大学生服务外包
创新创业大赛

第十五届中国大学生服务外包创新创业大赛

GrammarGPT：基于大语言模型（LLM）的文本智能批改系统

项目详细方案

参赛选题：【A04】基于人工智能大模型技术的智慧教育应用产品设计【移动创新院】
团队编号：2404106　　团队名称：GrammarGPT
团队成员：李嘉鹏、王颖彬、高朗、崔昊阳、赵亚泽
指导老师：李瑞轩

目录

目录 ---- 1
1 引言和项目背景 ---- 1
1.1 概述 ---- 1
1.2 相关研究案例分析 ---- 2
1.3 项目可行性分析 ---- 3
1.4 人员管理和项目进度管理 ---- 3
2 创意描述 ---- 6
3 特色综述 ---- 8
4 实现思路 ---- 11
4.1 需求分析 ---- 11
4.2 原型系统设计 ---- 12
5 功能简介和系统结构 ---- 16
5.1 功能简介 ---- 16
5.2 系统结构 ---- 16
5.3 关键函数设计 ---- 19
5.4 关键算法设计 ---- 22
5.5 数据管理说明 ---- 23
6 开发工具与技术 ---- 24
6.1 数据集收集和构建 ---- 24
6.2 微调原理与参数 ---- 26
6.3 WEB 应用实现 ---- 27
7 系统实现 ---- 28
7.1 应用对象和环境 ---- 28
7.2 代码管理 ---- 28
7.3 关键函数说明 ---- 30
7.4 测试计划和测试用例 ---- 33
7.5 系统整体效果分析 ---- 34
7.6 用户反馈 ---- 38
8 结语 ---- 39
8.1 核心问题解决程度的定论 ---- 39
8.2 未来展望 ---- 39
参考文献 ---- 40
附录 ---- 41
附录 A 关键代码 ---- 41
附录 B 用户使用说明书与知识产权相关信息 ---- 54
附录 C 文本纠错标注手册 ---- 65

1 引言和项目背景

1.1 概述

随着人工智能技术的快速发展，生成式 AI（AIGC）凭借其出色的内容生成能力，已在教学领域进行了一定程度的应用，并展现出了巨大的潜力。除了促进中小学教育的均衡和普及，基于 AIGC 的“虚拟教师”也能够在本科与研究生阶段的“高等教育”中焕发光彩。AIGC 作为一种潜在的虚拟教师，能根据教学的不同环节和不同场景独特的特点和需求，提供合适的个性化教学服务，真正实现教育增效。这不仅有助于提高学生的学习效率和教师的工作效率，也能为教育带来革命性的变革。

近年来，本科及硕士研究生项目招生规模逐年递增，科研教育越发成为广受关注的教育模式。与此同时，本科生与研究生在教师指导下发表中文学术论文及学位论文的产量也越发增加。**一篇中文学术论文在发表前，往往需要经历教师与学生的多次打磨和修改，双方均需要为了纠正表述的规范性、逻辑结构的严密性以及语言的流畅性等而投入大量时间，效率、精度均较低。**在紧张的研究学习周期中，这些修改要求检查者反复通读论文，从而给导师和学生带来了不必要的重复性教学劳动和巨大的时间开销，严重影响了专业理论的学习和应用。**这是本项目需解决的核心问题。**

自 2022 年 3 月以来，以 GPT-3.5 为代表的大语言模型逐渐步入公众视野，这类模型展现出强大的自然语言理解、推理与生成能力，为科研和高等教育注入了新的活力。论文的检查与修改作为科研教育“评”的环节，需要在保证高效率的同时发挥教学功能。为此，**我们以大语言模型为核心，面向中文学术学位论文的智能修正和批改构建了一套高性能虚拟助教系统**。在该系统中，学生或教师只需输入 docx/pdf/txt 格式的中文论文稿，很快即可获得一份完备的、有较高参考价值的论文修改意见清单，其中**标注了所有潜在的语法错误和拼写错误，并且系统还会结合上下文语境对每段文字提出专业的修改建议**。这样一来，教师能够将更多的精力关注到学生的研究进展上，学生仅需等待数分钟便可获得专业的论文指导意见，在加速学术论文写作与修改的同时也学习了科研类文章的写作方法，锻炼了逻辑能力、思辨能力和创造力，从而能更加专注地投入到研究与发现的旅程中。

目前该项目已经作为“虚拟助教”，分析并修正了多份本科生课程设计实验报告以及研究生中文学术论文等文稿，且给出了专业且广受师生认同的修改意见，获得了学生及教师的高度评价。在数据集方面，我们采取多种方式生成了一个大规模的自动标注和人工标注混合的、贴合教学批改场景的中文病句-改句训练数据集，设计了全自动的中文病句-改句训练数据生产流水线。而在模型方面，我们对 ChatGLM-6b、Phoenix-inst-chat-7b 等大模型进行了全参数微调，调研并使用了 ERNIE 4.0、InternLM-chat-20b-sft 等大模型，并针对不同子任务选择了最优模型。从社会角度看，该虚拟助教系统填补了中文学术领域自动化论文批改系统的空白，提高了学术生产效率，实现了“千人千面”的个性化指导意见推荐，也实现了教育资源共享；从高等教育角度来说，该虚拟助教系统使高校研究型教育和学习变得更加紧凑，减轻了教师与学生撰写和批改论文的负担，解决了教学中的重复工作，让学生在享受高效论文修正服务的同时提高科研写作能力，从而使得教师与学生能更多地将时间花费在打磨科研项目中，产出更有影响力的工作，赋能智慧教育。

1.2 相关研究案例分析

目前，人工智能在教育领域的应用越来越广泛。对于文本自动纠错与批改领域，我们调研了国内外同类型项目的研究现状，掌握了目前该领域的大致样貌。

训练数据层面　当前，中文语病语料库存在的主要问题有二：总体样本数量较小、数据集精细化程度不高。目前公开的中文文本错误数据集还不充分，尤其是针对具体错误类型的数据集。并且这部分数据集的语料取材大多较为广泛，没有针对具体错误类型进行深入研究。

第一，针对拼写纠错模型的训练，相关研究主要选用了最新的 SIGHAN 基准数据集。这是一个存在 22000 条错误样例的文本数据集，包含 461 种类型的拼写错误。它的主要问题是精细化程度不高，只能满足通用纠错模型的训练，在科研论文这种修改限制高、拼写错误隐蔽的领域不能体现很好的测试效果。

第二，现有语法错误数据集普遍较少。对于本项目所涉及的中文学术类文本纠错，目前还缺乏能准确表现论文常见语法错误的自动生成样本算法。因此首先我们采用人工标注的方法生成一个小规模高质量的数据集；接着采用 self-instruct 方法利用 ERNIE 4.0 模型进行数据集自动生成，在保证数据集质量的基础上扩大数据集的规模，使之可以用于 Phoenix-inst-chat-7b 模型的微调。

模型与算法层面　近年来，随着机器学习算法模型不断改进，文本纠错问题取得了显著进展。2022 年，Zhengxiao Du 等提出了自回归空白填充预训练的通用语言模型（GLM），该模型通过添加二维位置编码和允许任意顺序预测跨度，改进了空白填充预训练，从而在自然语言理解、条件生成和无条件生成的任务中，在相同模型大小和数据的情况下优于 BERT、T5 和 GPT 等大模型，并且在语句纠错方面表现出很大潜力。GLM 具体运行机制可参考图 1。

Phoenix-inst-chat-7b 模型是由香港中文大学（深圳）开发的预训练模型，擅长以特定角色与用户对话，这为我们将模型迁移到高等教育的背景下和中文论文智能批改的任务下创造了条件。

以上多种不同模型在特定任务上的特征为本项目选择 ChatGLM-6B、Phoenix-inst-chat-7b 等模型作为文本纠错模型基准进行改进和微调提供了充分理论依据。

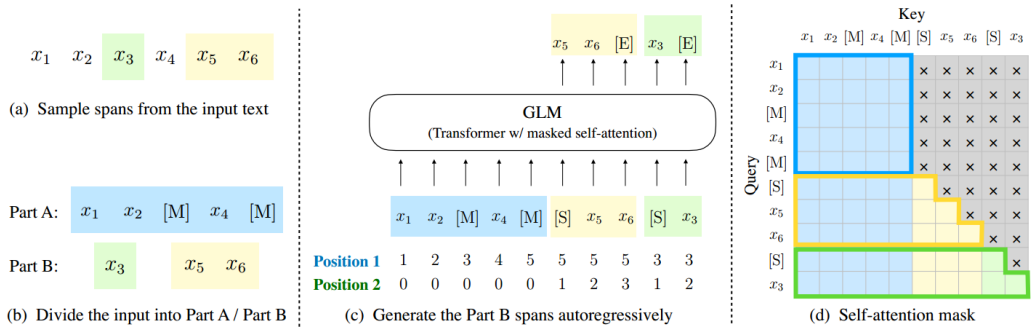


图 1　GLM 模型运行机制示例

1.3 项目可行性分析

技术可行性 本项目要求实现文本智能纠错，需要即时输出反馈结果，因此需要较高算力。目前，我们拥有 8 张 NVIDIA GeForce RTX 3090 显卡和一定量的 NVIDIA A100 计算级显卡，具有充足的算力资源。同时，为了呈现出简单易用的用户界面，还需要进行前后端的设计与对接、云端服务部署等。

