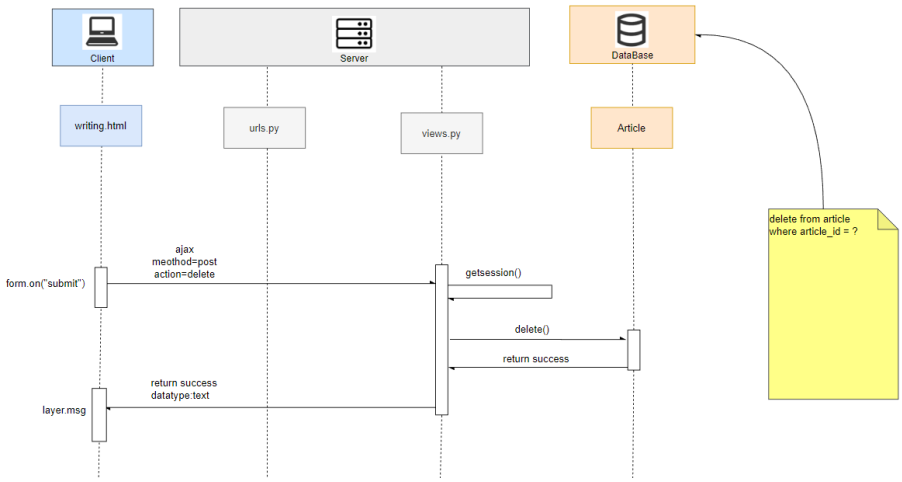

图 4.2-2 删除文章时序协作图

修改文章时序协作图如图 4.2-3 所示。

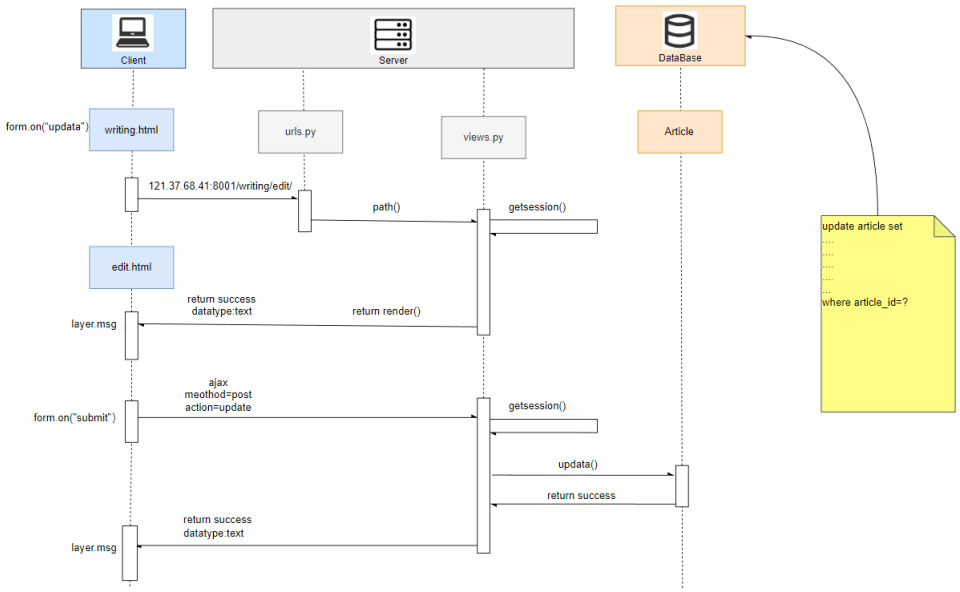

图 4.2-3 修改文章时序协作图

查询文章时序协作图如图 4.2-4 所示。

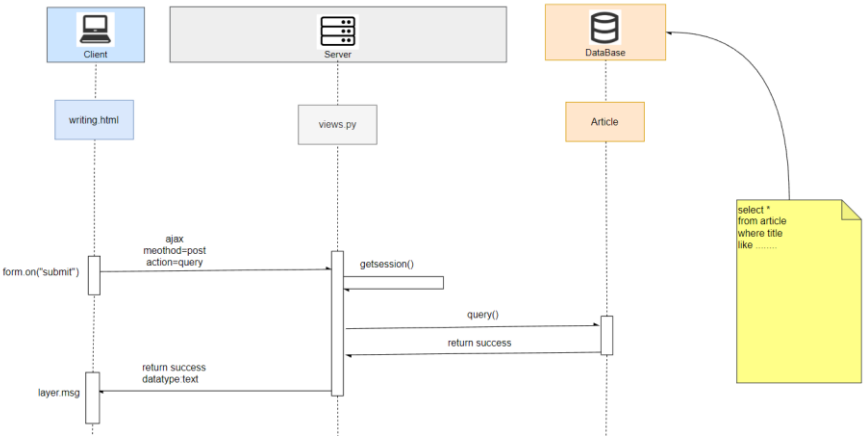

图 4.2-4 查询文章时序协作图

4.3 首页数据可视化

4.3.1 ECharts 数据可视化展示

ECharts 是一个使 JavaScript 实现的开源可视化库，涵盖各行业图表，满足各种需求。相比于使用 Python 的 Matlibplot 等库，它可以实现和用户的动态交互，但数据又需要从数据库中读取才行，因此，我们在前端页面使用 ECharts 展示数据，使用 Django 模板语言从后端传输数据，后端使用 Python 对 ORM 数据库模型读取相应的数据，从而实现数据的可视化展示。如近七天用户发布文章数量、文章发表时间分布、文章类型占比等。

例如要展示近七天用户和全站的发布文章数量，用户数据选用 bar 条形图，全站数据选用 line 折线图，横坐标由后端传输七天日期列表 list_recent_a_week，

纵坐标为用户和全站的数据列表 data_user 和 data_all，后端使用 Article.objects.filter 方法查询数据库并整理数据，最后使用 render 方法传输上述信息到页面。时序图如图 4.3-1 所示。

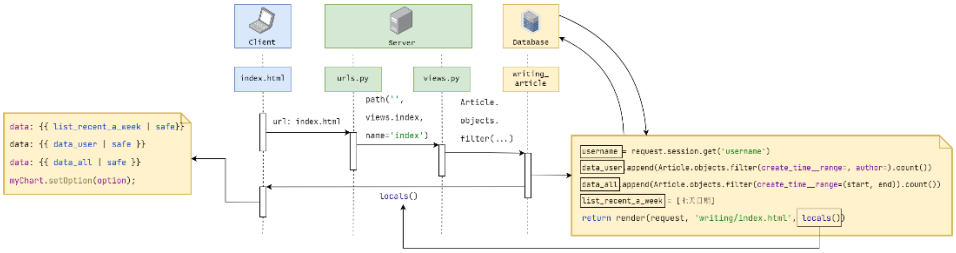

图 4.3-1 ECharts 数据可视化展示功能时序图（以显示用户、全站一周写作量为例）

4.3.2 爬虫数据展示

为了丰富用户主页，增添了网络热门话题榜单卡片，显示百度、B 站、中国矿业大学近期重要话题或者新闻，这里采用 Python 爬虫第三方库 Requests 进行爬虫，其可以通过调用来帮助我们实现自动爬取 HTML 网页页面以及模拟人类访问服务器自动提交网络请求。对于提取到的网页，使用 BeautifulSoup 库进行解析优化，其是 Python 用来解析爬取的网页源代码的一个库，因其码的简洁性，因此应用得比较多。对于解析高度结构化的 HTML，相比于正则表达式，用 BeautifulSoup 能更快速便捷地进行解析和提取。时序图如图 4.3-2 所示。

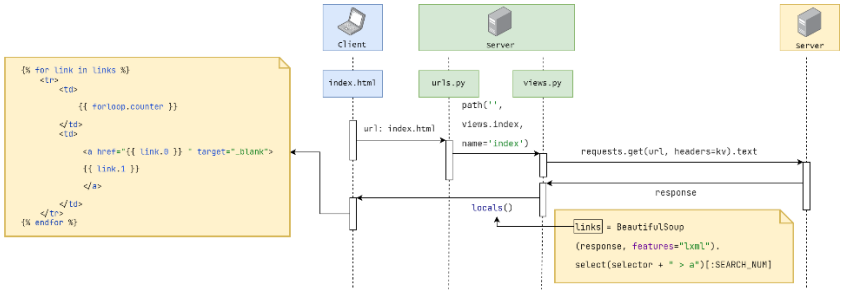

图 4.3-2 爬虫热点榜单数据展示功能时序图

4.4 文章标题生成

文章标题生成算法如图 4.4-1 所示。

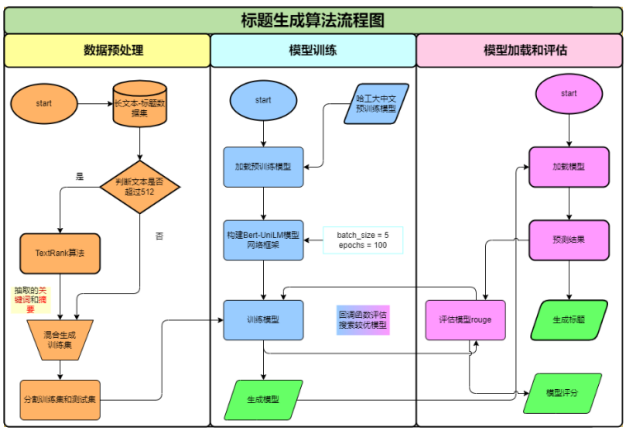

图 4.4-1 标题生成算法流程图

4.5 本章小结

详细设计是系统编码实现的重要依据，本章从四个角度规划了系统的实现：用户登陆注册、文章表增删改查、首页数据可视化、文章标题生成。通过绘制出系统的控制流程图、时序图以及算法流程图，严格遵循软件工程的思想，给出了系统的详细设计描述。

五、算法与模型原理

5.1 算法整体流程

5.1.1 数据分析

这次比赛官方共提供了 5860 个标注样本，以“(原文, 标题)”这样的数据对形式出现，所以我们的模型原则上适用于所有单条样本格式为“(原文, 摘要)”的监督式标题生产任务。

下面是训练数据的一些统计信息：

1、总量：5860；

2、输入：平均字数 2568，字数标准差 1122，最大字数 13064，最小数字 866；

3、输出：平均字数 12，字数标准差 5，最大字数 22，最小数字 2；

4、指标：以词为单位的加权 Rouge。

因此，简单来说这大概就是一个“输入 3000 字、输出 10 字”的文本生成任务，其难度在于两千多的平均长度远远超出了我们平时处理的文本长度。5860 是全部发布的数据集，可以直接跑出还不错的效果，整个模型对数据量的依赖不是特别严重。

5.1.2 样本预览

训练集如图 5.1-1 所示：

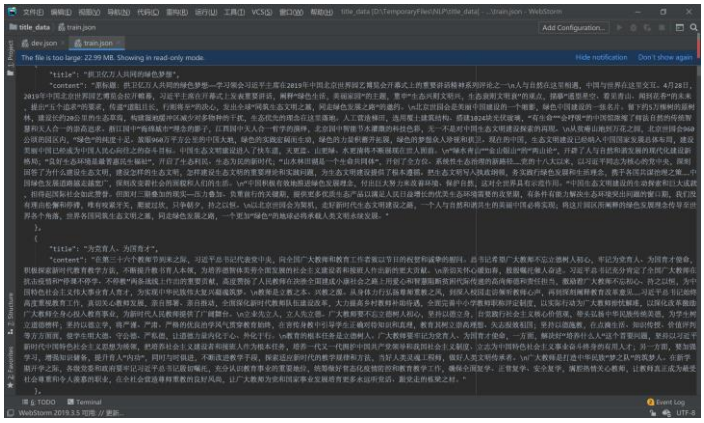

图 5.1-1 训练集展示

训练结果对比展示如图 5.1-2 所示：

图 5.1-2 训练结果对比

5.1.3 建模思路

文本摘要可以大致分为两类——抽取型摘要和生成型摘要：

抽取型摘要：这种方法依赖于从文本中提取几个部分，例如短语、句子，把它们堆叠起来创建摘要。因此，这种抽取型的方法最重要的是识别出适合总结文本的句子。

生成型摘要：这种方法应用先进的 NLP 技术生成一篇全新的总结。可能总结中的文本甚至没有在原文中出现。

综合上述数据特性，我们不难想到应该采取“**抽取+生成**”相结合的方式进行摘要，并配合一些新方法来保证摘要的忠实程度与提升最终的效果。

5.2 抽取式算法 TextRank

5.2.1 概述

TextRank 算法是一种用于文本的基于图的排序算法，通过把文本分割成若干组成单元（句子），构建节点连接图，用句子之间的相似度作为边的权重，通过循环迭代计算句子的 TextRank 值，最后抽取排名高的句子组合成文本摘要。[1]

TextRank 算法主要包括：关键词抽取、关键短语抽取、关键句抽取。

（1）关键词抽取（keyword extraction）

关键词抽取是指从文本中确定一些能够描述文档含义的术语的过程。对关键词抽取而言，用于构建顶点集的文本单元可以是句子中的一个或多个字；根据这些字之间的关系（比如：在一个框中同时出现）构建边。根据任务的需要，可以使用语法过滤器（syntactic filters）对顶点集进行优化。语法过滤器的主要作用是将某一类或者某几类词性的字过滤出来作为顶点集。

（2）关键短语抽取（keyphrase extration）

关键词抽取结束后，我们可以得到的 N 个关键词，在原始文本中相邻的关键词构成关键短语。因此，从 get_keyphrases 函数的源码中我们可以看到，它先调用 get_keywords 抽取关键词，然后分析关键词是否存在相邻的情况，最后确定哪些是关键短语。

（3）关键句抽取（sentence extraction）

句子抽取任务主要针对的是自动摘要这个场景，将每一个 sentence 作为一个顶点，根据两个句子之间的内容重复程度来计算他们之间的“相似度”，以这个相似度作为联系，由于不同句子之间相似度大小不一致，在这个场景下构建的是以相似度大小作为 edge 权重的有权图。