团队成员均具有丰富的计算机领域专业知识，如数据挖掘、大数据分析、机器学习、深度学习、前后端应用开发、软件测试与迭代等，在项目推进过程中具有良好的团队凝聚力和共同目标。团队成员积极参与创新创业实践，并取得了丰富成果。

针对本项目，团队成员已有的技术积累包括 VS Code 的使用、远程服务器 ssh 的连接与使用、Python / C++ / TypeScript 多语言编程、大模型微调、全栈开发（包括 vue、flask 架构等）等，具有合作设计开发项目系统并进行调试的专业能力。目前项目已产出可用系统（demo 演示见视频），证明本项目具有充分的技术可行性。

经济可行性 本项目的定位是一种方便科研人员进行论文修改的 WEB 应用。在开发完成后，我们将会首先进行小范围推广，扩大项目的知名度与影响力，并首先在校园内启动试用。试用期后我们将针对性地收集用户反馈，并对系统进行优化后再次上线。由于目前市面上小型服务器的租赁成本并不高，因此本项目可以长期在云上进行运营，并通过流量变现等方式获得回报。总体来说，本项目的收益/投资比较高。

综合上述分析，我们认为本项目的开发是必要且可行的。

1.4 人员管理和项目进度管理

通过系统地分析项目目标，本小组共同商议并制定了合理的人员分工安排与项目进度计划，通过对任务设置阶段性期限，确保项目有序推进。项目的开发主要分为四个部分：前期调研阶段、初步探索阶段、分模块实现阶段、试运行、测试和正式上线阶段。不同阶段的不同任务如表 1 所示。

表 1 项目阶段性任务安排

阶段	任务	
前期调研阶段	模型侧	阅读语法纠错和拼写纠错的前沿论文，了解现有工作的优缺点
		查找用于中文语法纠错的数据集
		阅读论文，查找语法错误数据集的生成方法
		爬取论文，构建正确句子数据集
		阅读 LLM 论文，对 LLM 形成初步了解
	前后端侧	进行系统的总体设计，完成需求分析 UML、系统架构图等
		进行前端页面初步设计

初步探索阶段	模型侧	阅读语法纠错任务评价指标前沿论文，找到一个好的评价指标并跑通测试代码
		对比不同的数据集生成算法，找出效果较好的算法
		部署 ChatGLM-6B、Phoenix-inst-chat-7b 和 InternLM-chat-20b-sft 模型，并熟悉其使用
		直接使用 ChatGLM-6B 和 Phoenix-inst-chat-7b 模型进行语法纠错，查看其效果
		调研 InternLM-chat-20b-sft 模型在段落逻辑分析上的特征
		阅读有关模型微调的前沿论文并进行实践，查找效果最好的微调方法，使用 M2 标准进行评测
		尝试用一些微调算法微调 ChatGLM-6B 和 Phoenix-inst-chat-7b 模型，查看效果并对比
	前后端侧	学习使用开源编辑器框架 tiptap
		调研文档格式转换的方法
分模块实现阶段	模型侧	对 ChatGLM-6B 和 Phoenix-inst-chat-7b 模型进行全参数微调
		人工标注小规模的高质量数据集
		撰写中文学术类文本纠错标注手册，便于前期数据集构建
		使用前期找到的数据集生成方法生成大规模数据集用于模型微调
		编写使用微调好的模型进行语法纠错的函数，便于展示系统使用
		将整个模型在服务器上进行正式部署
		与后端配合实现展示系统的后端搭建
	前后端侧	实现数据库操作功能
		实现基本界面框架
		实现用户验证相关的后端支持
		实现用户登录注册界面

		实现个人信息和文件管理界面
		实现基本的编辑器界面并调用模型进行纠错
		完善编辑器功能，支持多种富文本样式
		实现文件管理功能的后端支持
		实现文档格式转换
	进行系统联调	
试运行、测试和正式上线阶段	对系统进行测试，并改正发现的 Bug	
	系统正式上线准备工作	
	系统正式上线运行	

2 创意描述

本项目的创意凝练如下。

创意维度	描述
系统	首个中文学术论文智能批改系统
数据集	构造了大规模中文通用语法错误数据集 设计了全自动的中文病句-改句训练数据生产流水线 提出了多种潜在的领域业务数据采集来源
模型	对多种大模型进行部署和全参数微调，针对特定下游任务选取最优模型 检错、纠错性能达到本领域领先水平
实现	设计、实现了一个 WEB 应用并进行了鲁棒性测试

在本科以及研究生教育阶段，学生往往缺乏系统的中文写作训练，这导致在撰写论文时，频繁发生中文拼写和语法错误。为此，虚拟助教系统需要拥有较为充足的中文语病分析的知识和技能，以便更加准确地对学生的论文进行批改。

在数据集方面，针对缺乏大规模、高质量的、来自实际教育场景下的中文论文语法错误批改数据集的困难，**我们设计了一系列方法生成了一个大规模的自动标注和人工标注混合的、面向教学批改场景的中文病句-改句数据集，并设计了全自动的中文病句-改句数据生产流水线**。具体方法详见第 3 章“特色综述”，需要注意的是，高质量的数据集对模型训练及微调具有极大帮助。

除了通过标准的程序方式进行数据生成之外，我们还可以直接**通过业务数据信息采集的方式获取数据**。例如，本系统接受用户输入并生成批改结果，**用户输入的文件本身及系统批改结果就可以作为训练输入语料**；除此之外，现阶段**互联网上也存在大量开源高质量通用数据集**供选用。我们也可以**通过知网等学术平台获取训练语料**。一些潜在的业务数据来源如图 2-数据所示。

图 2-数据 潜在领域业务数据采集来源

在模型方面，我们对 ChatGLM-6B、Phoenix-inst-chat-7b 等模型进行了全参数微调。相较于 GPT3、ERNIE 等超大参数量级的模型，这些模型参数量较少，训练、推理的时空开销较少，技术经济性较高，因此非常适合虚拟助教系统参与到中文论文智能修正的环节中。参数调试方面，全参数微调相较于 lora、ptuning 等高效参数微调方法，可以调整全部的参数。我们的**多轮测试结果表明，全参数微调在中文论文批改工作中的语法纠错等高难度 NLP 下游任务中表现好于其它微调方式，在检错、纠错等方面均接近学术界 SOTA（State-Of-The-Art）水平**，并且只需要 2 张 80G A100 显卡即可完成全参数微调，所需显存约为 120G。不过，**在一些下游任务（如构造数据、段落逻辑分析）上，我们也根据项目实际情况调研并使用了 ERNIE 4.0、InternLM-chat-20b-sft 等模型**。我们使用的模型种类如图 2-模型所示。

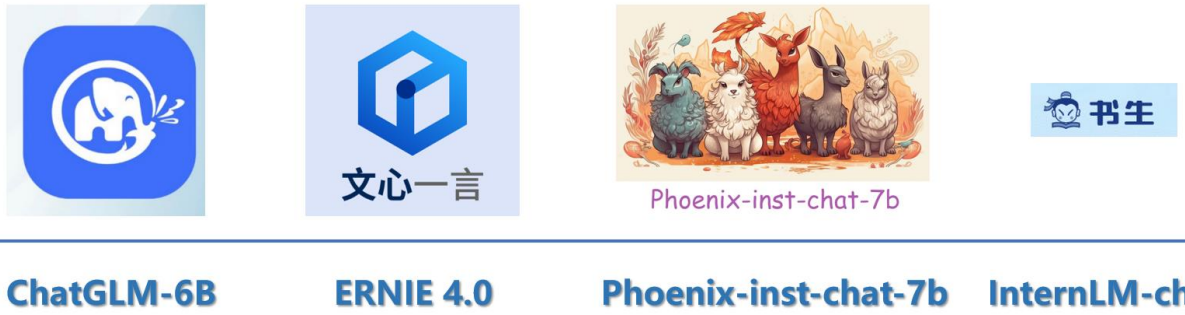


图 2-模型 项目使用的模型种类

在项目应用方面，我们为虚拟助教系统开发了一个**中文科研论文虚拟助教网页端应用（Web 应用）**。多种软件工程方面的测试结果显示，本虚拟助教网站功能齐全、健壮性强、方便学生和教师使用。

针对本项目创新点的详细介绍，请进一步阅读第 3 章“特色综述”、第 4 章“实现思路”、第 5 章“功能简介和系统结构”、第 6 章“开发工具与技术”以及第 7 章“系统实现”。

3 特色综述

本项目的特色主要集中在数据、模型以及系统三个层面。下面将分别进行论述。

数据上：我们构建了一套中文病句-改句数据集和全自动中文病句-改句数据生产流水线。我们将中文病句划分为两类，分别是遵循模板的病句（如“通过...使得...”）和不遵循模板的病句（如歧义、事实性逻辑错误等），并分别针对其特点设计自动生成流水线。

对于前者，我们在网络上广泛搜集初高中语文考试的语病分析试题，从中抽取出超过 25 种病句模板。然后，通过引导推理能力卓越的大模型，向模板中注入不同的内容，从而构建适合不同模板的病句。然后，人为指导模型修正这类病句（即设计提示词 prompt 指导模型对自己生成的语句执行规定的字符串处理操作），以获得对应的改句。

对于后者，我们首先收集若干有实时性错误或歧义的病句，然后识别这些病句中的命名实体进行替换，同时在不引发额外语病的前提下替换谓语、定语等成分。最终，我们将两部分数据混合并得到了完整的微调数据集，共包含 10000 条数据。

上述两种中文病句-改句数据集批量生成的方法示例如图 2 所示。

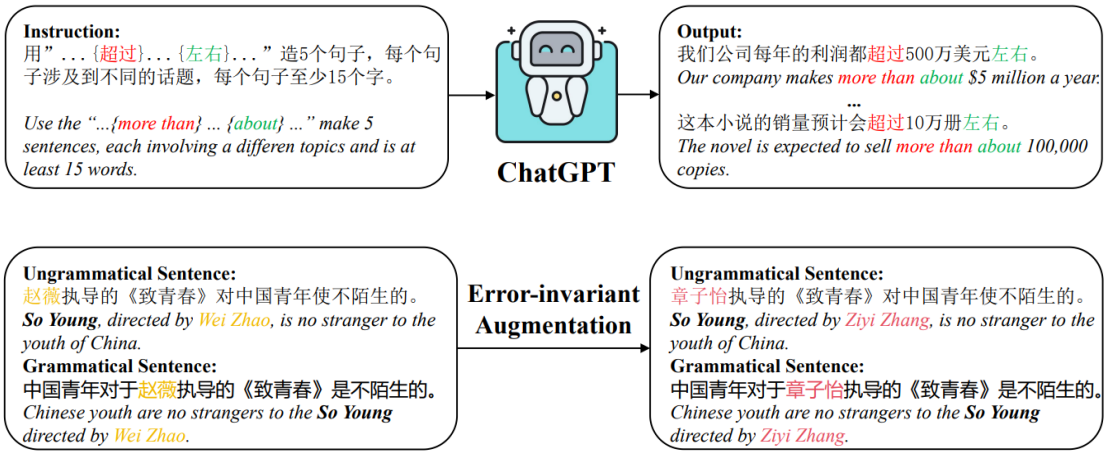


图 2 数据集批量生成流水线方法说明

模型上：我们通过模型的微调获得了强大的中文语法纠错模型，并设计了基于多种文档格式（如 docx/pdf/txt 等）的大模型论文语言分析流水线。这使得系统能精准纠错，并能在短时间内完成分析，输出模型的修改意见文档。

我们用上文提到的中文病句-改句数据集对基座大语言模型（如 ChatGLM-6B、Phoenix-inst-chat-7b 等）进行全参数微调。相比同期有竞争力的工作 S2S-BART，微调后的模型在文本检错、纠错等评价标准上展现出了强悍的性能[13][14]。不同模型之间的性能对比结果如表 2 和图 2-测试所示。

表 2 模型性能对比

Model/Dataset	NaCGEC			NLPCC			MuCGEC		
	Pre	Rec	F0.5	Pre	Rec	F0.5	Pre	Rec	F0.5
GPT3.5-Turbo	28.69	33.92	29.60	36.69	36.26	36.61	10.25	17.83	11.20
ChatGLM-6B	24.60	9.82	18.91	34.31	9.38	22.41	16.98	7.89	13.80
LLaMA-7B	17.63	7.98	14.20	24.54	10.02	19.03	8.44	4.75	7.30
S2S-BART	22.31	10.14	17.99	-	-	-	-	-	-
SOTA	**68.11**	**43.87**	**61.33**	**51.76**	**33.49**	**46.67**	-	-	-
Ours（本项目）	63.99	31.99	47.99	44.48	25.27	36.39	**28.74**	**18.09**	**25.28**

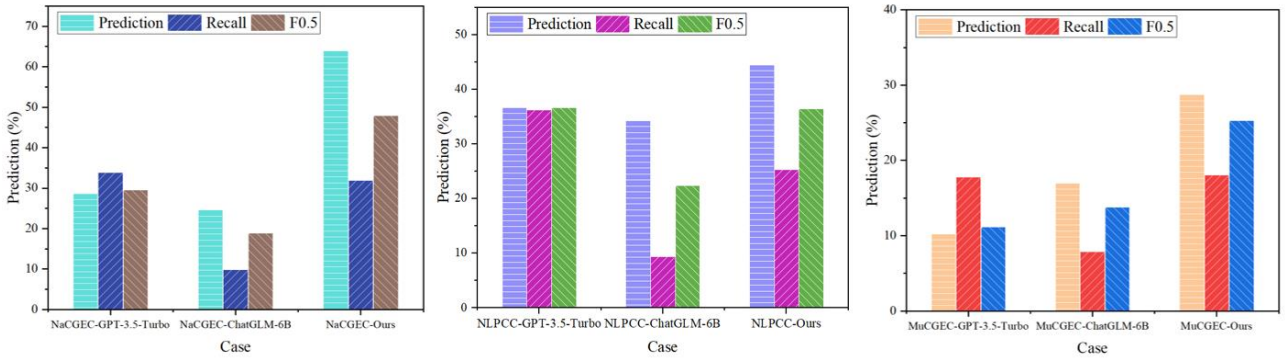


图 2-测试 模型性能对比（每张图右边的一组为本项目微调后模型的性能）

由于不同数据集上达到最优效果的模型也有所不同，考虑到部署成本、管理复杂程度以及执行速度等诸多因素，我们使用更为稳定轻量的微调模型。

对于论文语言分析以及段落逻辑分析任务，我们使用了 InternLM-chat-20b-sft 模型。我们通过 dox2html 解析器，将用户输入的 docx 文档解析成 html 对象，从中提取出篇章结构；之后根据论文结构，每次选择若干段落作为单个子任务，同时进行内部上下文之间的逻辑关系评估和语言质量评估。最后，将模型的评估结果与原文按照固定的格式写入一个 html 格式的报告模板中。当模型将所有的子任务全部输出完毕，我们将该 html 报告模板通过开源转化工具 html2pdf 保存为 pdf 文档，最后从前端返回给用户。这样就实现了文本批改结果的导出与下载功能。

系统上：对于不同的子任务，我们使用了不同的模型以达到最佳效果，这包括 ChatGLM-6B、Phoenix-inst-chat-7b、InternLM-chat-20b-sft 等不同模型。最终我们在系统中集成了所有合适的模型并搭建了 WEB 应用，用户只需输入文字内容或论文文稿，就可以产出实时纠错结果（显示在网页前端界面中）和修改意见报告。

从系统架构角度看，本系统使用前后端分离架构进行开发。由于大语言模型推理的计算量较大但所需的辅存空间较小，而用户信息存储等其它模块计算量较小但所需存储空间较大，所以本系统的后端分成两个部分。系统的大语言模型单独部署在具有 8 块 NVIDIA GeForce RTX 3090 GPU（或更好硬件条件）的高性能服务器上，并以 API 调用的方式为后端的其它部分提供服务。系统的其它部分以及前端应用，例如进行用户身份鉴定和用户文档存储等的模块，部署在磁盘空间为 9.1 T 的存储服务器上。详细的系统结构设计请阅读第 5 章“功能简介和系统结构”。