5.2.2 流程

1. 第一步是把所有文章整合成文本数据
2. 接下来把文本分割成单个句子
3. 然后，我们将为每个句子找到向量表示（词向量）。
4. 计算句子向量间的相似性并存放在矩阵中
5. 然后将相似矩阵转换为以句子为节点、相似性得分为边的图结构，用于句子 TextRank 计算。
6. 最后，一定数量的排名最高的句子构成最后的摘要。

主要流程如图 5.2-1 所示。

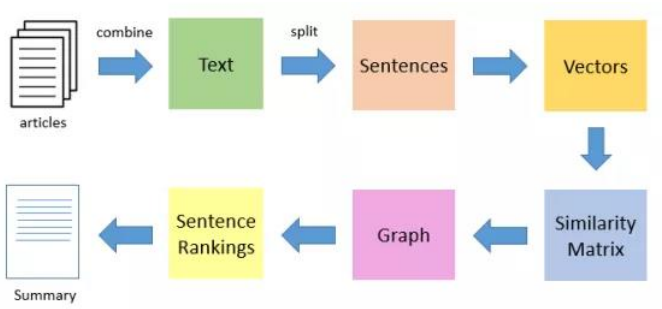

图 5.2-1 TextRank 基本算法流程

关键词提取：图构造完成后，单词的 TR 值计算公式为：

$$WS(V_i)=(1-d)+d*\sum_{V_j\in In(V_i)}\frac{w_{ji}}{\sum_{V_k\in Out(V_j)}w_{jk}}WS(V_j)$$

摘要生成：将文本中的每个句子看作图中的一个节点，句子之间的“链接关系”由句子间的相似性体现。句子相似性有多种计算方式，这里使用一种很简单的方法，计算两个句子共有词比例。句子相似性公式：

$$\text{Similarity}(S_i,S_j)=\frac{|\{w_k|w_k\in S_i\&w_k\in S_j\}|}{\log(|S_i|)+\log(|S_j|)}$$

5.2.3 特点

优点：

- 无监督方式，无需构造数据集训练。
- 算法原理简单且部署简单。
- 继承了 PageRank 的思想，效果相对较好，相对于 TF-IDF 方法，可以更充分的利用文本元素之间的关系。

缺点：

- 结果受分词、文本清洗影响较大，即对于某些停用词的保留与否，直接影响最终结果。
- 虽然与 TF-IDF 比，不止利用了词频，但是仍然受高频词的影响，因此，需要结合词性和词频进行筛选，以达到更好效果，但词性标注显然又是一个问题。

5.3 模型准备

5.3.1 seq2seq 模型

在自然语言生成的任务中，大部分是基于 seq2seq 模型实现的。[2]
seq2seq 模型是由 encoder，decoder 两部分组成的，其结构如图 5.3-1 所示：

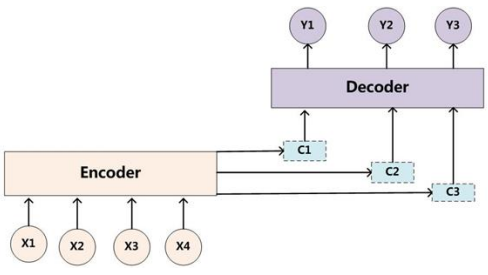

图 5.3-1 seq2seq 模型基本结构

原则上 encoder，decoder 可以由 CNN，RNN，Transformer 三种结构。这里主要介绍 Transformer 架构。（三种不同的特征提取）

CNN 卷积 conv 由于 NLP 问题与图像处理不同，故很少用

以前以 RNN 为主搭配 LSTM 神经网络进行。

5.3.2 RNN + LSTM

长期短期记忆网络（LSTMs）是一种特殊的 RNN，它可以解决梯度消失的问题，能轻松地学习到长期依赖的信息。[3]其基本结构如图 5.3-2 所示。

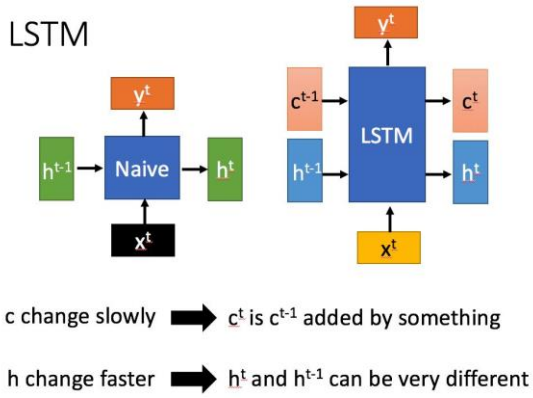

图 5.3-2 LSTM 基本结构

其中对于传递下去的 c^t 改变得很慢，通常输出的是 c^t 上一个状态传过来的 c^{t-1} 加上一些数值。

而 h^t 则在不同节点下往往会有很大的区别。

具体计算如图 5.3-3 所示：

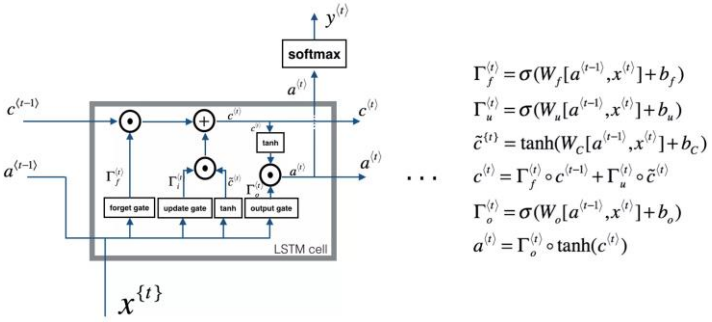

图 5.3-3 LSTM 计算流程

其实在 NLP 领域 RNN 基本上被抛弃了。

RNN(LSTM, GRU)训练时迭代，串行的，需等当前词处理完，再处理下一个词。Transformer 的训练（encoder，decoder）是并行的，所有词是同时训练 ，增加了计算效率。另一个就是 Transformer 句子是整体处理，而不是逐字处理。

5.3.3 Transformer 框架

Transformer 中抛弃了传统的 CNN 和 RNN，整个网络结构完全是由 Attention 机制组成。[4]更准确地讲，Transformer 由且仅由 self-Attenion 和 Feed Forward Neural Network 组成。其基本结构如图 5.3-4 所示。

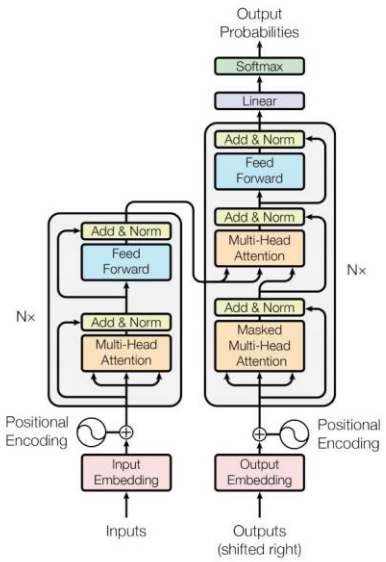

图 5.3-4 Transformer 基本结构

Transformer 模型总体的样子如图 5.3-5 所示：总体来说，还是和 Encoder-Decoder 模型有些相似，左边是 Encoder 部分，右边是 Decoder 部分。

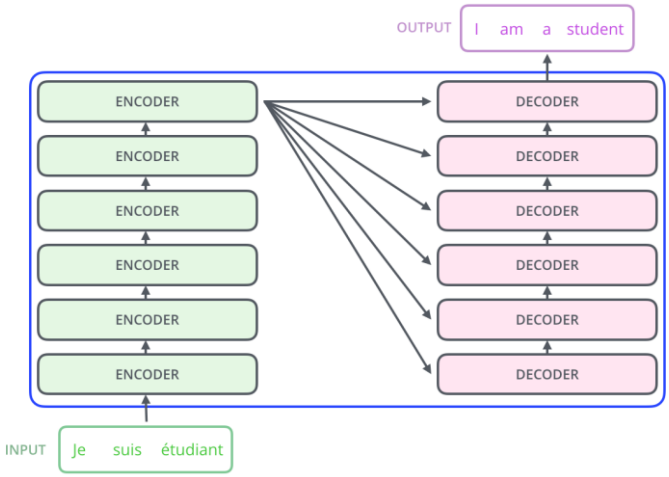

图 5.3-5 Transformer 模型总体结构

Encoder：输入是单词的 Embedding，再加上位置编码，然后进入一个统一的结构，这个结构可以循环很多次（N 次），也就是说有很多层（N 层）。每一层又可以分成 Attention 层和全连接层，再额外加了一些处理，比如 Skip Connection，做跳跃连接，然后还加了 Normalization 层。其实它本身的模型还是很简单的。

Decoder：第一次输入是前缀信息，之后的就是上一次产出的 Embedding，加入位置编码，然后进入一个可以重复很多次的模块。该模块可以分成三块来看，第一块也是 Attention 层，第二块是 cross Attention，不是 Self-Attention，第三块是全连接层。也用了跳跃连接和 Normalization。

输出：最后的输出要通过 Linear 层（全连接层），再通过 softmax 做预测。

5.3.4 RNN 和 Transformer 对比

RNN

- 顺序处理：句子必须逐字处理
- RNN 指的是一个序列当前的输出与之前的输出也有关，具体的表现形

式为网络会对前面的信息进行记忆，保存在网络的内部状态中，并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包含输入层的输出还包含上一时刻隐藏层的输出

- 它采取线性序列结构不断从前往后收集输入信息

Transformer

- 非顺序处理：句子是整体处理，而不是逐字处理
- 单个的 Transformer Block 主要由两部分组成：多头注意力机制(Multi-Head Attention)和前馈神经网络(Feed Forward)，Transformer Block 代替了 LSTM 和 CNN 结构作为了我们的特征提取器，使得 Transformer 不依赖于过去的隐藏状态来捕获对先前单词的依赖性，而是整体上处理一个句子，以便允许并行计算，减少训练时间，并减少由于长期依赖性而导致的性能下降

5.3.5 GPT

1. 核心思想

GPT 的核心思想是先通过无标签的文本去训练生成语言模型，再根据具体的 NLP 任务（如文本蕴涵、QA、文本分类等），来通过有标签的数据对模型进行 fine-tuning。[5][6][7]

具体来说，在这篇论文中提出了半监督的方法，即结合了无监督的预训练和有监督的 fine-tuning。论文采用两阶段训练。首先，在未标记数据集上训练语言模型来学习神经网络模型的初始参数。随后，使用相应 NLP 任务中的有标签的数据地将这些参数微调，来适应当前任务。

2. 模型结构

模型的结构是使用了多层的单向 Transformer 结构(《Attention is All you need》提出)。图 5.3-6 是 GPT 语言模型的结构：（这里采用编码器）

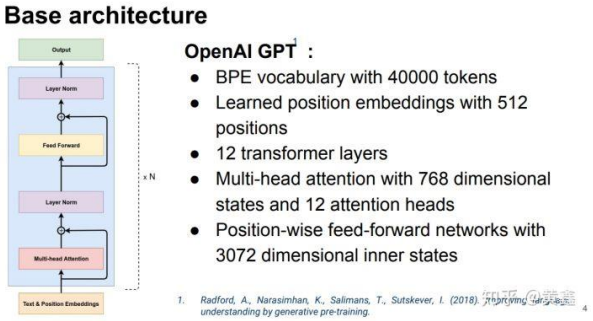

图 5.3-6 GPT 语言模型结构

训练的两个阶段如下：

1.无监督的预训练

对于无标签的文本$U=\{u_1,\cdots,u_n\}$，最大化语言模型的极大似然函数：

$$L_1(\mathcal{U})=\sum_i \log P(u_i \mid u_{i-k},...,u_{i-1};\Theta)$$

这里的 k 是文本上下文窗口的大小。

论文中使用的是多层 Transformer 的 decoder 的语言模型，input 为词嵌入以及单词 token 的位置信息；再对 transformer_block 的输出向量做 softmax，output

为词的概念分布。具体公式如下：

$$h_0 = UW_e + W_p$$

$$h_l = \text{operatorname}transformer\,\text{block}(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \text{operatorname}softmax(h_n W_e^T)$$

2.有监督的 fine-tuning（微调）

在对模型预训练之后，采用有监督的目标任务对模型参数微调。假设一个有标签的数据集，假设每一条数据为一个单词序列$x_1, \cdots x_m$以及相应的标签 y，通过之前预训练的模型获得输出向量h_l^m，再送入线性输出层，来预测标签 y

$$P(y \mid x^1, \ldots, x^m) = softmax(h_l^m W_y)$$

Loss 函数为：

$$L_2(C) = \sum_{(x,y)} \log P(y \mid x^1, \ldots, x^m)$$

最后，将两阶段的目标函数通过超参λ相加训练整个模型：

$$L_3(C) = L_2(C) + \lambda * L_1(C)$$

3. 具体任务的模型微调

图 5.3-7 给出了主要的模型微调方式。

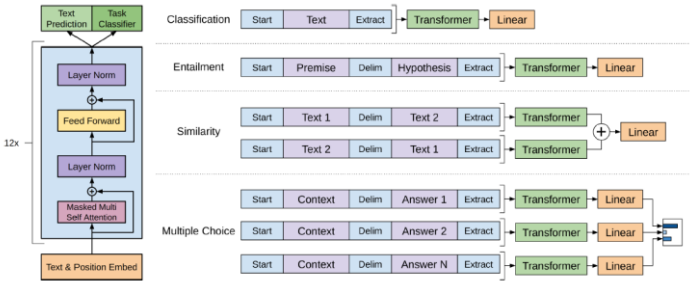

图 5.3-7 具体任务的模型微调

对于文本分类，只需要在预训练模型上微调。对于 QA 任务或者文本蕴含，因为预训练模型是在连续序列上训练，需要做一些调整，修改输入结构，将输入转化为有序序列输入。

5.4 Bert 模型

5.4.1 概述

BERT 的全称是 Bidirectional Encoder Representation from Transformers，即双向 Transformer 的 Encoder，因为 decoder 是不能获要预测的信息的。模型的主要创新点都在 pre-train 方法上，即用了 Masked LM 和 Next Sentence Prediction 两种方法分别捕捉词语和句子级别的 representation。[8]