4 实现思路

4.1 需求分析

经过前期讨论，团队确定了项目开发的总体方向，思维导图 Mindmap 如图 3 所示。

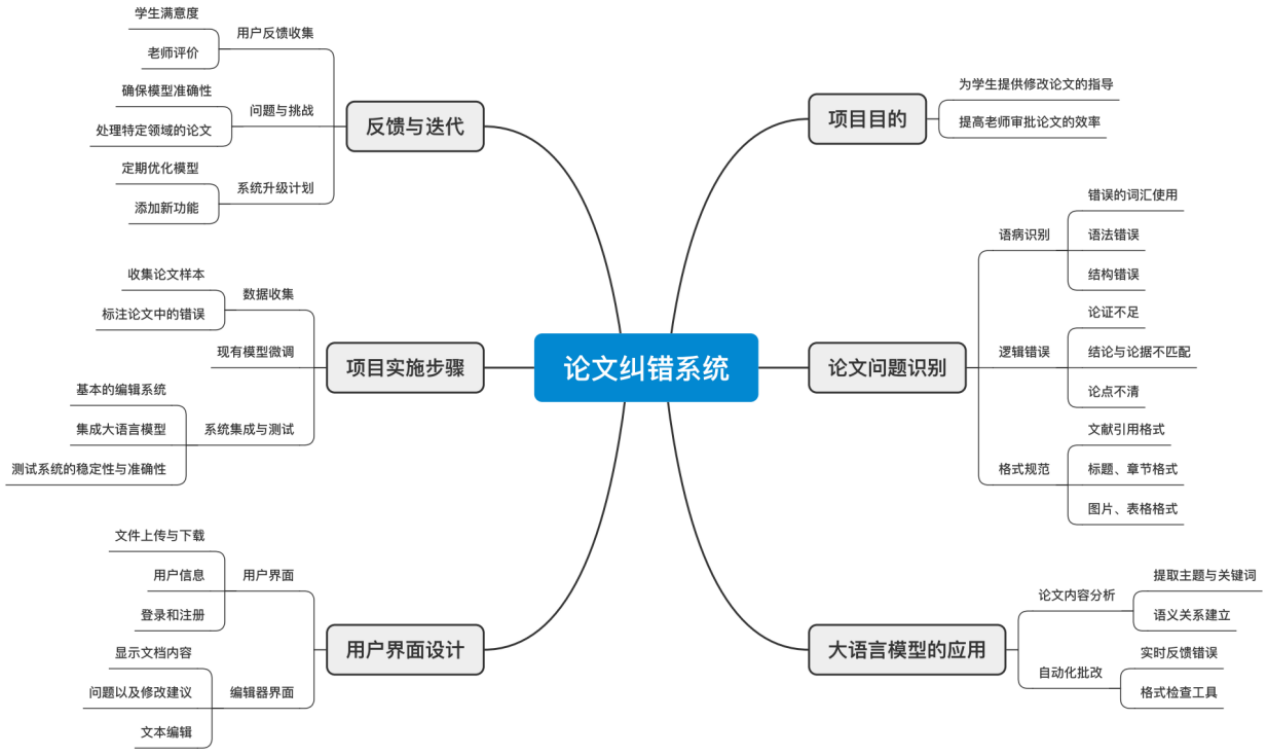


图 3　本项目 Mindmap

用户需求 用户对本系统的需求包括注册账号、上传文件、根据错误识别提示作出接受或不接受操作、保存或下载文件、查看个人中心等。

功能需求 本系统需要完成多项功能，包括用户基本信息登录与注册、文本的上传与下载、文本内容解析、文本编辑器实时显示文档内容和标注错误位置、实时文本编辑、文本存储、设置操作教程等。为了让用户获得良好的使用体验，需要设计简洁明了的用户界面，同时具有良好的引导机制。

非功能需求 ①效率：本系统需要对用户输入进行快速识别与反馈，检错时间控制在 3 秒以内；

②准确性：本系统返回的检错结果需要是准确可靠的，对于典型病句，期望达到 70%以上的错误识别率；

③安全性：本系统将用户注册时提供的用户名、密码等隐私信息加密存储在后端数据库中；

④可移植性：本系统分别设计模型端、前端、后端三个部分，彼此之间相互联系，基本功能函数采用模块化管理，同时设置了完备的部署教程，具有良好的可复用性与可移植性。

4.2 原型系统设计

在正式进行开发工作前，我们运用墨刀工具设计了项目的原型系统（链接：https://modao.cc/proto/slY3rTVjs28orgmrZPxNN/sharing?view_mode=read_only），主要分为“首页”、“登录页面”、“个人中心”、“编辑器”、“团队介绍”五个部分，以便于更准确直观地说明主要功能和用户交互界面，如图 4 所示。