5.4.2 模型结构

由于模型的构成元素 Transformer 已经解析过，就不多说了，BERT 模型的结构如图 5.4-1 最左：

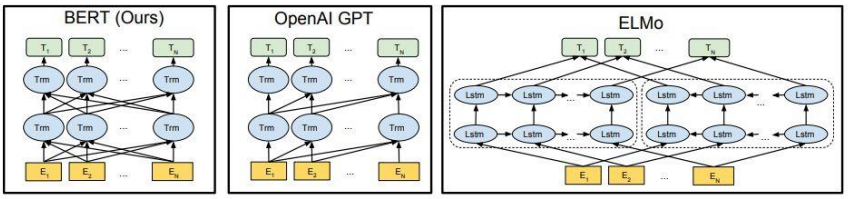

图 5.4-1 BERT 模型的结构

对比 OpenAI GPT(Generative pre-trained transformer)，BERT 是双向的 Transformer block 连接；就像单向 RNN 和双向 RNN 的区别，直觉上来讲效果会好一些。

对比 ELMo，虽然都是“双向”，但目标函数其实是不同的。ELMo 是分别以 $P(w_i|w_1,\cdots,w_{i-1})$ 和 $P(w_i|w_{i+1},\cdots,w_n)$ 作为目标函数，独立训练处两个 representation 然后拼接，而 BERT 则是以 $P(w_i|w_1,\cdots,w_{i-1},w_{i+1},\cdots,w_n)$ 作为目标函数训练 LM。

5.4.3 Embedding

这里的 Embedding 由三种 Embedding 求和而成，如图 5.4-2 所示：

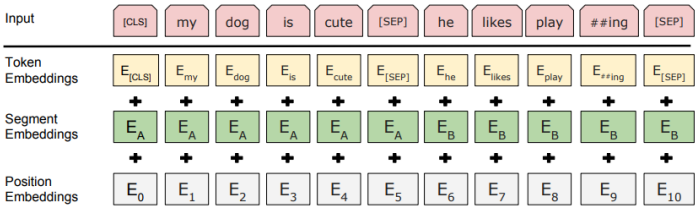

图 5.4-2 Embedding 结构

其中：

Token Embeddings 是词向量，第一个单词是 CLS 标志，可以用于之后的分类任务

Segment Embeddings 用来区别两种句子，因为预训练不光做 LM 还要做以两个句子为输入的分类任务

Position Embeddings 和之前文章中的 Transformer 不同，不是三角函数而是学习出来的

5.4.4 Pre-training 预训练

Task 1#: Masked LM

第一步预训练的目标就是做语言模型，从上文模型结构中看到了这个模型的不同，即 bidirectional。关于为什么要如此的 bidirectional，作者在 reddit 上做了解释，意思就是如果使用预训练模型处理其他任务，那人们想要的肯定不止某个词左边的信息，而是左右两边的信息。而考虑到这点的模型 ELMo 只是将 left-to-right 和 right-to-left 分别训练拼接起来。直觉上来讲我们其实想要一个 deeply bidirectional 的模型，但是普通的 LM 又无法做到，因为在训练时可能会“穿越”（关于这点我不是很认同，之后会发文章讲一下如何做 bidirectional LM）。所以作者用了一个加 mask 的 trick。[9]

在训练过程中作者随机 mask 15%的 token，而不是把像 cbow 一样把每个词都预测一遍。最终的损失函数只计算被 mask 掉那个 token。

Mask 如何做也是有技巧的，如果一直用标记[MASK]代替（在实际预测时是碰不到这个标记的）会影响模型，所以随机 mask 的时候 10%的单词会被替代成其他单词，10%的单词不替换，剩下 80%才被替换为[MASK]。具体为什么这么分配，作者没有说。要注意的是 Masked LM 预训练阶段模型是不知道真正被 mask 的是哪个词，所以模型每个词都要关注。

具体操作如图 5.4-3 所示。

数据生成器将执行以下操作，而不是始终用[MASK]替换所选单词：

80%的时间：用[MASK]标记替换单词，例如，my dog is hairy → my dog is [MASK]
10%的时间：用一个随机的单词替换该单词，例如，my dog is hairy → my dog is apple
10%的时间：保持单词不变，例如，my dog is hairy → my dog is hairy. 这样做的目的是将表示偏向于实际观察到的单词。

图 5.4-3 数据生成器的具体操作

因为序列长度太大（512）会影响训练速度，所以 90%的 steps 都用 seq_len=128 训练，余下的 10%步数训练 512 长度的输入。

Task 2#: Next Sentence Prediction

因为涉及到 QA 和 NLI 之类的任务，增加了第二个预训练任务，目的是让模型理解两个句子之间的联系。训练的输入是句子 A 和 B，B 有一半的几率是 A 的下一句，输入这两个句子，模型预测 B 是不是 A 的下一句。预训练的时候可以达到 97-98%的准确度。

注意：作者特意说了语料的选取很关键，要选用 document-level 的而不是 sentence-level 的，这样可以具备抽象连续长序列特征的能力。

5.4.5 Fine-tunning 微调

对具体任务需要进行一些调整，如图 5.4-4 所示。

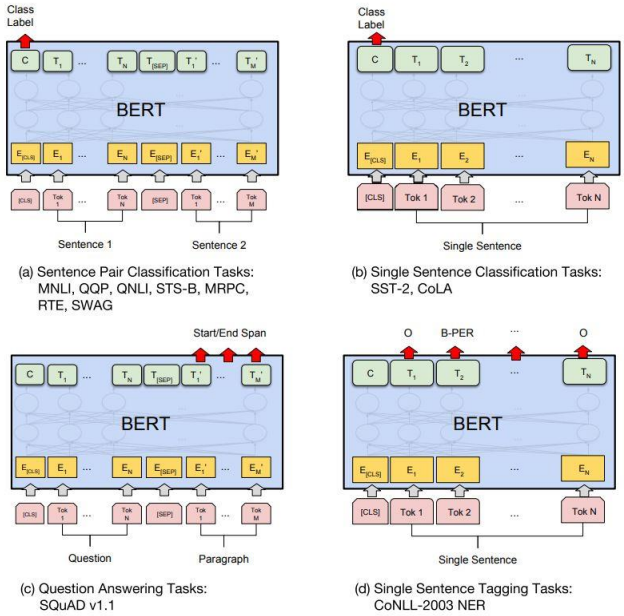

图 5.4-4 Bert 微调

5.5 Bert+UNILM 模型

5.5.1 概述

UNILM 全名 Unified Language Model Pre-training for Natural Language Understanding and Generation，其实也是提出了一种预训练方法，而且很简洁，直接复用 BERT 的结构和参数就好。NLU 直接用 BERT 做，NLG 直接把 BERT 的 S1 [SEP] S2 当成 encoder-decoder，虽然没有那个结构，但是心中有那个思想。

5.5.2 模型结构

模型结构如图 5.5-1 所示。

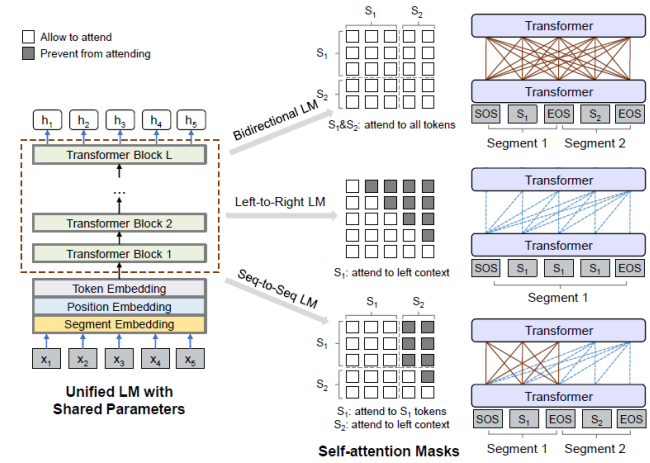

Figure 1: Overview of unified LM pre-training. The model parameters are shared across the LM objectives (i.e., bidirectional LM, unidirectional LM, and sequence-to-sequence LM). We use different self-attention masks to control the access to context for each word token. The right-to-left LM is similar to the left-to-right one, which is omitted in the figure for brevity.

图 5.5-1 UNILM 结构

作者很巧妙的抓住了 MASK 这个点，认为不管是什么 LM，本质都是在训练时能获取到什么信息，在实现层面其实就是 mask 什么输入的问题。所以完全可以把 Seq2Seq LM 整合到 BERT 里，在 S1 [SEP] S2 [SEP] 中，S1 用 encoder 编码，S2 中的 token 只能获取 S1 和自己之前的 token 信息，如上图最下面那个 mask 矩阵（其实图比我说的清楚）。

在预训练阶段，UniLM 模型通过三种不同目标函数的语言模型（包括：双向语言模型，单向语言模型和序列到序列语言模型），去共同优化同一个 Transformer 网络；为了控制对将要预测的 token 可见到的上下文，作者使用了不同的 self-attention mask 来实现。换句话说，就是通过不同的掩码来控制预测单词的可见上下文词语数量，实现不同的模型表征。

5.5.3 Pre-training

1. Input representation：这里和 BERT 一样使用了三个 Embedding，但是参考 WordPiece 把 token 都处理成了 subword，增强了生成模型的表现能力。另外，作者强调了 segment embedding 可以帮助区分不同的 LM。
2. Transformer：没有变化，只是强调了会通过不同的 Mask 矩阵控制 LM 任务。
3. 单向 LM：只输入单句。其他单向 LM 是计算每一个 token 预测的损失，

而作者依旧采用 Masked LM 的思路，只计算 Mask 的 token 损失。个人认为是由于 BERT 为了实现 BiLM，在预训练时都是做完形填空（给[MASK]预测当前应有的编码），显然不适合给 x1 预测 x2 的情形。因此为了统一单向 LM 和 BiLM 的结构，采用了 Masked left-to-right LM 这种方式。

4. Seq2Seq LM：输入两句。第一句采用 BiLM 的编码方式，第二句采用单向 LM 的方式。同时训练 encoder(BiLM)和 decoder(Uni-LM)。处理输入时同样也是随机 mask 掉一些 token。
5. 同样包含了 Next sentence 任务。
6. 训练时，在一个 batch 里，优化目标的分配是 1/3 的时间采用 BiLM 和 Next sentence，1/3 的时间采用 Seq2Seq LM，1/6 的时间分别给从左到右和从右到左的 LM。这里我没懂具体如何实现的，是一个 batch 里放不同的数据？懂得童鞋们说一下～
7. 参数是从 BERT-large 初始化的。
8. 加 Mask 的频率和 BERT 一样，但是对于加 Mask，80%的时间随机 mask 一个，20%时间会 mask 一个 bigram 或 trigram，增加模型的预测能力。[10]

5.5.4 Fine-tunning

NLU：参考 BERT

NLG：在精调期间只 mask S2 句子中的 token，而且 S2 的[SEP]也会被随机 mask，让模型学习如何停止。

生成式摘要：增加了抽取式摘要作为辅助任务，根据 first token 预测 input sentence 是否出现在抽取式数据中。

5.6 模型训练

5.6.1 服务器

使用“九天-毕昇一站式人工智能学习和实战平台”。

中国移动和北京邮电大学联合研发，为 AI 学习者提供充沛的 GPU 算力、丰富的数据和学习实战资源，服务课程学习、比赛打榜、工作求职等全流程场景，并面向高校提供在线教学、科研开发的一站式解决方案。界面如图 5.6-1 所示。

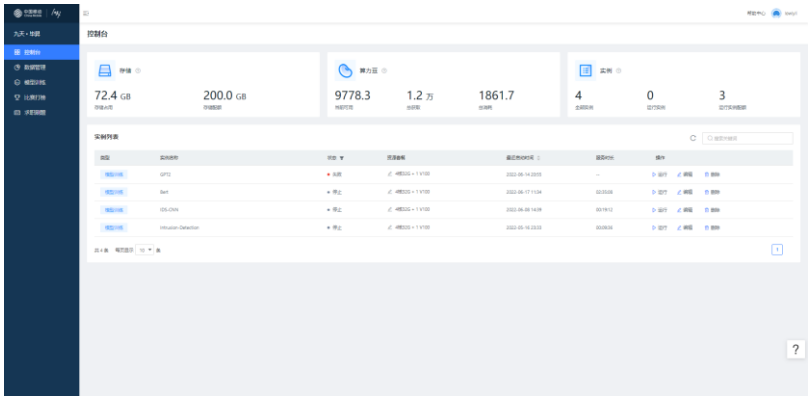

图 5.6-1 九天-毕昇一站式人工智能学习和实战平台

这里选取 V100 的显卡服务器进行微调训练，训练界面如图 5.6-2 所示。

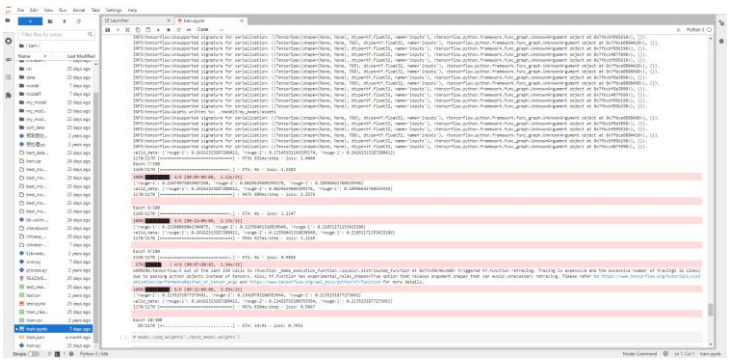

图 5.6-2 微调训练界面

5.6.2 代码分析

这里利用 bert4keras 库进行实现。bert4keras 是一个基于 keras 的预训练模型加载框架，目前支持多种预训练模型（BERT、ALBERT、RoBERTa、ALBERT、NEZHA、GPT2、T5 等），并支持多种环境（python 2.7、python 3.x）和后端（keras、tf.keras、tf 1.x、tf 2.x）。

bert4keras 秉承 keras 的人性化理念，在框架上充分借鉴了 keras 的设计，尽量做到优雅、简练而不简单。预训练采用哈工大开源预训练模型——中文 BERT-wwm，主页如下。