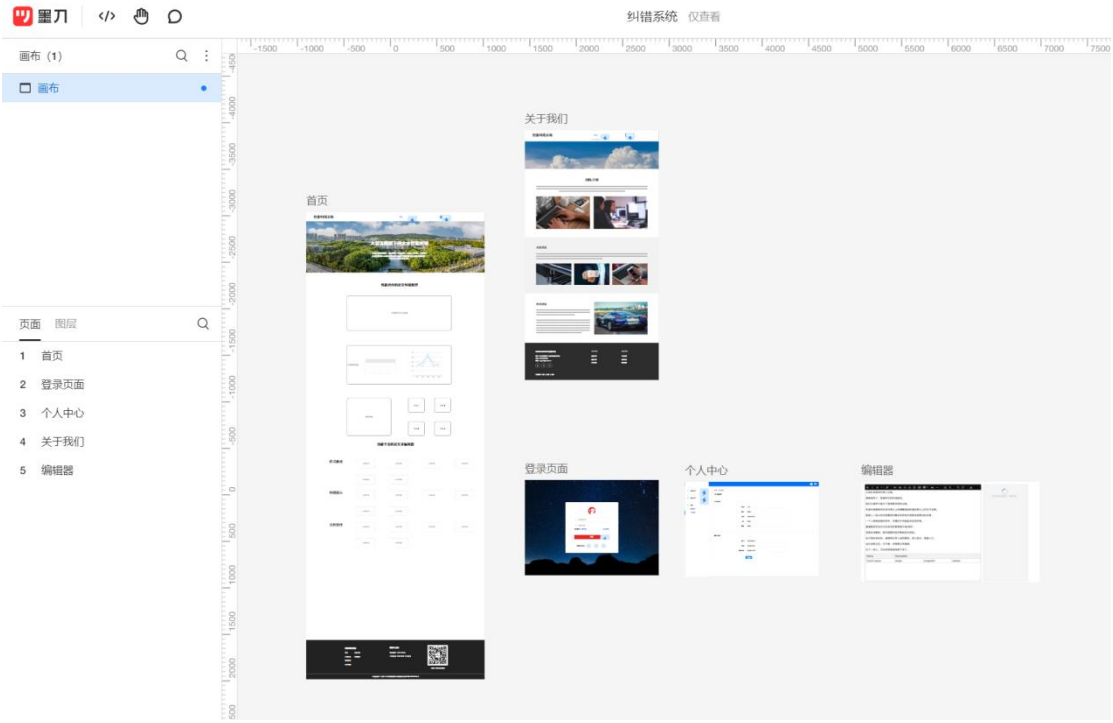


图 4　原型系统总体架构

上述各部分的原型系统设计分别如图 5~图 8 所示。

图 5 “首页”部分原型系统设计

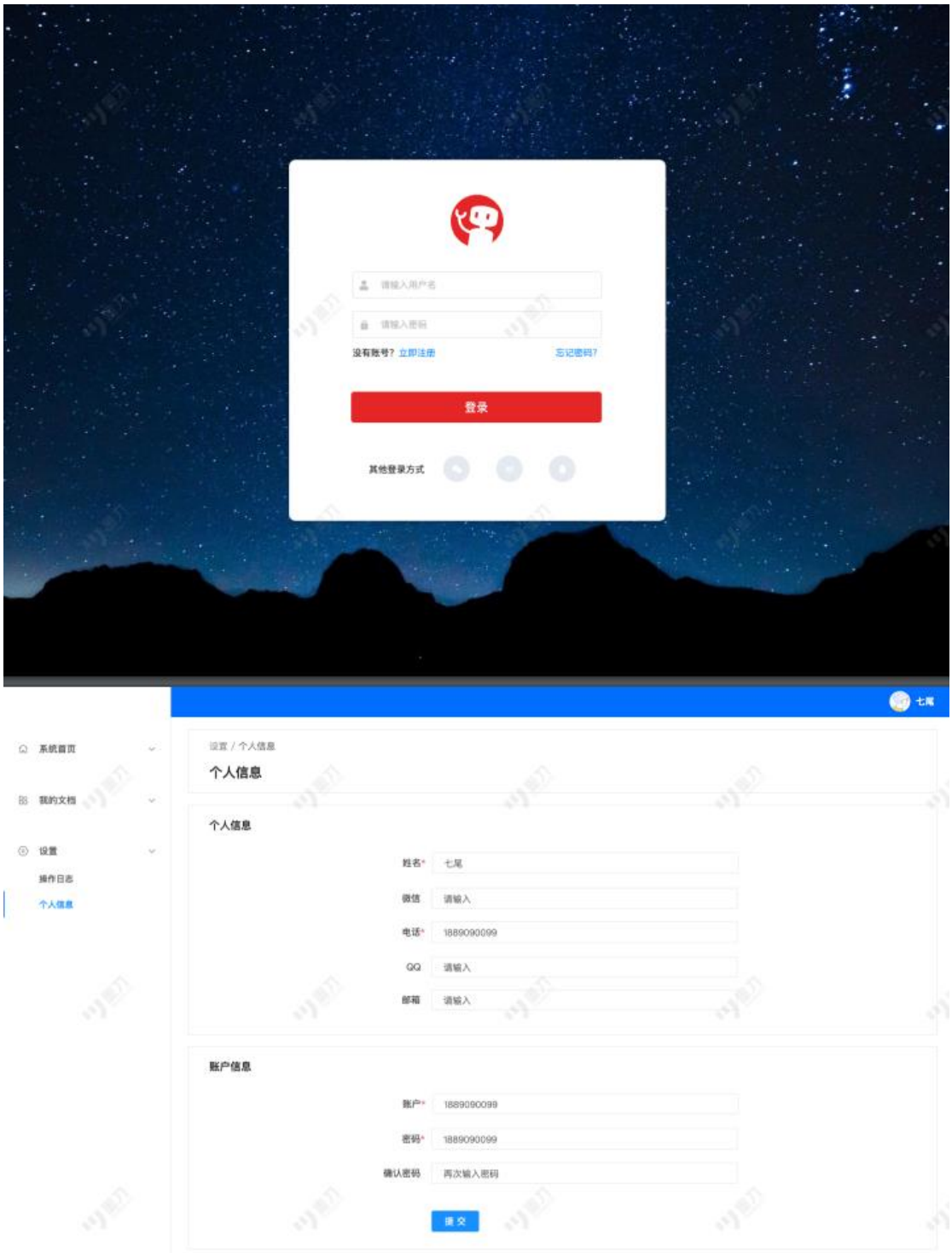


图 6“登录页面”、“个人中心”部分原型系统设计

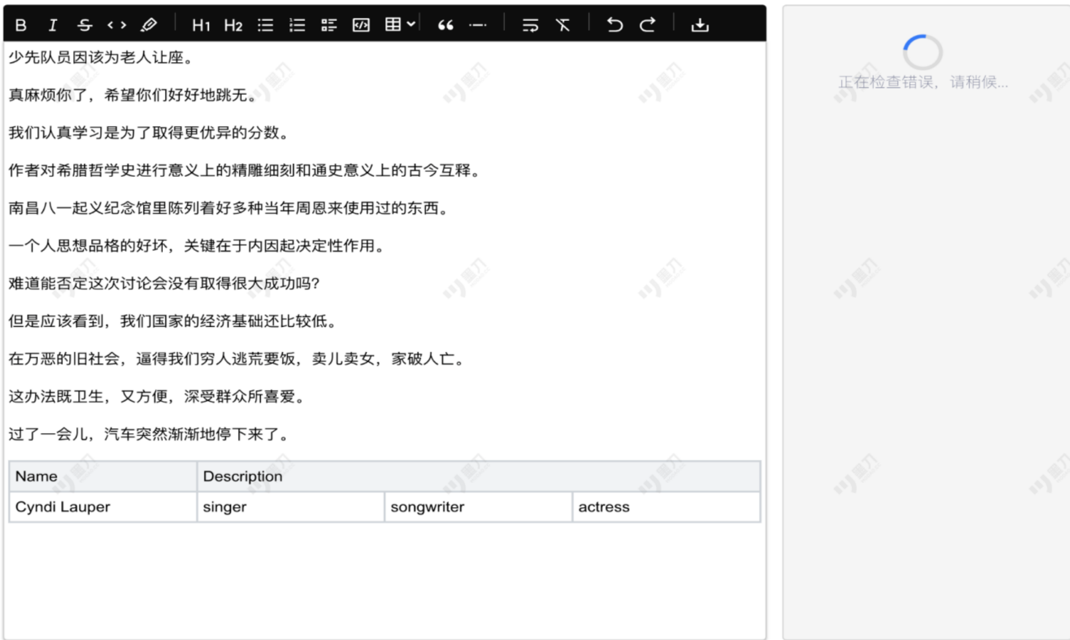


图 7 “编辑器”部分原型系统设计

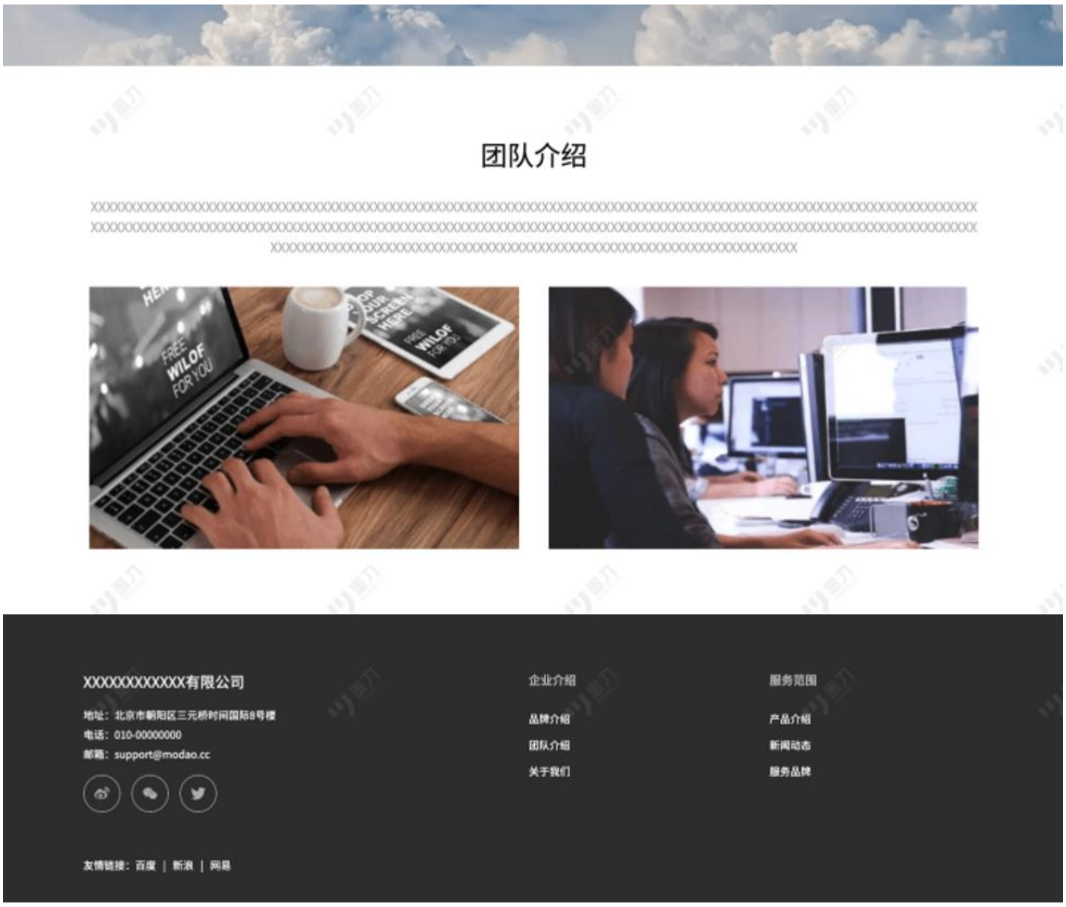


图 8 “团队介绍”部分原型系统设计

5 功能简介和系统结构

5.1 功能简介

本智能批改系统致力于实现以下几个功能：**中文拼写及语法纠错批改、论文语言分析与论文段落逻辑关系分析**。

对于中文拼写以及语法纠错批改任务，我们参考了近期的相关研究工作，对大语言模型 phoenix-inst-chat-7b 在自制的中文病句-改句数据集上开展全参数微调，模型在 MuCGEC、NaCGEC 和 NLPCC 三个通用中文纠错评价基准任务（benchmark）上都取得了本领域领先水平，超越同期有竞争力的工作（如 S2S-BART），在很大程度上能胜任实际应用中对中文文本纠错的需求。

对于论文语言分析任务，我们使用性能更加强大的大语言模型 InternLM-chat-20b-sft。该模型拥有出色的中文语言理解能力和推理能力。作为批改系统的一部分，其可以接受一段中文论文文本，在内容、语言流畅性和逻辑连贯性等方面对其开展分析，最后得出修改意见。

对于段落逻辑关系分析任务，我们同样使用 InternLM-chat-20b-sft。向该模型输入连续的三段论文文本，可以让模型对段落间的逻辑关系进行分类，并且结合写作背景判断该逻辑关系是否通顺，是否需要调换、添加或者删减段落。

5.2 系统结构

5.2.1 系统概述

本系统以一个 WEB 应用的形式实现。用户注册或者登录后，可上传 word/pdf/txt 等格式的待修改论文文档，或者将待修改的论文内容直接粘贴到网站提供的编辑器中。系统将调用后端的 LLM 对论文进行拼写和语法纠错，并交互性地为用户提供修改意见。系统还可以根据用户的要求生成 pdf 格式的修改意见文档。

本系统采用了前后端分离的架构。在这种架构中，前端负责处理用户界面和用户交互，而后端负责处理数据存储和业务逻辑，因此可以将应用程序的前端和后端功能分开开发、部署和维护。前后端分离架构是一种较为先进的 WEB 应用架构，它的几点优势如下：

独立开发 前端和后端可以由不同人员独立开发，提高了开发效率和并行开发能力。

可维护性 前端和后端的代码分离，使系统的维护更加容易，可以针对性地修改和更新其中的一部分。

扩展性 前后端可以分别进行水平扩展，根据实际需求增加前端或后端的实例数量，提高系统的性能和可伸缩性。

技术选型灵活 前后端独立演进，可选择最合适的技术栈和工具，不受限于单一技术栈。

高可用性 前后端分离使得前端应用程序可以以静态文件的形式进行部署，通过内容分发网络实现不同区域间的快速访问。

由于大语言模型推理的计算量较大但所需的辅存空间较小，而用户信息存储等其它模块计算量较小但所需存储空间较大，所以本系统的后端分成两个部分。系统的大语言模型单独部署在具有 8 块 NVIDIA GeForce RTX 3090 GPU 的运算服务器上，并以 API 调用的方式为后端的其它部分提供服务。系统的其它部分以及前端应用（例如进行用户身份鉴定和用户文档

存储等的模块）部署在磁盘空间为 9.1 T 的存储服务器上。

本系统的模块关系如图 9 所示。

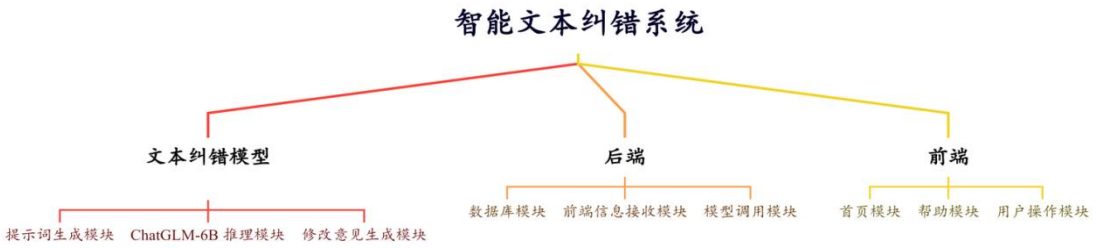


图 9 系统模块关系树

5.2.2 文本纠错模型

本小节主要介绍文本纠错模型的整体结构。

在本系统中，我们使用 ptuning 后的 ChatGLM-6B[1]或 Phoenix-inst-chat-7b 模型进行用户上传的文档拼写和语法错误的纠正。其中 ptuning 所用的大规模标注数据集采用 self-instruct[2]方法，由 ERNIE-Bot 4.0 模型自动生成。