图 5.6-3 中文 BERT-wwm 介绍界面

微调重新设计编码解码，与任务匹配，实现标题的生成（主流都是摘要，标题准确率太低），其代码如图 5.6-4 所示。

```
class AutoTitle(AutoRegressiveDecoder):
    """seq2seq解码器
    """
    @AutoRegressiveDecoder.wraps(default_rtype='probas')
    def predict(self, inputs, output_ids, states):
        token_ids, segment_ids = inputs
        token_ids = np.concatenate([token_ids, output_ids], 1)
        segment_ids = np.concatenate([segment_ids, np.ones_like(output_ids)], 1)
        return model.predict([token_ids, segment_ids])[:, -1]

    def generate(self, text, topk=1):
        max_c_len = maxlen - self.maxlen
        token_ids, segment_ids = tokenizer.encode(text, maxlen=max_c_len)
        output_ids = self.beam_search([token_ids, segment_ids],
                                      topk)  # 基于beam search
        return tokenizer.decode(output_ids)
```

```
class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids = [], []
        for is_end, (title, content) in self.sample(random):
            token_ids, segment_ids = tokenizer.encode(
                content, title, maxlen=maxlen
            )
            batch_token_ids.append(token_ids)
            batch_segment_ids.append(segment_ids)
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                yield [batch_token_ids, batch_segment_ids], None
                batch_token_ids, batch_segment_ids = [], []
```

图 5.6-4 实现标题生成

改进交叉熵，利用稀疏 Softmax 构建 Sparse Softmax。Sparse Softmax 的思想源于《From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label

Classification》、《Sparse Sequence-to-Sequence Models》等文章，大佬的改进这里借用。代码如图 5.6-5 所示。

```python
class CrossEntropy(Loss):
    """交叉熵作为Loss，并mask掉输入部分
    """
    def compute_loss(self, inputs, mask=None):
        y_true, y_mask, y_pred = inputs
        y_true = y_true[:, 1:]  # 目标token_ids
        y_mask = y_mask[:, 1:]  # segment_ids，刚好指示了要预测的部分
        y_pred = y_pred[:, :-1]  # 预测序列，错开一位
        loss = K.sparse_categorical_crossentropy(y_true, y_pred)
        loss = K.sum(loss * y_mask) / K.sum(y_mask)
        return loss
```

图 5.6-5 稀疏 Softmax 构建 Sparse Softmax

最终模型架构和结果：累计训练接近 1 亿个参数（NLP 模型普遍偏大），获得多个训练模型，如图 5.6-6 所示。

Launcher | train.ipynb | Code

```
Transformer-10-MultiHeadSelfAtt (None, None, 768)    0          Transformer-9-FeedForward-Norm[0]
                                                                Transformer-10-MultiHeadSelfAtten
Transformer-10-MultiHeadSelfAtt (None, None, 768)    1536       Transformer-10-MultiHeadSelfAtten
Transformer-10-FeedForward (Fee (None, None, 768)    4722432    Transformer-10-MultiHeadSelfAtten
Transformer-10-FeedForward-Drop (None, None, 768)    0          Transformer-10-FeedForward[0][0]
Transformer-10-FeedForward-Add  (None, None, 768)    0          Transformer-10-MultiHeadSelfAtten
                                                                Transformer-10-FeedForward-Dropou
Transformer-10-FeedForward-Norm (None, None, 768)    1536       Transformer-10-FeedForward-Add[0]
Transformer-11-MultiHeadSelfAtt (None, None, 768)    2362368    Transformer-10-FeedForward-Norm[0
                                                                Transformer-10-FeedForward-Norm[0
                                                                Transformer-10-FeedForward-Norm[0
                                                                Attention-UniLM-Mask[0][0]
Transformer-11-MultiHeadSelfAtt (None, None, 768)    0          Transformer-11-MultiHeadSelfAtten
Transformer-11-MultiHeadSelfAtt (None, None, 768)    0          Transformer-10-FeedForward-Norm[0
                                                                Transformer-11-MultiHeadSelfAtten
Transformer-11-MultiHeadSelfAtt (None, None, 768)    1536       Transformer-11-MultiHeadSelfAtten
Transformer-11-FeedForward (Fee (None, None, 768)    4722432    Transformer-11-MultiHeadSelfAtten
Transformer-11-FeedForward-Drop (None, None, 768)    0          Transformer-11-FeedForward[0][0]
Transformer-11-FeedForward-Add  (None, None, 768)    0          Transformer-11-MultiHeadSelfAtten
                                                                Transformer-11-FeedForward-Dropou
Transformer-11-FeedForward-Norm (None, None, 768)    1536       Transformer-11-FeedForward-Add[0]
MLM-Dense (Dense)               (None, None, 768)    590592     Transformer-11-FeedForward-Norm[0
MLM-Norm (LayerNormalization)   (None, None, 768)    1536       MLM-Dense[0][0]
MLM-Bias (ScaleOffset)          (None, None, 14068)  14068      Embedding-Token[1][0]
MLM-Activation (Activation)     (None, None, 14068)  0          MLM-Bias[0][0]
cross_entropy (CrossEntropy)    (None, None, 14068)  0          Input-Token[0][0]
                                                                Input-Segment[0][0]
                                                                MLM-Activation[0][0]
==================================================================================================
Total params: 96,861,172
Trainable params: 96,861,172
Non-trainable params: 0
```

图 5.6-6 输出模型结构及其参数数量

最终参数为：`maxlen = 512, batch_size = 5, epochs = 100.`

5.7 模型评估

5.7.1 ROUGE 评估

ROUGE 用作机器翻译评价指标的初衷是这样的：在 SMT（统计机器翻译）时代，机器翻译效果稀烂，需要同时评价翻译的准确度和流畅度；等到 NMT（神

经网络机器翻译）出来以后，神经网络脑补能力极强，翻译出的结果都是通顺的，但是有时候容易瞎翻译。

ROUGE 的出现很大程度上是为了解决 NMT 的漏翻问题（低召回率）。所以 ROUGE 只适合评价 NMT，而不适用于 SMT，因为它不管候选译文流不流畅。

ROUGE-N

“N”指的是 N-gram，其计算方式与 BLEU 类似，只是 BLEU 基于精确率，而 ROUGE 基于召回率。

ROUGE-N 主要统计 N-gram 上的召回率，对于 N-gram，可以计算得到 ROUGE-N 分数，计算公式如下：

$$ROUGE-N=\frac{\sum_{S\in ReferenceSummaries}\sum_{gram_n\in S}Count_{math}(\text{gram}_n)}{\sum_{S\in \text{ReferenceSummaries}}\sum_{g^n_{\text{gram}}\in S}Count(\text{gram}_n)}$$

公式的分母是统计在参考译文中 N-gram 的个数，而分子是统计参考译文与机器译文共有的 N-gram 个数。

5.7.2 BLEU 评估

所谓 BLEU，最开始是用于机器翻译中。他的思想其实很 native，对于一个给定的句子，有标准译文 S1，还有一个神经网络翻译的句子 S2。BLEU 的思想就是对于出现机器翻译 S2 的所有短语，看有多少个短语出现在 S1 中，然后算一下这个比率就是 BLEU 的分数了。首先根据 n-gram 划分一个短语包含单词的数量，有 BLEU-1,BLEU-2,BLEU-3,BLEU-4。分别就是把文章划分成长度为 1 个单词的短语，长度为 2 个单词的短语，然后统计她们出现在标准译文中个数，在分别除以划分总数，就是对应的 BLEU-1 分数，BLEU-2 分数。其实就是准确率。看这些划分中有多少是出现在标准译文当中的。一般而言：unigram 的准确率可以用于衡量单词翻译的准确性，更高阶的 n-gram 的准确率可以用来衡量句子的流畅性，n{1,2,3,4}

但是 BLEU 会有个缺陷，假如我就翻译一个单词，而这个单词正好在标准译文中，那岂不是准确率 100%，对于这个缺陷，BLEU 算法会有个长度惩罚因子，就是翻译太短了就会有惩罚，不过，总的来说，还是偏向于短翻译分数高一点。

BLEU 指标先计算生成翻译与相应的参考翻译的 n-gram 精确率，具体的公式为：

$$p_n=\frac{\sum_{C\in Candidate}\sum_{gram_n\in C}Count_{clip}(\text{gram}_n)}{\sum_{C`\in \text{Candidate}}\sum_{g^n_{\text{gram}}\in C`}Count(\text{gram}_n)}$$

其中分母部分计算生成翻译中 n-gram 的个数，分子部分计算参考摘要和自动摘要共有的 n-gram 的个数。

然后再计算具体的 BLEU 值：

$$BLEU=BP\cdot\exp\left(\sum_{n=1}^{N}w_n\log p_n\right)$$

5.7.3 验证结果

设计评估指标，并利用测试集，进行检验，寻找较优模型，代码如图 5.7-1 所示。

```python
def evaluate(self, data, topk=1):
    total = 0
    res = []
    rouge_1, rouge_2, rouge_l, bleu = 0, 0, 0, 0
    for title, content in tqdm(data):
        total += 1
        title = ' '.join(title).lower()
        pred_title = ' '.join(autotitle.generate(content, topk)).lower()
        res.append(pred_title)
        if pred_title.strip():
            scores = self.rouge.get_scores(hyps=pred_title, refs=title)
            rouge_1 += scores[0]['rouge-1']['f']
            rouge_2 += scores[0]['rouge-2']['f']
            rouge_l += scores[0]['rouge-l']['f']
            bleu += sentence_bleu(
                references=[title.split(' ')],
                hypothesis=pred_title.split(' '),
                smoothing_function=self.smooth
            )
    pd.DataFrame(res).to_csv('res3.csv', index=None)
    rouge_1 /= total
    rouge_2 /= total
    rouge_l /= total
    bleu /= total
    return {
        'rouge-1': rouge_1,
        'rouge-2': rouge_2,
        'rouge-l': rouge_l,
        'bleu': bleu,
    }
```

图 5.7-1 评估代码设计

最终目前的评估得分输出如图 5.7-2 所示，即：Rouge-1：0.3，Rouge-2：0.19，Rouge-l：0.28，Bleu：0.13。

```
metrics = evaluator.evaluate(test_data)  # 评测模型
print('valid_data:', metrics)

✓ 201m 17.8s

100%|██████████| 1679/1679 [3:21:17<00:00,  7.19s/it]

valid_data: {'rouge-1': 0.3041298785856503, 'rouge-2': 0.19703214401602984, 'rouge-l': 0.2872118283372123, 'bleu': 0.13044759471052714}
```

图 5.7-2 评估得分输出结果

5.7.4 效果展示

原标题与生成标题对比展示如图 5.7-3 所示。

```
df = pd.read_csv('./result1.csv')
df
```

	title	predict_title
0	中国持续发力削减全球"赤字"	望海楼破解四大赤字的中国担当
1	小康路上一个也不能少	团结奋斗是脱贫攻坚最有力武器
2	优质内容是不变的刚需	把握全媒体时代的大趋势
3	进入"重大灾难期"的美国，蓬佩奥们为何趁火打劫？	企图胁迫全球抗疫的蓬佩奥正在向全球叫板
4	辛识平别让"督查泛滥"拖累基层战"疫"	让督查检查不能过滥、过滥
...	...	...
1674	携手共建人类卫生健康共同体	为全球抗疫注入人类命运共同体新动力
1675	党的领导是推进全面依法治国的根本保证	坚持党对全面依法治国的领导
1676	激活乡村振兴的内生动力	激活乡村振兴的内生动力
1677	唐续精神血脉	发扬精神葆本色
1678	为世界经济和全球治理把准航向	为世界经济和全球治理把准航向

1679 rows × 2 columns

	A	B
178	"一国两制"具有深远历史意义	望海楼—国两制是澳门特色制度
179	以知识产权全链条保护促进高质量发展	让知识产权更好保护
180	让科研指挥棒更科学	让三评激发科技创新活力
181	涵养深沉的家国情怀	涵养家国情怀的深沉情怀
182	激发共产党人奋勇前进的根本动力	以奋勇前进的根本动力
183	今天，我们和周果一样心怀期待	巴黎圣母院，让我们热泪盈眶
184	全面小康大家一起走	社会主义道路一个也不能少
185	仲祖文切实加强对换届工作的领导和指导	确保四个有效
186	中国有足够信心底气战胜任何困难挑战	和音发出中国答卷的中国答卷
187	必须高度警惕麻痹思想厌战情绪骄傲心理松劲心态	牙关铁健筑牢疫情防控的钢墙铁壁
188	用新发展理念赋局成势	以短板思维拓展发展新路
189	始终把人民至上生命至上放在首位	为维护人民健康筑牢坚实堤坝
190	谱写人民警察光荣历史新篇章	对党忠诚、服务人民执法公正的支援是对人民警察最好的支援
191	筑牢民族复兴的精神支撑	强化理想信念和文化自信
192	聆听改革开放的健康足音	深刻变革开放的原洪力量
193	为敢担当的干部担当	让敢闯敢干的人成为更多看门人
194	青年党员成长"加蓝"与"加压"需并重	在练好三本功中成长
195	光明日报社论同心同德拼搏进取决胜全面建成小康社会	共襄复兴伟业
196	亚洲的博鳌世界的博鳌	开放的中国方案，越来越多
197	深刻领会"不忘初心，牢记使命"主题教育的总要求	认真贯彻落实主题教育
198	瞩望上海合作组织青岛之约	坚守初心携手前进
199	集中力量办大事	凝聚制度优势，办大事更有光芒
200	武汉离机"带货"，拓宽外贸出口通道	着改货助力航空货运高质量
201	人民军队为人民	向新时代最可爱的人民致敬
202	做钟扬那样的播种者	后人应该继续给时代的进步送去火种
203	稳定人心破困局提振信心再出发	望海楼不能让香港经济成为暴力的牺牲品
204	历经风雨砥砺，中国永远在这儿	云理轩般忧启圣多难兴邦

result1

图 5.7-3 原标题与生成标题对比展示

5.7.5 模型的不足与改进

- 运行时间太慢，长文本先做降维处理，导致过程繁琐，后续模型 bert 虽然轻量化可时间复杂度高，以本电脑轻薄本为例子，无 GPU 状态供电，预测一个需要 8-20s，而在不供电（轻薄本自动降频），需要 1m20s（难以忍受）。
- 模型目前 rouge-1 为 0.3，虽然效果良好，但未达到最优，仍然有优化空间，扩大算力和训练次数，增加回调函数可以使模型更优
- 可以引入 Copy 等机制提高模型的 rouge 得分。
- 斯坦福大学 CS 博士新作：新型 Attention 提速 2-4 倍，BERT 单节点训练最快（2022.5.27 发布）https://arxiv.org/abs/2205.14135
- 一种快速、内存高效的注意力算法，被命名为 FlashAttention，如图 5.7-4 所示。通过减少 GPU 内存读取 / 写入，FlashAttention 的运行速度比 PyTorch 标准注意力快 2-4 倍，所需内存减少 5-20 倍。

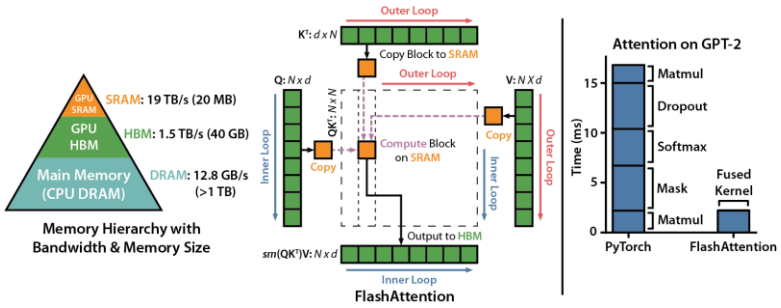

图 5.7-4 FlashAttention 主要结构

5.8 本章小结

首先介绍了算法的整体流程，进行了数据分析与样本预览，提出了“**抽取+生成**”的建模思路，接着描述了抽取式算法 TextRank 的流程与特点。模型部分，从 seq2seq 开始介绍，解释了 RNN+LSTM 的模型思想，接着介绍了目前应用广泛的 tansformer 模型,并将 RNN 与 Transformer 进行了比较，以及基于 Transform 的 GPT 模型。之后提出了项目应用的模型：BERT+UniLM 模型。应用 ROUGE 以及 BLEU 评价指标对模型的效果进行评价。最后分析了模型的不足之处以及日后的改进方法。

六、智能创作平台的编码与实现

整体代码在编写过程中，同样遵循“高耦合低内聚”的开发原则，分模块进行开发设计。

6.1 用户管理

6.1.1 用户注册

用户注册前端页面为 *register.html*，前端页面主要实现了<form action="/register/" method="post">表单提交、JS 随机验证码、JS 用户名、密码、邮箱正则合法性验证、msg 显示前端提示的错误或者后端服务器返回的错误消息。主要代码如代码 6.1-1 所示。

代码 6.1-1 用户注册前端主要代码

user/templates/user/register.html

```
<script type="text/javascript">
        $(function () {
            if ($("div.errorMsg").text() == "") {
                $("div.errorMsg").hide();
            }
            var
chars=['A','B','C','D','a','b','c','d','0','1','2','3'];
            var randCode="";
            for(var i=0;i<4;i++){
                var randPosition =
Math.floor(Math.random()*(chars.length-1));  //每次生成一个随机数的位置
                randCode += chars[randPosition];//带有随机位置作为下标
            }
            $(".given-code").text(randCode);
            console.log(randCode);

            // 给注册绑定单击事件
            $("#btn_submit").click(function () {
                // 验证用户名：必须由字母，数字下划线组成，并且长度为5 到12 位
                var usernameText = $("#username").val();
                var usernamePatt = /^\w{5,12}$/;
                if (!usernamePatt.test(usernameText)) {
                    $("div.errorMsg").show().text("用户名非法：用户名应长
度 5 到 12 位，仅含字母，数字下划线");
                    return false;
                }

                // 验证密码：必须由字母，数字下划线组成，并且长度为5 到12 位
```

user/templates/user/register.html

```
                var passwordText = $("#password").val();
                var passwordPatt = /^\w{5,12}$/;
                if (!passwordPatt.test(passwordText)) {
                    $("div.errorMsg").show().text("密码不合法：密码应长度 5
到 12 位，仅含字母，数字下划线");

                    return false;
                }

                // 验证确认密码：和密码相同
                var repasswordText = $("#repassword").val();
                if (repasswordText != passwordText) {
                    $("div.errorMsg").show().text("确认密码和密码不一致！
");
                    return false;
                }

                // 邮箱验证：xxxxx@xxx.com
                var emailText = $("#email").val();
                var emailPatt = /^[a-z\d]+(\.[a-z\d]+)*@([\da-z](-
[\da-z])?)+(\.{1,2}[a-z]+)+$/;
                if (!emailPatt.test(emailText)) {
                    $("div.errorMsg").show().text("邮箱格式不合法：邮箱格
式应为 xxxxx@xxx.com");
                    return false;
                }

                var codeText = $("#code").val();
                if (codeText == null || codeText == "") {
                    $("div.errorMsg").show().text("验证码不能为空！ ");
                    return false;
                }

                if (codeText !== $(".given-code").text()) {
                    $("div.errorMsg").show().text("验证码不正确！ ");
                    return false;
                }

                // 去掉错误信息
                $("div.errorMsg").text("").hide();
            });

        });
```

user/templates/user/register.html

```
    </script>
</head>
<body>

<div class="main">
    <div class="main-header">
        <div>
            <img src="{% static 'media/web/icon.jpg' %}">
            <span>欢迎注册写作小助手</span>
        </div>
    </div>
    {% if msg %}
        <div class="errorMsg">{{ msg }}</div>
    {% else %}
        <div class="errorMsg"></div>
    {% endif %}
    <form action="/register/" method="post">
        {%csrf_token%}
        <div class="input-box">
            <span class="icon" id="id"></span>
            <input class="login" type="text" placeholder="请输入用户名"
name="username" id="username" value="{{ username }}" />
        </div>
        <div class="input-box">
            <span class="icon" id="pw"></span>
            <input class="login" type="password" placeholder="请输入密
码" name="password" id="password" value="{{ password }}" />
        </div>
        <div class="input-box">
            <span class="icon" id = "repw"></span>
            <input class="regist" type="password" placeholder="确认密
码" name="repassword" id="repassword" value="{{ password }}"/>
        </div>
        <div class="input-box">
            <span class="icon" id="mail"></span>
            <input class="regist" type="text" placeholder="请输入邮箱地
址" name="email" id="email" value="{{ email }}"/><br/>
        </div>
        <div class="input-box" >
            <span class="icon" id="icode"></span>
            <input class="regist" type="text" placeholder="请输入验证码
"  name="code" id="code"/>
            <span class="given-code">1234</span>
```

user/templates/user/register.html

```
        </div>
        <div class="button-box">
            <input type="submit" value="注册" id="btn_submit"/>
        </div>
    </form>
    <div class="tips">
        <span>已有帐户？<a href="/login/">立即登录</a></span>
    </div>
</div>
</body>
</html>
```

用户注册后端程序要通过 *urls.py* 找到 *views.py* 文件中的 *registerView* 函数，如果是通过 post 请求进入，也就是注册界面表单进入的，则进行注册功能的实现，只有用户名不存在的时候才能注册成功，并且跳转到登陆界面，否则要将相应的错误信息返回到注册界面。该部分的函数代码如代码 6.1-2 所示。

代码 6.1-2 用户注册后端主要代码

user/views.py > registerView(request)

```
def registerView(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")
        email = request.POST.get("email")
        if User.objects.filter(username=username):
            msg = "用户名已存在"
        else:
            user = User.objects.create_user(username=username,
password=password, email=email)
            msg = "注册成功"
            return redirect("/login/")
    return render(request, "user/register.html", locals())
```

注册界面展示如图 6.1-1 所示。

图 6.1-1 注册界面

注册合法性测试如图 6.1-2 所示，经测试验证，注册功能正常。

图 6.1-2 注册合法性测试

6.1.2 用户登录

用户登录前端页面为 *login.html*，与注册页面基本一致，前端页面主要实现了<form action="/login/" method="post">表单提交、JS 随机验证码、JS 用户名、密码、正则合法性验证、msg 显示前端提示的错误或者后端服务器返回的错误消息。主要代码如代码 6.1-3 所示。

代码 6.1-3 用户登录前端主要代码

user/templates/user/login.html

```
…
    <script type="text/javascript">
        // 页面加载完成之后
        $(function () {
            if ($("div.errorMsg").text() === "") {
                $("div.errorMsg").hide();
            }
            // 给注册绑定单击事件
            $("#btn_submit").click(function () {

                var usernameText = $("#username").val().trim();
                if (usernameText === '') {
                    $("div.errorMsg").show().text("用户名为空！");
                    return false;
                }

                var passwordText = $("#password").val().trim();
                if (passwordText === '') {
                    $("div.errorMsg").show().text("密码为空！");
                    return false;
                }

                // 去掉错误信息
```

user/templates/user/login.html

```
                $("div.errorMsg").text("").hide();
            });

        });
    </script>
</head>
<body>

<div class="main">
    <div class="main-header">
        <div>
            <img src="{% static 'media/web/icon.jpg' %}">
            <span>欢迎登录写作小助手</span>
        </div>
    </div>
    {% if msg %}
        <div class="errorMsg">{{ msg }}</div>
    {% else %}
        <div class="errorMsg"></div>
    {% endif %}
    <form action="/login/" method="post" class="login">
        {% csrf_token %}
        <div class="input-box">
            <span class="icon" id="id"></span>
            <input class="login" type="text" placeholder="请输入用户名"
name="username" id="username" value="{{ username }}"/>
        </div>
        <div class="input-box">
            <span class="icon" id="pw"></span>
            <input class="login" type="password" placeholder="请输入密
码" name="password" id="password"/><br/>
        </div>
        <div class="button-box">
            <input type="submit" value="登录" id="btn_submit"/>
        </div>
    </form>
    <div class="tips">
        <span>还未注册？<a href="/register/">立即注册</a></span>
    </div>
</div>
</body>
</html>
```

用户登录后端程序要通过 *urls.py* 找到 *views.py* 文件中的 *loginView* 函数，如果是通过 post 请求进入，也就是登录界面表单进入的，则进行登录功能的实现，只有用户名不存在的时候才能登录成功，设置一天的 session，并且跳转到用户个人主界面，否则要将相应的错误信息（用户不存在、密码不正确）返回到登录界面。同时还考虑到，如果某用户 session 仍存在，也就是没有注销的时候不能再次登录，应该直接进入用户主界面，需要进行判断。该部分的函数代码如代码 6.1-4 所示。

代码 6.1-4 用户登陆后端主要代码

user/views.py > loginView(request)

```
def loginView(request):
    if request.user.is_authenticated:
        return redirect("/writing")
    else:
        if request.method == "POST":
            username = request.POST.get("username")
            password = request.POST.get("password")
            if User.objects.filter(username=username):
                user = authenticate(username=username,
password=password)
                if user:
                    login(request, user)
                    request.session['username'] = username
                    msg = "登录成功"
                    request.session.set_expiry(86400)
                    return redirect("/writing")
                else:
                    msg = "用户名密码错误，或被限制登录"
            else:
                msg = "用户名不存在"
    return render(request, "user/login.html", locals())
```

登录界面展示如图 6.1-3 所示。

图 6.1-3 登录界面

登录合法性测试如图 6.1-4 所示，经测试验证，登录功能正常。

图 6.1-4 登录合法性测试

6.1.3 修改密码

用户修改密码前端页面为 *user.html*，前端页面主要实现了下拉菜单显示修改用户信息跳转链接、表单提交、正则合法性验证、`layer.msg` 显示前端提示的错误或者后端服务器返回的错误消息，使用 JS 和 ajax 进行数据传输。主要代码如代码 6.1-5 所示。

代码 6.1-5 修改密码界面主要代码

writing /templates/writing/user.html

```
<div class="layui-header">
    <ul class="layui-nav">…</ul>
    <ul class="layui-nav layui-layout-right">
        <li class="layui-nav-item layui-hide layui-show-md-inline-
block">
            <a href="javascript:;">…</a>
            <dl class="layui-nav-child">
                <dd><a href="{% url 'writing:user' %}">修改信息
</a></dd>
                <dd><a href="{% url 'user:logout' %}">退出登录</a></dd>
            </dl>
        </li>
    </ul>
</div>
…
<form class="layui-form" action="" lay-filter="user_edit">
    {% csrf_token %}
    <div class="layui-form-item">
        <div class="layui-inline">
            <label class="layui-form-label">输入旧密码</label>
            <div class="layui-input-inline">
                <input type="password" name="old_password"
autocomplete="off" class="layui-input">
            </div>
        </div>
    </div>
    <div class="layui-form-item">
        <div class="layui-inline">
```

writing /templates/writing/user.html

```
            <label class="layui-form-label">输入新密码</label>
            <div class="layui-input-inline">
                <input type="password" name="new_password" lay-
verify="new_password" autocomplete="off"
                       class="layui-input">
            </div>
        </div>
    </div>
    <div class="layui-form-item">
        <div class="layui-inline">
            <label class="layui-form-label">再次输入</label>
            <div class="layui-input-inline">
                <input type="password" name="repeat_password"
autocomplete="off" class="layui-input">
            </div>
        </div>
    </div>
    <div class="layui-form-item">
        <div class="layui-input-block">
            <button type="submit" class="layui-btn" lay-submit=""
lay-filter="btn_edit">立即提交</button>
        </div>
    </div>
</form>
…

<script>
    layui.use(['form', 'layedit'], function () {
        var form = layui.form;

        //自定义验证规则
        form.verify({
            new_password: function (value) {
                var passwordPatt = /^\w{5,12}$/;
                if (!passwordPatt.test(value)) {
                    return "密码不合法：密码应长度 5 到 12 位，仅含字母，数字下
划线";
                }
            },
        });

        //提交事件
        form.on('submit(btn_edit)', function (data) {
```

writing /templates/writing/user.html

```
            $.ajax({
                url: "{% url 'writing:user_update' %}",
                type: 'post',
                data: data.field,
                dataType: "text",
                success: function (result) {
                    // console.log(data)
                    console.log("result: " + result)
                    if (result === 'success') {
                        layer.msg('提交成功', {icon: 1});
                        location.reload();
                    } else {
                        layer.msg(result, {icon: 2});
                    }
                }
            });
            return false;//阻止表单跳转
        });
    });
</script>
```