文本纠错模型分为 3 个主要功能模块：提示词生成模块、ChatGLM-6B（或 Phoenix-inst-chat-7b）推理模块以及修改意见生成模块，如图 10 所示。

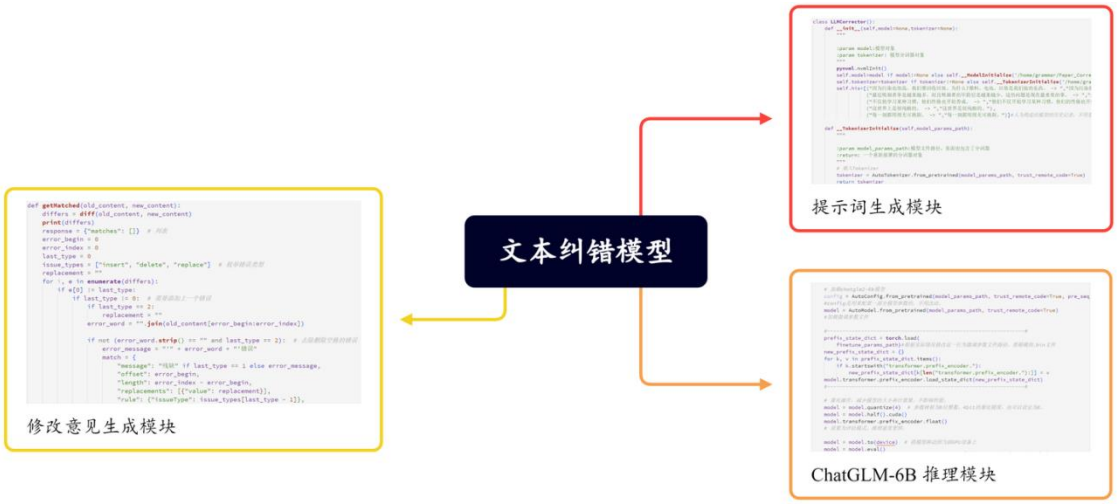


图 10　文本纠错模型系统结构图

提示词生成模块用于根据后端调用 API 时传入的句子来生成输入模型的提示词。我们的提示词由任务类型、范例以及输入实例构成，具体的提示词模板如下：

> prompt = """中文语法纠错任务：将一段中文句子进行修正，如果句子没有明显语病则返回原句。
>
> 下面是一些范例：
>
> 学生大概做飞机去北京。 -> 学生大概坐飞机去北京。\n
>
> 我觉得他很大胆，他不去外国。他觉得如果要改转变中国，所以就要从中国做的研究。 -> 我觉得他很大胆，他不去外国。他觉得如果要改变中国，就要在中国做研究。\n

```
吸烟者反对这样的措施。 -> 吸烟者反对这样的措施。\n

请对下述句子进行修正。返回你修改后的句子，无需其它说明和解释。
[输入实例] ->
"""
```

ChatGLM-6B（或 Phoenix-inst-chat-7b）推理模块用于进行推理，判断句子有无语病、语病的类型，并推理出修改后的正确句子。当文本纠错模型进行初始化时，预先训练好的参数文件会被加载到这个模块中，使模型可以正确回答提示词中指出的任务。这个模块也是整个文本纠错模型的核心模块，它的回答质量和推理所用时间将高度影响到用户的使用体验。

修改意见生成模块根据 ChatGLM-6B（或 Phoenix-inst-chat-7b）推理模块的返回结果，生成格式化的修改意见。这个修改意见作为 API 的返回结果传递给后端。

5.2.3 后端

本小节主要介绍后端模块的整体结构。后端主要由 Flask 框架和 Python 语言开发。

后端主要包括以下 3 个主要功能模块：数据库模块、前端信息接收模块、模型调用模块，如图 12 所示。

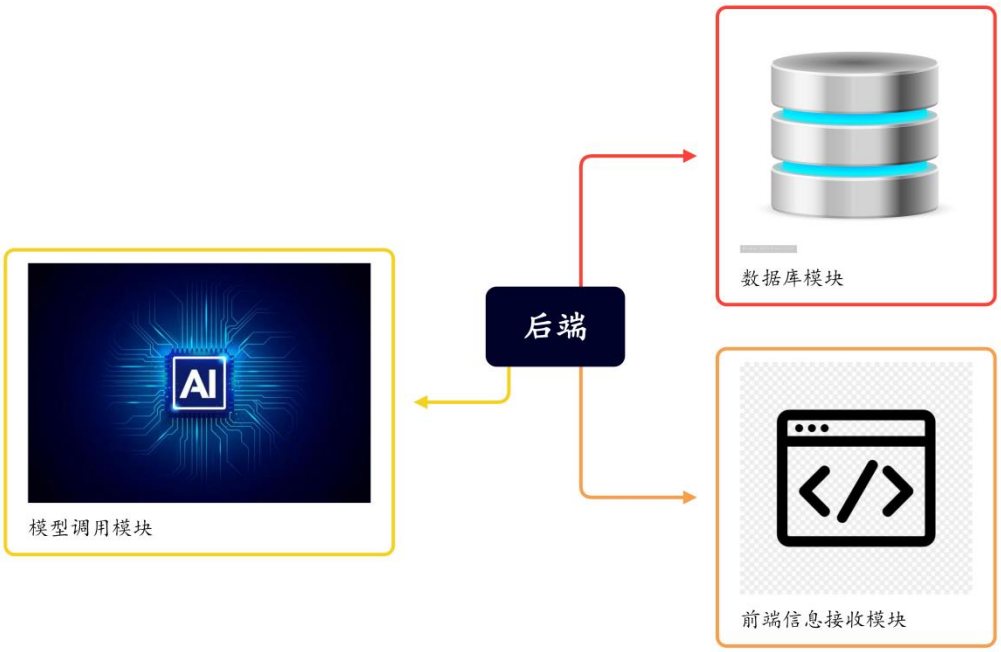


图 11　后端系统结构图

数据库模块是整个系统中最核心的部分，负责存储用户注册信息、记录用户登录信息以及保存用户的文件。我们采用 MySQL 数据库实现数据的增、删、改、查操作。在用户进行注册时，数据库模块会接收用户提供的注册信息，并将其存储在数据库中；当用户登录时，数据库模块会验证用户的登录信息并返回相应结果；用户上传文件后，数据库会保存这些文件，并确保它们的安全性、一致性和完整性。

前端接收模块负责处理前端向后端传达的任务。当用户上传文件时，前端接收模块会接收这些文件并对其进行处理。处理完成后，它会通过模型调用模块获得用户文本的纠错结果。

模型调用模块在接到前端接收模块的调用请求之后，调用文本纠错模型并等待模型的结

果，模型返回结果后，模型调用模块会将其返回给前端接收模块。

5.2.4 前端

本小节主要介绍前端模块的整体结构。在本系统中，前端主要由 Vue 框架、node.js 运行环境和 TypeScript 语言进行开发。

前端页面主要包括 3 个主要模块：首页模块、帮助模块和用户操作模块，如图 13 所示。

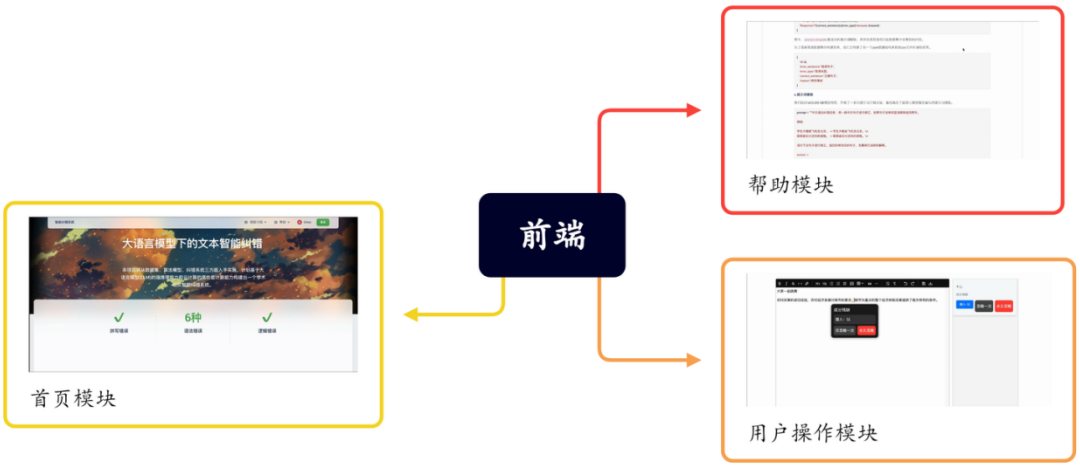


图 12　前端系统结构

其中，首页模块包含首页一个页面；帮助模块包含多个项目展示页以及帮助文档页；用户操作模块包含登录页、用户主页和编辑器页面。下面简要介绍这些页面的内容和功能。

首页模块 在首页模块中，首页页面简单介绍了系统的主要功能和优势，同时提供了用户进入其它页面的入口按钮。

帮助模块 在帮助模块中，各种项目展示页展示了项目的系统结构图、成员名单等相关信息，介绍了项目的开发流程。而帮助文档页为用户提供了整个系统的使用说明来指引用户操作。帮助文档页还为系统管理员提供了部署文档，管理员可根据需求将系统部署在自己的服务器上。