修改密码后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *user_update* 函数，非法的时候将相应的错误信息（新密码为空、两次密码不一致、新旧密码相同）返回到登录界面。正确时返回 success，前端接收到后会刷新页面，回到登陆界面，需要重新登陆，该部分的函数代码如代码 6.1-6 所示。

代码 6.1-6 修改密码后端代码

writing/views.py > user_update(request)

```
@login_required(login_url='/login/')
def user_update(request):
    msg = ''
    old_password = request.POST.get('old_password')
    new_password = request.POST.get('new_password')
    repeat_password = request.POST.get('repeat_password')
    if request.user.check_password(old_password):
        if not new_password:
            msg = '新密码不能为空'
        elif new_password != repeat_password:
            msg = '两次密码不一致'
        elif old_password == new_password:
            msg = '新旧密码相同'
        else:
            request.user.set_password(new_password)
```

writing/views.py > user_update(request)

```
            request.user.save()
            msg = 'success'
    else:
        msg = '原密码输入错误'
    return HttpResponse(msg)
```

修改密码下拉菜单选择修改信息，如图 6.1-5 所示。

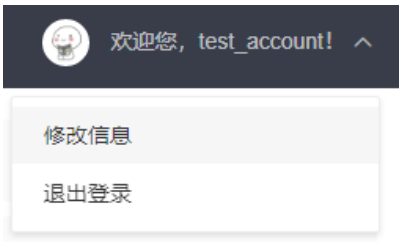

图 6.1-5 下拉菜单选择修改信息

修改密码合法性测试如图 6.1-6 所示，经测试验证，修改密码功能正常，用户可以使用修改密码后的用户密码进行登录，旧密码失效。

图 6.1-6 修改密码合法性测试

6.1.4 用户注销

用户注销链接非独立页面，每个前端页面的导航栏右侧下拉菜单均有注销链接，主要前端代码如代码 6.1-7 所示。

代码 6.1-7 注销用户相关前端代码

writing /templates/writing/.html*

```
<div class="layui-header">
    <ul class="layui-nav">…</ul>
    <ul class="layui-nav layui-layout-right">
        <li class="layui-nav-item layui-hide layui-show-md-inline-
block">
            <a href="javascript:;">…</a>
            <dl class="layui-nav-child">
                <dd><a href="{% url 'writing:user' %}">修改信息
</a></dd>
                <dd><a href="{% url 'user:logout' %}">退出登录</a></dd>
```

writing /templates/writing/.html*

```
            </dl>
        </li>
    </ul>
</div>
…
```

注销相关后端代码如代码 6.1-8 所示。

代码 6.1-8 注销用户后端代码

user/view.py > logout(request)

```
def logoutView(request):
    logout(request)
    return redirect('user:login')
```

用户注销下拉菜单效果如图 6.1-7 所示。

图 6.1-7 下拉菜单选择退出登录

6.2 文章管理

6.2.1 新增文章

新增文章页面 *create.html* 用来创作新的文章，若不采用 AI 生成，可以手动输入文字内容、标题、摘要等，点击立即提交即可进行文章新建保存到文章数据库中。这里还使用 JS 实现对标题摘要合适长度的判断。AI 生成部分见后面章节。新增文章前端页面主要代码如代码 6.2-1 所示。

代码 6.2-1 新增文章前端页面主要代码

writing/template/writing/create.html

```
<form action="" class="layui-form" lay-filter="edit">
    {% csrf_token %}
    <div class="layui-row layui-col-space15">

        <div class="layui-col-md12">
            <div class="layui-card">
                <div class="layui-card-header">文章属性</div>
                <div class="layui-card-body">
                    <div class="layui-form-item">
```

writing/template/writing/create.html

```
                    <div class="layui-inline">
                        <label class="layui-form-label">作者名
</label>
                        <div class="layui-input-inline">
                            <input type="text" name="author"
value="{{ username }}"
                                   class="layui-input"
                                   disabled
                                   style="background-color: #C9C9C9;
color: #009f95">
                        </div>
                    </div>
                </div>
                <div class="layui-form-item">

                </div>

                <div class="layui-form-item">
                    <label class="layui-form-label">标题</label>
                    <div class="layui-input-block">
                        <input type="text" name="title" lay-
verify="title" value=""
                               placeholder="请输入标题" class="layui-
input">
                    </div>
                </div>

                <div class="layui-form-item layui-form-text">
                    <label class="layui-form-label">摘要</label>
                    <div class="layui-input-block">
                        <textarea name="abstract" lay-
verify="abstract" placeholder="请输入内容"
                                  class="layui-textarea"></textarea>
                    </div>
                </div>

                <div class="layui-form-item">
                    <label class="layui-form-label">类型</label>
                    <div class="layui-input-block">
                        <select name="category">
                            <option value="党建" selected="">党建
</option>
                            <option value="科技">科技</option>
```

writing/template/writing/create.html

```
                            <option value="国际">国际</option>
                            <option value="文化">文化</option>
                            <option value="经济">经济</option>
                        </select>
                    </div>
                </div>

                <div class="layui-form-item">
                    <label class="layui-form-label">关键词</label>
                    <div class="layui-input-block">
                        <input type="text" name="keyword" value=""
                               placeholder="关键词" class="layui-
input">
                    </div>
                </div>

                <div class="layui-form-item">
                    <div class="layui-input-block">
                        <button type="button" id='back'
                                class="layui-btn layui-btn-primary
layui-border-green">返回列表
                        </button>
                        <button type="button" class="layui-btn
layui-btn-normal"
                                id="LAY-component-form-setval">AI 生成
                        </button>
                        <button type="submit" class="layui-btn" lay-
submit="" lay-filter="btn_edit">立即提交
                        </button>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="layui-col-md12">
        <div class="layui-card">
            <div class="layui-card-header">文章新建</div>
            <div class="layui-card-body">
                <textarea name="body" id="article_edit" lay-
verify="content"
                          style="display: none;"></textarea>
```

writing/template/writing/create.html

```
            </div>
        </div>
    </div>
</form>
…

<script>
    layui.use(['form', 'layedit'], function () {
        var form = layui.form
            , layer = layui.layer
            , layedit = layui.layedit;

        //建立编辑器
        var editIndex = layedit.build('article_edit', {
            tool: ['left', 'center', 'right', '|', 'strong',
'italic', 'underline', 'del']
        });

        //自定义验证规则
        form.verify({
            title: function (value) {
                if (value.length < 4) {
                    return '标题至少得 4 个字符啦';
                } else if (value.length > 60) {
                    return '标题最多 60 个字符啦';
                }
            },
            abstract: function (value) {
                if (value.length < 5) {
                    return '摘要至少得 2 个字符啦';
                } else if (value.length > 500) {
                    return '摘要最多 500 个字符啦';
                }
            },
            content: function (value) {
                return layedit.sync(editIndex);
            }

        });


        //提交事件
        form.on('submit(btn_edit)', function (data) {
```

writing/template/writing/create.html

```
            $.ajax({
                url: "{% url 'writing:add'%}",
                type: 'post',
                data: data.field,
                dataType: "text",
                success: function (result) {
                    // console.log(data)
                    console.log("result: " + result)
                    if (result == 'success') {
                        layer.msg('提交成功', {icon: 1});
                    } else {
                        layer.msg('提交失败', {icon: 2});
                    }
                }
            });
            return false;//阻止表单跳转
        });

        layui.$('#LAY-component-form-setval').on('click', function
() {
            var content = layedit.getContent(editIndex)
            console.log("content=>", content)

        $('#back').on('click', function () {
            window.location.href = '{% url 'writing:list' %}';
        });

    });
</script>
```

添加文章后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *add* 函数，正确时返回 success，前端接收到后弹出成功提示，回到登陆界面，该部分的函数代码如代码 6.2-2 所示。

代码 6.2-2 新增文章后端程序代码

writing/view.py > add(request)

```
@login_required(login_url='/login/')
def add(request):
    try:
        Article.objects.create(title=request.POST.get('title'),
                               author=request.POST.get('author'),
                               abstract=request.POST.get('abstract'),
                               category=request.POST.get('category'),
```

writing/view.py > add(request)

```
                                body=request.POST.get('body'),
                                keyword=request.POST.get('keyword'))
        return HttpResponse("success")
    except:
        return HttpResponse("fail")
```

创建文章页面及测试情况如图 6.2-1 所示，同时对不符合要求的标题的摘要进行提交测试，经验证新建文章功能有效。

图 6.2-1 新增文章界面及其测试

6.2.2 查询文章

查询文章页面 *writing.html* 用来总览当前用户已经创建过的所有文章，使用了 Layui 的数据表格对数据进行展示。页面中还提供了按照文章类型搜索、按标题内容搜索、按正文内容搜索等查询方法对符合条件的文章进行查询，并且可以将文章进行分页显示。查询文章前端页面主要代码如代码 6.2-3 所示。

代码 6.2-3 查询文章前端页面主要代码

writing/template/writing/writing.html

```
...
```

writing/template/writing/writing.html

```
<div class="layui-inline">
    <div class="layui-inline">
        <button type="button" class="layui-btn" id="add">新建文章
</button>
    </div>
    <div class="layui-inline">
        <form action="" class="layui-form">
            {% csrf_token %}
            <div class="layui-inline">
                <label class="layui-form-label">文章类型</label>
                <div class="layui-input-block">
                    <select name="search_category">
                        <option value="" selected=""></option>
                        <option value="党建">党建</option>
                        <option value="科技">科技</option>
                        <option value="国际">国际</option>
                        <option value="文化">文化</option>
                        <option value="经济">经济</option>
                    </select>
                </div>
            </div>
            <div class="layui-inline">
                <label class="layui-form-label">搜索类型</label>
                <div class="layui-input-block">
                    <select name="search_type">
                        <option value="" selected=""></option>
                        <option value="title">按标题搜索</option>
                        <option value="abstract">按摘要搜索</option>
                        <option value="body">按内容搜索</option>
                    </select>
                </div>
            </div>
            <div class="layui-inline">
                <div class="layui-input-inline">
                    <input name="search" autocomplete="off"
id="input_search" class="layui-input">
                </div>
                <button type="submit" class="layui-btn" id="search"
lay-submit="" lay-filter="search">搜索
                </button>
            </div>
        </form>
    </div>
```

writing/template/writing/writing.html

```
</div>
<table id="article" lay-filter="article" lay-
data="{id:'article'}"></table>
…
<script>
    layui.use(['table', 'form'], function () {
        var table = layui.table;
        var form = layui.form;
        var token_value = $('[name="csrfmiddlewaretoken"]').val();
        {#console.log(token_value);#}
        table.render({
            elem: '#article'
            , height: 500
            , url: '{% url 'writing:query'%}'
            , method: 'post'
            , where: {'csrfmiddlewaretoken': token_value}
            , page: true //开启分页
            , request: {
                pageName: 'pageNo' //页码的参数名称，默认：page
                , limitName: 'pageSize' //每页数据量的参数名，默认：limit
            }
            , cols: [[ //表头
                {type: 'checkbox', fixed: 'left'}
                , {field: 'id', title: '文章 ID', width: 100, sort:
true}
                , {field: 'title', title: '标题', width: 200, sort:
true}
                , {field: 'category', title: '类型', width: 100, sort:
true}
                , {field: 'author', title: '作者', width: 100, sort:
true}
                , {field: 'abstract', title: '摘要', width: 300, sort:
true}
                , {field: 'keyword', title: '关键词', width: 200, sort:
true}
                , {field: 'body', title: '内容', width: 300}
                , {
                    field: 'create_time',
                    title: '创建时间',
                    width: 180,
                    sort: true,
                    templet: "<div>{%
```

writing/template/writing/writing.html

```
verbatim %}{{ layui.util.toDateString( d.create_time, 'yyyy-MM-dd
HH:mm:ss') }}{% endverbatim %}</div>"
            }
            , {
                field: 'update_time',
                title: '修改时间',
                width: 180,
                sort: true,
                templet: "<div>{%
verbatim %}{{ layui.util.toDateString( d.update_time, 'yyyy-MM-dd
HH:mm:ss') }}{% endverbatim %}</div>"
            }
            , {field: 'right', title: '操作', width: 150, toolbar:
'#barDemo', fixed: 'right'}
        ]]
    });
…
    form.on('submit(search)', function (data) {
        table.reload('article', {
            url: '{% url 'writing:query'%}',
            method: "post",
            page: {curr: 1},
            where: data.field,
            request: {
                pageName: 'pageNo',  //页码的参数名称，默认：page
                limitName: 'pageSize' //每页数据量的参数名，默认：limit
            },
        })
        return false;
    });
});
</script>
```

查询文章后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *query* 函数，正确时返回一个含有信息的 json 文件，其中包含一个含有正确查询结果的列表，前端接收到后查询 msg 可以弹出成功提示，如果未查询好 msg 为 0，前端返回错误提示。在查询的同时，还考虑了分页问题，可以按照指定的数量对查询结果进行分页查询。该部分的函数代码如代码 6.2-4 所示。

代码 6.2-4 查询文章后端程序代码

writing/view.py > query(request)

```
@login_required(login_url='/login/')
def query(request):
```

writing/view.py > query(request)

```
    pageNo = int(request.POST.get("pageNo"))
    pageSize = int(request.POST.get("pageSize"))
    left = (pageNo - 1) * pageSize
    right = pageNo * pageSize
    type = request.POST.get("search_type")
    search = request.POST.get("search")
    category = request.POST.get("search_category")
    if not category:
        category = ""
    if type == 'title':
        article =
Article.objects.filter(author=request.session.get('username'),
category__contains=category,
                                    title__contains=search)
    elif type == 'abstract':
        article =
Article.objects.filter(author=request.session.get('username'),
category__contains=category,
                                    abstract__contains=search)
    elif type == 'body':
        article =
Article.objects.filter(author=request.session.get('username'),
category__contains=category,
                                    body__contains=search)
    else:
        article =
Article.objects.filter(author=request.session.get('username'),
category__contains=category)
    context = {
        "code": 0,
        "msg": "",
        "count": article.count(),
        "data": list(article[left:right].values())
    }
    return JsonResponse(context)
```

总览界面如图 6.2-2 所示，对查询所有文章与条件查询进行测试，经过验证，查询功能有效。

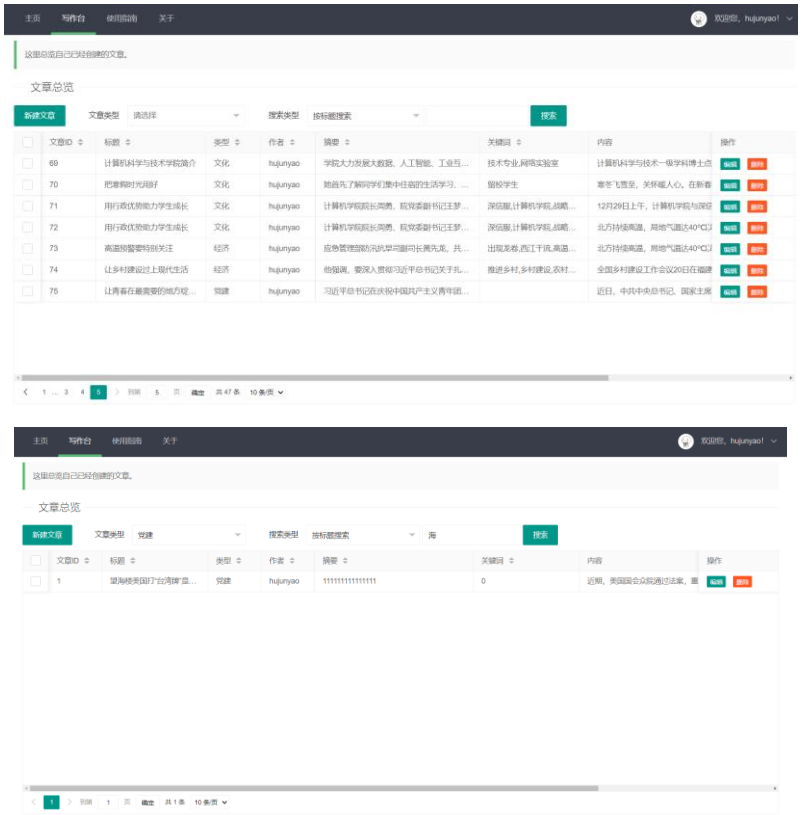

图 6.2-2 查询所有文章与条件查询测试

6.2.3 删除文章

删除文章依托查询总览页面 *writing.html*，选择页面中 Layui 数据表格的当前数据行的删除图标即可删除当前行对应地文章。删除文章前端页面主要代码如代码 6.2-5 所示。

代码 6.2-5 删除文章前端页面主要代码

writing/template/writing/writing.html

```
table.on('tool(article)', function (obj) {
    var data = obj.data
        , layer = layui.layer;
    // 删除
    if (obj.event === 'del') {
        layer.confirm("确定删除这行数据吗？", function () {
            $.ajax({
                url: "{% url 'writing:delete' %}",
                method: "post",
                data: {'article_id': data.id},
                success: function (res) {
                    if (res === "success") {
                        layer.msg('删除成功', {icon: 1});
                        obj.del();
                    } else {
                        layer.msg('删除失败', {icon: 2});
```

writing/template/writing/writing.html

```
                }
            },
            async: false
        })
        layer.close();
    });

} else if (obj.event === 'edit') {
    …
}
});
```

添加文章后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *delete* 函数，正确时返回 success，前端接收到后弹出成功提示，回到登陆界面，该部分的函数代码如代码 6.2-6 所示。

代码 6.2-6 删除文章后端程序代码

writing/view.py > delete(request)

```
@login_required(login_url='/login/')
def delete(request):
    if request.GET.get('article_id'):

Article.objects.filter(pk=request.GET.get('article_id')).delete()
        return HttpResponse("success")
    else:
        return HttpResponse("fail")
```

对总览界面的删除文章功能进行测试，经验证，删除功能有效，测试截图如图 6.2-3 所示。

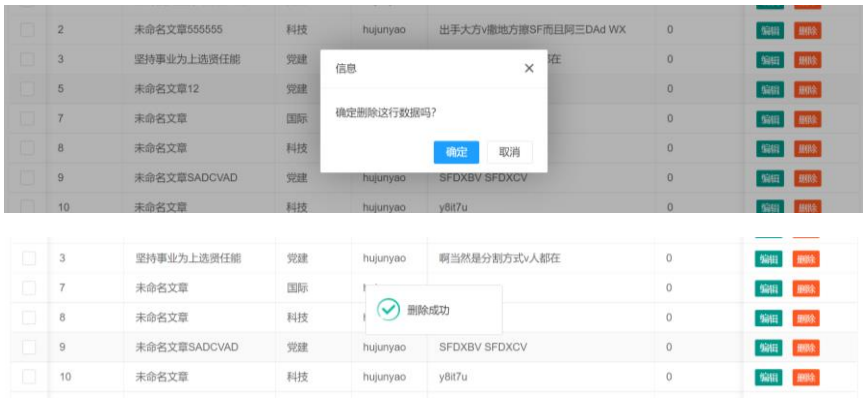

图 6.2-3 删除文章测试

6.2.4 修改文章

修改文章页面 *update.html* 与 *create.html* 极为相似，故不再展示，其最大的不同就是修改页面是通过 iframe 来实现的，而不是在新的页面中进行更新。修改文章依托查询总览页面 *writing.html*，选择页面中 Layui 数据表格的当前数据行的修改图标即可修改当前行对应地文章，进入弹窗页面。修改文章前端页面主要代码如代码 6.2-7 所示。

代码 6.2-7 修改文章前端页面主要代码

writing/template/writing/writing.html

```
table.on('tool(article)', function (obj) {
    var data = obj.data
        , layer = layui.layer;
    if (obj.event === 'del') {
        …
    } else if (obj.event === 'edit') {
        layer.open({
            type: 2
            , title: '文章属性编辑'
            , maxmin: true
            , content: "../" + data.id
            , area: ['1000px', '620px']
            , cancel: function (index, layero) {
                table.reload('article');
            }
        });
    }
});
```

添加文章后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *update* 函数，正确时返回 success，前端接收到后弹出成功提示，回到登陆界面，该部分的函数代码如代码 6.2-8 所示。

代码 6.2-8 修改文章后端程序代码

writing/view.py > update(request)

```
@login_required(login_url='/login/')
def update(request):
    try:
        selected_article =
Article.objects.get(pk=request.POST['id'])
        selected_article.abstract = request.POST['abstract']
        selected_article.title = request.POST['title']
        selected_article.body = request.POST['body']
        selected_article.category = request.POST['category']
        selected_article.keyword = request.POST['keyword']
        selected_article.save()
```

writing/view.py > update(request)

```
        return HttpResponse("success")
    except:
        return HttpResponse("fail")
```

修改弹窗如图 6.2-4 所示，同时对修改功能进行测试，经验证，可以修改文章相关属性，修改功能正常。

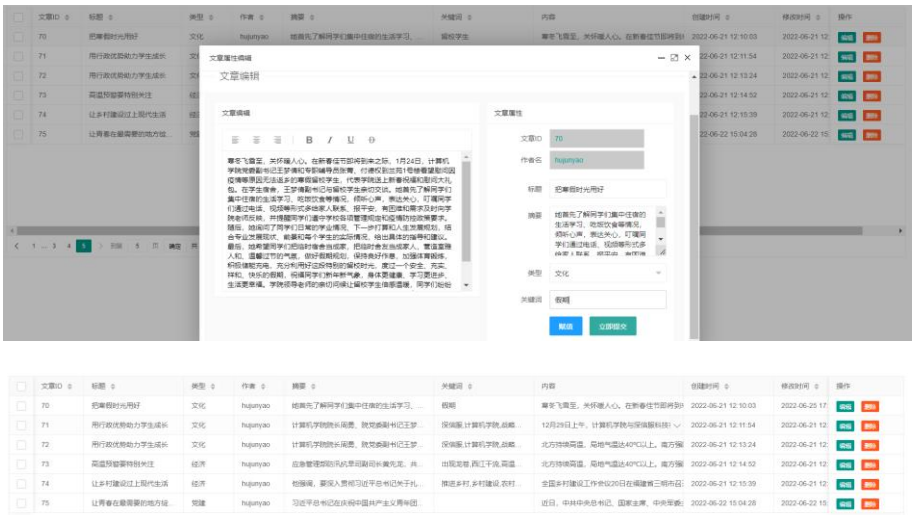

图 6.2-4 修改文章测试

6.3 用户主页

6.3.1 数据可视化（以近一周创作量为例）

用户主页 *index.html* 是由多个 Layui 卡片构成的，每一个卡片显示一定的数据，以创作量卡片为例，展示了全站和用户近一周的文章新建次数。

前端主要是需要接收后端传送来的横坐标（七个日期名称列表）、近一周登录用户的创作量列表、全站用户的创作量列表。这里需要使用 ECharts 和混合条形折线图进行实现，其主要前端代码如代码 6.3-1 所示。

代码 6.3-1 一周创作量前端主要代码

writing/template/writing/index.html

```
<div class="layui-card">
    <div class="layui-card-header">一周写作量统计</div>
    <div class="layui-card-body">
        <div id="main" style="width: auto; height:300px;"></div>
        <script type="text/javascript">
            var myChart =
echarts.init(document.getElementById('main'));
            var option = {
                ...
```

writing/template/writing/index.html

```
                title: {
                    text: '近一周新建文章数量',
                },
                …
                xAxis: {data: {{ list_recent_a_week | safe}}},
                yAxis: [
                    {
                        type: 'value',
                        name: '用户',
                        position: 'left',
                        alignTicks: true,
                    },
                    {
                        type: 'value',
                        name: '全站',
                        position: 'right',
                        alignTicks: true,
                    }
                ],
                series: [
                    {

                        name: '用户',
                        type: 'bar',
                        data: {{ data_user | safe }}
                    },
                    {
                        name: '全站',
                        type: 'line',
                        yAxisIndex: 1,
                        data: {{ data_all | safe }}
                    }
                ]
            }
            myChart.setOption(option);
        </script>
    </div>
</div>
```

查询后端程序要通过 writing 应用的 *urls.py* 找到 *views.py* 文件中的 *index* 函数，其中有一部分函数负责查询近一周创作量的功能，将查询的数据最后返回给前端，该部分的函数代码如代码 6.3-2 所示。

代码 6.3-2 近一周创作量后端程序代码

writing/view.py > index(request)

```
@login_required(login_url='/login/')
def index(request):
    username = request.session.get('username')

    data_user = []
    data_all = []
    for i in range(-6, 1):
        start = datetime.date.today() + datetime.timedelta(days=i)
        end = datetime.date.today() + datetime.timedelta(days=i + 1)

data_user.append(Article.objects.filter(create_time__range=(start,
end), author=username).count())

data_all.append(Article.objects.filter(create_time__range=(start,
end)).count())
    list_recent_a_week = [(datetime.date.today() +
datetime.timedelta(days=i)).strftime('%m-%d') for i in range(-6,
1)]
…

return render(request, 'writing/index.html', locals())
```

近一周创作量卡片效果如图 6.3-1 所示，经验证，可以准确显示近一周相关数据，查询和展示功能正常。

图 6.3-1 近一周创作量卡片效果

6.3.2 热点关键词词云展示

用户主页 *index.html* 的另一种 Layui 卡片，展示了全站近一周的文章的关键词。前端根据日期获取保存在服务器的图片，后端每天需要查询一次数据库，利用 wordcloud 库创作出新的词云图。

热点关键词词云前端如代码 6.3-3 所示，获得当天日期的图片即可。

代码 6.3-3 热点关键词词云前端主要代码

writing/template/writing/index.html

```
<div class="layui-card">
            <div class="layui-card-header">全站近七天写作热点</div>
            <div class="layui-card-body">
                <div id="main3"
                     style="   height: 300px;display: flex;align-
items: center;justify-content: center;">
                    <img src="{% static
'media/web/wordcloud' %}/wordcloud_{{ today | safe }}.png"
style="max-width: 100%;
max-height: 100%;">
                </div>
            </div>
        </div>
```

热点关键词词云后端如下所示，获得当天日期、读取关键词、生成、保存、命名词云图片即可。其代码如代码 6.3-4 所示。

代码 6.3-4 热点关键词词云后端代码

writing/view.py > index(request)

```
@login_required(login_url='/login/')
def index(request):
…
# 近七天所有人文章内容关键词统计云图，一天只绘制一个，减少计算量
today = datetime.date.today().strftime("%y%m%d")
static_root = "static"
filename = static_root + "/media/web/wordcloud/wordcloud_" + today
+ ".png"
if not os.path.exists(filename):
    start = datetime.date.today() - datetime.timedelta(days=7)
    end = datetime.date.today()
    all_text =
list(Article.objects.filter(create_time__range=(start,
end)).values_list("keyword", flat=True))
    text = ','.join(all_text)

    wordcloud = WordCloud(
        background_color="white",
        max_words=50,
        width=1200,
        height=800,
        # mask=background,
        font_path= static_root + '/media/font/font.TTF'
```

writing/view.py > index(request)

```
    )
    wordcloud.generate(text)
    wordcloud.to_file(static_root +
"/media/web/wordcloud/wordcloud_" +
datetime.date.today().strftime("%y%m%d") + ".png")

…
return render(request, 'writing/index.html', locals())
```