用户操作模块 在用户操作模块中，登录页在用户点击登录按钮后通过 axios 框架向后端服务器发起登录请求，并在登录成功后自动跳转到首页。当用户处于登录状态下时，可以通过点击导航栏中的链接进入个人界面，用户在该界面中可以上传和管理自己的文档。点击“编辑文档”按钮即可跳转到编辑器界面并展示对应文件。编辑器界面主要有编辑模块和纠错模块，其中编辑模块可以修改文本格式，插入图表等；纠错模块负责向后端发送当前文本，并根据返回的结果进行纠错提示的渲染。

5.3 关键函数设计

5.3.1 后端数据结构设计

后端的用户数据存储在数据库中，建表语句如下：

```
CREATE TABLE `userdata` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`username` varchar(20) NOT NULL,
```

```
`password` varchar(255) NOT NULL,
`role` tinyint(1) NOT NULL,
`telephone` varchar(255) NOT NULL,
`email` varchar(255) NOT NULL,
`address` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `telephone` (`telephone`),
UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

5.3.2 前端数据结构设计

前端主要处理用户登录状态相关信息，文件编辑相关信息和纠错结果信息。具体的数据结构参考图13。

图13中，UserDetail是用户信息类，包括用户名、邮箱等用户相关信息。LoginInfo是登录信息类，存有用户名、用户id和其对应的token。在业务中，前端向后端发起登录请求后，如果成功则会收到一个LoginResponse对象，其中的loginInfo存有需要的token等信息，而userDetail字段为空。如果前端发起查询用户信息请求，则会收到userDetail存在的LoginResponse对象，此时loginInfo字段为空。

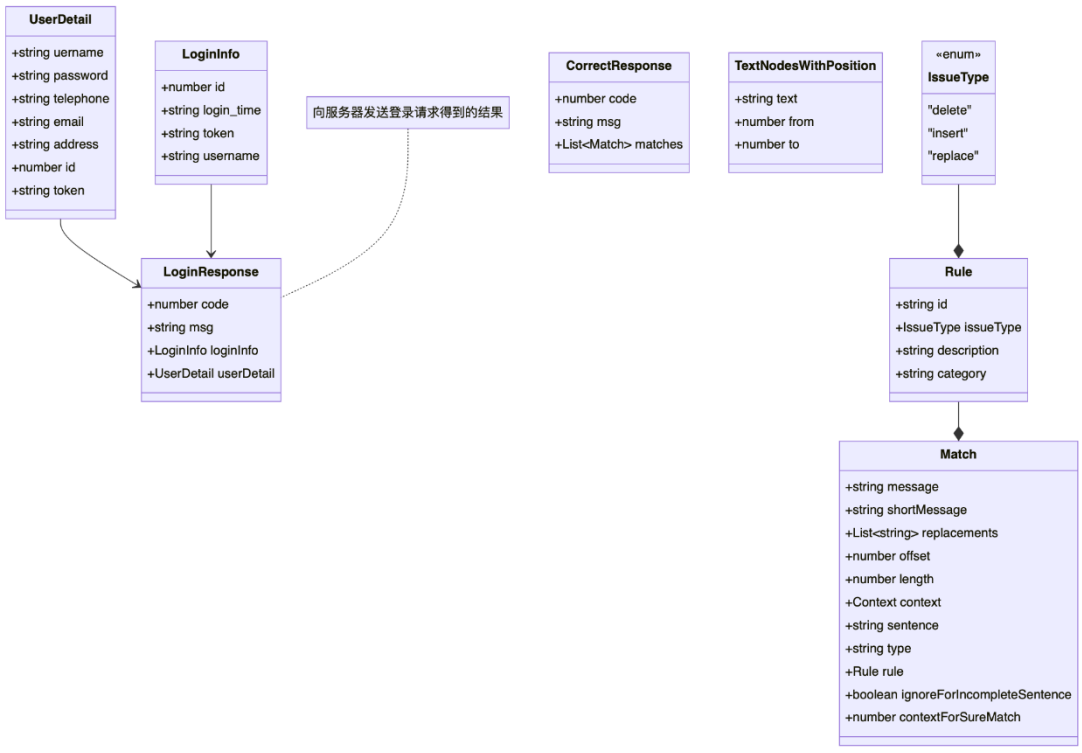


图13　前端数据类图

5.3.3 模型端数据结构设计

为了实现模型的纠错功能，我们构建了一个名为LLMCorrector的类。该类共定义了6个内置函数和1个接口。用户只需要实例化LLMCorrector类，便可通过简单地调用接口完成模型的部署以及循环纠错的功能。以下是对该类函数的简单描述。

```
class LLMCorrector():
    def __init__(self,model=None,tokenizer=None):
        """
        初始化函数
        :param model:模型对象
        :param tokenizer:模型分词器对象
        :return: model 对象、tokenizer 对象、对话历史列表
        """
    def __TokenizerInitialize(self,model_params_path):
        """
        初始化模型必需的分词器
        :param model_params_path:模型文件路径，里面也包含了分词器
        :return: 一个重新部署的分词器对象
        """
    def __ModelInitialize(self,model_params_path,finetune_params_path):
        """
        初始化模型对象
        :param model_params_path:本地下载的模型参数地址
        :param finetune_params_path: 微调参数地址，需要精确到文件名
        :return: 重新部署到 GPU 的模型对象
        """
    def __get_prompt(self,text):
        """
        构造模型提示词模板
        :param text:待纠错的句子
        :return: 适合模型识别的提示词，跟历史 his 一起使用
        """
    def GEC_Correction(self,text):
        """
        接口，模型纠错
        :param text：输入模型的句子。
        :return:修改后返回的编辑序列列表。
        """
    def judge(self,response,text,max_unchanged_words=0,threshold=10):
```