热点关键词词云展示效果如图 6.3-2 所示，可以较为直观的展现出近期的热点话题如疫情防控、高校毕业、高考志愿填报等。

图 6.3-2 热点关键词词云展示

6.3.3 爬虫热点话题榜单展示（以中国矿业大学新闻网为例）

用户主页 *index.html* 的另一种 Layui 卡片，展示了主要网站近期较为重要的话题消息或者新闻（百度、B 站、中国矿业大学新闻网）。前端主要使用 Layui 静态表格展现列表，后端使用爬虫功能对榜单或者信息列表进行标题和链接的爬取。

热点话题榜单前端如代码 6.3-5 所示，使用 Django 模板语言对后端传送的列表进行循环输出即可。

代码 6.3-5 热点话题榜单前端主要代码

writing/template/writing/index.html

```
<div class="layui-card">
    <div class="layui-card-header">全网热搜榜单</div>
    <div class="layui-card-body">
        <div class="layui-tab layui-tab-brief" lay-
filter="docDemoTabBrief">
            <ul class="layui-tab-title">
                …
                <li><img src="{% static 'media/web/cumt.png' %}"
height="22px"></li>
```

writing/template/writing/index.html

```
            </ul>
            <div class="layui-tab-content">
                …
    <div class="layui-tab-item" style="height:
180px;overflow:auto;">
    <table class="layui-table" lay-even="" lay-skin="nob">
    <colgroup><col width="20"><col width="200"><col></colgroup>
    <thead><tr><th>序号</th><th>矿大新闻</th></tr></thead><tbody>
                    {% for cumt in cumt_link_and_title %}
                        <tr>
                            <td>{{ forloop.counter }}</td>
                            <td><a
href="http://xwzx.cumt.edu.cn{{ cumt.0 }}"
target="_blank">{{ cumt.1 }}</a></td>
                        </tr>
                    {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
```

热点话题榜单后端如代码 6.3-6 所示，设置好爬虫所需配置（浏览器标识、cookie 等），使用 requests 库访问相应链接，使用 BeautifulSoup4 库选择对应地标题和链接进行保存，然后传送给前端。

代码 6.3-6 热点话题榜单后端代码

writing/view.py > index(request)

```
kv = {'user-agent': *******}
SEARCH_NUM = 20
url = "http://xwzx.cumt.edu.cn/513/list.htm"
selector = 'span.Article_Title > a'
response = requests.get(url, headers=kv)
response.encoding = response.apparent_encoding
response = response.text
cumt = BeautifulSoup(response,
features="lxml").select(selector)[:20]
cumt_link_and_title = []
for c in cumt:
    cumt_link_and_title.append([c.get("href"), c.get_text()])
```

热点话题榜单展示效果如图 6.3-3 所示，可以较为直观的展现出近期的新闻，方便用户直接了解其他网站当前热门事件便于写作。

图 6.3-3 热点话题榜单展示效果

6.4 AI 生成模型

AI 生成模型的触发在新建文章页面，点击 AI 生成按钮将调用其后端程序。前端主要代码如代码 6.4-1 所示。

代码 6.4-1 AI 生成前端主要代码

writing/template/writing/create.html

```
layui.$('#LAY-component-form-setval').on('click', function () {
    var content = layedit.getContent(editIndex)
    var loading_index = layer.load(0, {shade: false});
    $.ajax({
        url: "{% url 'writing:generate'%}",
        type: 'post',
        data: {'content': content},
        dataType: "text",
        success: function (result) {
            var obj = eval("(" + result + ")");
            if (obj.code === 100) {
                layer.msg('生成成功', {icon: 1});
                form.val('edit', {
                    "title": obj.title,
                    "abstract": obj.summary,
                    "keyword": obj.keyword,
                });
            } else {
                layer.msg('生成失败', {icon: 2});
            }
            layer.close(loading_index);
        }
    });
});
```

AI 生成模型前端触发 ajax 后要通过 *urls.py* 找到 *views.py* 文件中的 *generate* 函数，先将传送来的消息的 HTML 标记语言通过正则表达式去除，再调用生成标题、摘要、关键词的功能，将其以 json 文件的格式返回至页面更新。其代码如代码 6.4-2 所示。

代码 6.4-2 AI 生成模型后端调用 API 代码

writing/view.py > generate(request)

```
def generate(request):
    text = request.POST.get('content')
    text = re.sub(r'(\\n)|(<.*?>)', '', text)
    text = re.sub(r"\s+", " ", text)
    print(text)

    summary = get_summer(text)
    print(summary)
    keyword = get_key(text)
    print(keyword)
    title = get_title(text)
    print(title)

    code = 100
    msg = {
        'code': code,
        'title': title,
        'summary': summary,
        'keyword': keyword
    }

    return JsonResponse(msg)
```

上述代码中的 `get_summer`、`get_key` 和 `get_title` 均为调用相关封装好的 AI 算法，从而生成对应地摘要、关键词、标题，具体算法已在前面章节介绍，不再重述。主要有关的两个函数是 *generate_title.py* 和 *keyword_and_summary.py*。

keyword_and_summary.py 的代码如代码 6.4-3 所示。使用了 textrank 方法生成关键词和摘要。

代码 6.4-3 *keyword_and_summary.py* 代码

keyword_and_summary.py

```
# 基本参数
maxlen = 512
batch_size = 1
epochs = 1
```

keyword_and_summary.py

```
# bert 配置
config_path = 'title_model/chinese_wobert_L-12_H-768_A-12/bert_config.json'
checkpoint_path = 'title_model/chinese_wobert_L-12_H-768_A-12/bert_model.ckpt'
dict_path = 'title_model/chinese_wobert_L-12_H-768_A-12/vocab.txt'

def get_title(content):
    # 加载并精简词表，建立分词器
    token_dict, keep_tokens = load_vocab(
        dict_path=dict_path,
        simplified=True,
        startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
    )
    tokenizer = Tokenizer(token_dict, do_lower_case=True)

    class CrossEntropy(Loss):
        """交叉熵作为loss，并mask掉输入部分
        """

        def compute_loss(self, inputs, mask=None):
            y_true, y_mask, y_pred = inputs
            y_true = y_true[:, 1:]  # 目标token_ids
            y_mask = y_mask[:, 1:]  # segment_ids，刚好指示了要预测的部分
            y_pred = y_pred[:, :-1]  # 预测序列，错开一位
            loss = K.sparse_categorical_crossentropy(y_true, y_pred)
            loss = K.sum(loss * y_mask) / K.sum(y_mask)
            return loss

    model = build_transformer_model(
        config_path,
        checkpoint_path,
        application='unilm',
        keep_tokens=keep_tokens,  # 只保留keep_tokens中的字，精简原字表
    )

    output = CrossEntropy(2)(model.inputs + model.outputs)

    model = Model(model.inputs, output)
    model.compile(optimizer=Adam(1e-5))

    # model.summary()
```

keyword_and_summary.py

```
    class AutoTitle(AutoRegressiveDecoder):
        """seq2seq 解码器
        """

        @AutoRegressiveDecoder.wraps(default_rtype='probas')
        def predict(self, inputs, output_ids, states):
            token_ids, segment_ids = inputs
            token_ids = np.concatenate([token_ids, output_ids], 1)
            segment_ids = np.concatenate([segment_ids,
np.ones_like(output_ids)], 1)
            return model.predict([token_ids, segment_ids])[:, -1]

        def generate(self, text, topk=1):
            max_c_len = maxlen - self.maxlen
            token_ids, segment_ids = tokenizer.encode(text,
maxlen=max_c_len)
            output_ids = self.beam_search([token_ids, segment_ids],
                                          topk)  # 基于beam search
            return tokenizer.decode(output_ids)

    autotitle = AutoTitle(start_id=None,
end_id=tokenizer._token_end_id, maxlen=80)

    model.load_weights('title_model/model2/best_model.weights')
    # model.summary()
    content = get_summer_loc(content)
    pred_title = ' '.join(autotitle.generate(content, 1)).lower()
    pred_title = pred_title.replace(" ", "")
    return pred_title
```

generate_title.py 的代码如代码 6.4-4 所示。其将读入关键字句，并使用 bert 进一步生成新的标题。

代码 6.4-4 *generate_title.py* 代码

generate_title.py

```
# 基本参数
maxlen = 512
batch_size = 1
epochs = 1

# bert 配置
config_path = 'title_model/chinese_wobert_L-12_H-768_A-
```

generate_title.py

```
12/bert_config.json'
checkpoint_path = 'title_model/chinese_wobert_L-12_H-768_A-
12/bert_model.ckpt'
dict_path = 'title_model/chinese_wobert_L-12_H-768_A-12/vocab.txt'

def get_title(content):
    # 加载并精简词表，建立分词器
    token_dict, keep_tokens = load_vocab(
        dict_path=dict_path,
        simplified=True,
        startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
    )
    tokenizer = Tokenizer(token_dict, do_lower_case=True)

    class CrossEntropy(Loss):
        """交叉熵作为loss，并mask掉输入部分
        """

        def compute_loss(self, inputs, mask=None):
            y_true, y_mask, y_pred = inputs
            y_true = y_true[:, 1:]  # 目标token_ids
            y_mask = y_mask[:, 1:]  # segment_ids，刚好指示了要预测的部分
            y_pred = y_pred[:, :-1]  # 预测序列，错开一位
            loss = K.sparse_categorical_crossentropy(y_true, y_pred)
            loss = K.sum(loss * y_mask) / K.sum(y_mask)
            return loss

    model = build_transformer_model(
        config_path,
        checkpoint_path,
        application='unilm',
        keep_tokens=keep_tokens,  # 只保留keep_tokens中的字，精简原字表
    )

    output = CrossEntropy(2)(model.inputs + model.outputs)

    model = Model(model.inputs, output)
    model.compile(optimizer=Adam(1e-5))

    # model.summary()

    class AutoTitle(AutoRegressiveDecoder):
        """seq2seq解码器
```

generate_title.py

```
        """

        @AutoRegressiveDecoder.wraps(default_rtype='probas')
        def predict(self, inputs, output_ids, states):
            token_ids, segment_ids = inputs
            token_ids = np.concatenate([token_ids, output_ids], 1)
            segment_ids = np.concatenate([segment_ids,
np.ones_like(output_ids)], 1)
            return model.predict([token_ids, segment_ids])[:, -1]

        def generate(self, text, topk=1):
            max_c_len = maxlen - self.maxlen
            token_ids, segment_ids = tokenizer.encode(text,
maxlen=max_c_len)
            output_ids = self.beam_search([token_ids, segment_ids],
                                          topk)  # 基于beam search
            return tokenizer.decode(output_ids)

    autotitle = AutoTitle(start_id=None,
end_id=tokenizer._token_end_id, maxlen=80)

    model.load_weights('title_model/model2/best_model.weights')
    # model.summary()
    content = get_summer_loc(content)
    pred_title = ' '.join(autotitle.generate(content, 1)).lower()
    pred_title = pred_title.replace(" ", "")
    return pred_title
```

部署至服务器后，标题生成结果测试及其响应时间记录如图 6.4-1 所示，生成时间通常为 38.83s/篇，速度较慢，但生成效果较好。通过定量计算，最终得出算法的评估指标为：Rouge-1 为 0.3041，Rouge-2 为 0.1970，Rouge-L 为 0.3872，计划后期进行改进优化。

图 6.4-1 标题生成结果测试及其响应时间记录

6.5 本章小结

本章对详细设计中的模块进行了编码实现，遵循了“高耦合，低内聚”的开发原则，详细地描述了系统的开发流程。从用户管理、文章管理、用户主页、AI 模型生成四个角度，讲解了系统的前端页面代码、后端 Django 代码以及与对数据库进行增删改查的代码，并展示了对应的前端页面的图片。

七、总结与展望

7.1 项目总结

优点

本次项目我们使用了 textrank 算法与 NLP 中的 Bert_UniLM 模型生成了文章的摘要、关键词以及标题，前端使用了 Layui 进行界面的搭建，同时利用 ecahrt 进行可视化的数据展示，后端使用 Django 框架进行逻辑处理。整体上做到了标题、摘要、关键词的生成较为准确，前端的 UI 设计合理、美观，数据可视化程度高，系统运行流畅。对于题目的要求而言，完成度较高。

不足

1. 模型训练仍不够充分，主要是在时间方面效果不佳，没有算力，计算速度慢。模型训练部分可以对训练数据进行数据增强处理，以此来增强模型的鲁棒性以及使得模型更好的满足我们的需求。

2. 利用 Layui 进行前端界面开发时，可以整合重复代码、降低代码复杂度、去除无用变量等，简化程序代码，增强代码可阅读性。

3. 前端界面可以进一步美化，并引入更多的特色功能。

7.2 下一步计划

1. 文本分类算法的引入

除了自动生成标题、摘要以及关键词外，我们希望加上文本分类的深度学习算法与模型，增加系统的多样性，更好地满足用户地需求。

2. 搜寻更大的数据集

搜寻更大的数据集，训练目标检测模型，提高模型的检测精度，减少模型检测的缺帧率。

3. 网站验证码以及用户指南的嵌入

计划加入图形化验证码，目前的验证码由 JS 随机生成，无法防止恶意攻击。同时为了增加网站的易用性，计划增加用户指南页面，让用户能够轻松上手。

八、参考资料

[1] Mihalcea R , Tarau P . TextRank: Bringing Order into Texts[J]. 2004.
[2] Sutskever I , Vinyals O , Le Q V . Sequence to Sequence Learning with Neural Networks[C]// NIPS. MIT Press, 2014.
[3] Shi X , Chen Z , Wang H , et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting[C]// MIT Press. MIT Press, 2015.
[4] Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need[C]// arXiv. arXiv, 2017.
[5] Radford A . Language Models are Unsupervised Multitask Learners.
[6] Brown T B , Mann B , Ryder N , et al. Language Models are Few-Shot Learners[J]. 2020.
[7] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.
[8] Devlin J , Chang M W , Lee K , et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.
[9] Dong L , Yang N , Wang W , et al. Unified Language Model Pre-training for Natural Language Understanding and Generation[C]// Neural Information Processing Systems. 2019.
[10] Diao S , Bai J , Song Y , et al. ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations[J]. 2019.