```
"""
纠错结果检验，决定该次修改是否保留
:param response:模型返回结果
:param text:原句
:param max_unchanged_words:编辑序列中与原句最大重叠单词数
:return:bool 变量，表示改句修改是否修改过度，如果修改过度则丢弃
"""
```

剩下的类主要用于纠错功能，存储有错误类型、错误位置、错误长度及修改方法等信息。

5.4 关键算法设计

前后端登录算法的主要流程如图14所示。请求报文会携带用户名和密码信息，响应报文是LoginResponse类的对象，其中的token将会用作之后用户验证的凭证。

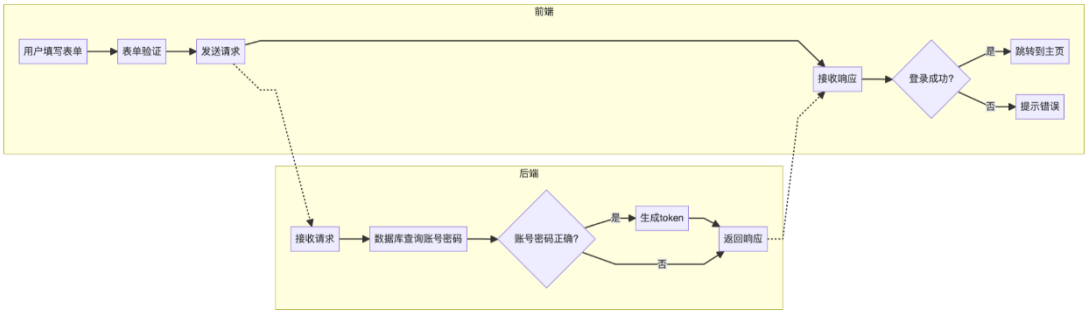


图14　登录算法图

文件请求流程如图15所示。文件请求报文包含用户名、token、文件名等信息。在后端会首先验证token，验证成功则会读取文件，读取成功后利用pandoc将文档解析为HTML文本，并将文本返回给前端用于后续纠错提示渲染。

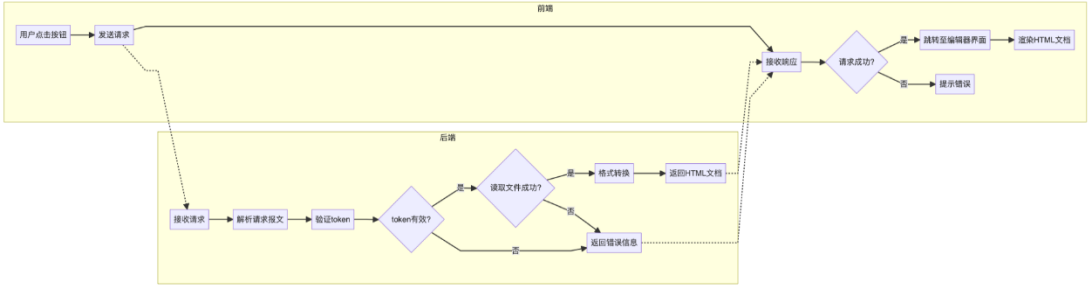


图15　文件请求流程图

文件纠错流程如图16所示。进入编辑器界面后，前端自动将文本分句并发送给后端。后端调用模型执行纠错，模型向后端返回纠正后的句子，后端再根据模型纠正结果和原句生成编辑序列返回给前端，前端根据编辑序列完成批注渲染，并显示文本修改选项框。

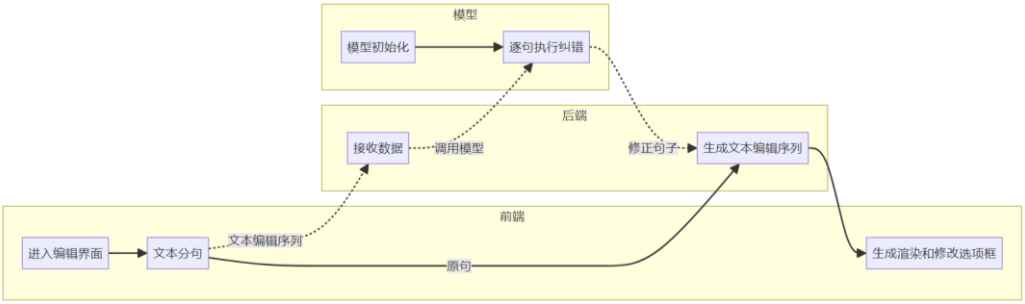


图16　文本纠错流程图

模型执行纠错功能的流程如图17所示。系统启动时，首先检查GPU是否充足，如果无足够显存则返回错误信息，系统启动失败。模型初始化后处于监听状态，接收待修改序列之后，将原句加入提示词模板并调用模型，针对返回结果与原句进行比对，如果相差过大则认为修改无效，返回原句；否则返回改句。

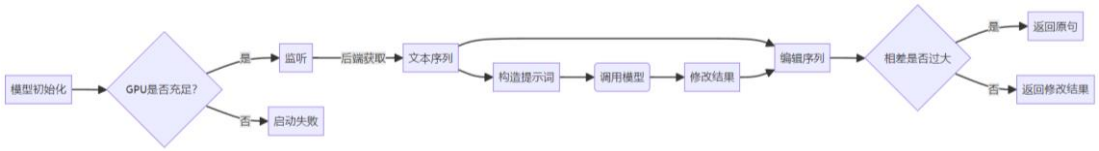


图17　模型执行纠错功能流程图

5.5 数据管理说明

5.5.1 用户数据

用户数据采用 MySQL 数据库管理系统管理。

当用户注册且检测注册信息无误后，即同意用户的注册请求，同时分配给用户一个唯一 id 作为主码，将用户的注册信息存入数据库。

5.5.2 用户登录状态

登录状态使用 Redis 数据库维护。Redis（Remote Dictionary Server）是一个使用 ANSI C 编写的开源、支持网络、基于内存、分布式的键值对存储数据库。具体而言，使用 Redis 的键值对存储并跟踪用户的登录状态。使用用户的唯一标识符（用户 ID）作为键，将登录状态作为值进行存储。

登录时，将用户的登录状态存储到 Redis 数据库中。使用一个属性来表示用户的登录状态，将用户 ID 作为键，采用布尔值表示是否已登录。在需要验证用户登录状态的地方，通过查询 Redis 数据库中存储的键值对来检查用户的登录状态。如果键存在且值为 true 表示用户已处于登录状态。当用户退出或会话超时，则更新或删除相应的键值对，将用户状态设置为已退出或过期，然后使用 del 命令删除键，或者设置新的布尔值（如 false）来更新登录状态。

5.5.3 用户文件存储

用户的文档采用文件系统存储，根据用户的 id，将不同用户的文件划分到不同的目录中。每当用户提交一份文件时，都会在用户对应 id 的目录下储存待纠错的文件。

6 开发工具与技术

我们在系统的实现过程中主要采用了以下技术。

6.1 数据集收集和构建

在前面，我们已经介绍了数据集的构造思路。我们将中文病句划分为两部分，分别是遵循模板的病句和不遵循模板的病句，并分别针对其特点设计自动生成流水线。

对于第一部分数据，我们通过从百度千帆大模型平台调用文心大模型 ERNIE 4.0 的 API 接口，**生成遵循模板的病句-改句对约 5000 条**。我们选取的病句模版如图 18 所示。

图 18 病句模板词云

一些基于病句模版构造的病句-改句数据集如下表所示。

病句模板	构造病句	构造改句
通过…使…	通过学习外语，使我能够更深入地了解不同文化。	学习外语使我能够更深入地了解不同文化。
随着…引起…	随着社交媒体的普及，人们越来越依赖于网络交流，引起了现实社交能力的下降。	随着社交媒体的流行，人们日渐依赖网络交流，导致现实社交能力逐渐减弱。
之所以…的原因是…	之所以我喜欢夏天，的原因是可以穿漂亮的裙子。	我喜欢夏天，因为可以穿漂亮的裙子。
经过…使…	经过无数次的尝试，使他终于破解了这个难题。	无数次的尝试使他终于破解了这个难题。
…超过…以上	这个项目的投资额度超过了两千万以上。	这个项目的投资额度超过了两千万。

防止…不再…	为了防止交通事故不再发生，交警加强了巡逻。	为了防止交通事故再次发生，交警加强了巡逻。
…	…	…

对于第二部分数据，我们首先利用大模型 ChatGPT-3.5-turbo 的 in-context learning 能力进行 few-shot text generation，指导模型自由生成大量病句，从中选取质量较高的部分；同时也从人工筛选的高质量数据集 NaCGEC 中挑选近 100 条具有歧义或者事实性错误的病句。通过使用开源近义词库 Synonym，将这两部分数据进行上下文无关的数据扩充。具体来说，针对每一条病句-改句对，利用 Synonym 进行命名实体识别（Named Entity Recognition, NER），对于相同的命名实体，统一替换为某个其他的命名实体。例如将句中不重要的人名、地名进行反复替换，这样可以帮助大模型学习到句子本身的结构，而非其余无关信息。除了人名、地名这类名词之外，我们还深入调研了 Synonym 词库，并尝试修改一些动词和形容词以进一步扩充数据。最终我们获得了**不遵循病句模板的病句-改句数据对共 5000 条**。

这部分数据的病句构造示例如下：

【例 1】原始数据集原句：2017 年《财富》全球论坛自宣布落户中国以来，《财富》杂志将目光聚焦在中国，以财富为媒，向世界展示一个充满底蕴又创新进取的中国。
原始数据集改句：自 2017 年《财富》全球论坛宣布落户中国以来，《财富》杂志将目光聚焦在中国，以财富为媒，向世界展示一个充满底蕴又创新进取的中国。
Synonym 构造后新的原句：2017 年《财富》全球论坛自宣布落户东京以来，《财富》杂志将目光聚焦在东京，以财富为媒，向世界展示一个充满底蕴又创新进取的东京。
Synonym 构造后新的改句：自 2017 年《财富》全球论坛宣布落户东京以来，《财富》杂志将目光聚焦在东京，以财富为媒，向世界展示一个充满底蕴又创新进取的东京。
【例 2】原始数据集原句：这次再版的长篇小说的作者都是很有名气的，如陈忠实的《白鹿原》，路遥的《平凡的世界》，莫言的《透明的红萝卜》。
原始数据集改句：这次再版的长篇小说都是很有名气的，如陈忠实的《白鹿原》，路遥的《平凡的世界》，莫言的《透明的红萝卜》
Synonym 构造后新的原句：这次再版的长篇小说的作者都是很有名气的，如古明華的《白鹿原》，卢科夫的《平凡的世界》，春兰的《透明的红萝卜》。
Synonym 构造后新的改句：这次再版的长篇小说都是很有名气的，如古明華的《白鹿原》，卢科夫的《平凡的世界》，春兰的《透明的红萝卜》。

最终，我们混合了以上两部分数据，**共得到约 10000 条病句-改句对作为最终的中文语病纠错数据集**。为了减轻模型微调的负担，我们从数据集中随机抽样，又得到了含 5000 条和 1000 条病句-改句对的训练集子集。

6.2 微调原理与参数

我们使用全参数微调的方法对原始大模型 Phoenix-inst-chat-7b 进行微调。该实验设计参考了[11]。

为了增强模型性能，我们使用 Huggingface 提供的 Transformers.trainer 训练框架对模型进行微调。Trainer 是在 Transformers 库中实现的一个完整的 PyTorch 模型训练和评估循环。只需要向它传递训练所需的必要组件（模型、分词器、数据集、评估函数、训练超参数等）就可以直接开始训练和评估。这使用户可以更轻松地开始训练，无需手动编写训练循环。与此同时，Trainer 有很大的定制空间，并提供了大量的训练选项，可根据训练需求进行精确调整。

Trainer 的主要作用是简化模型训练过程，它包含以的必要组件：

模型（model）：传递一个已构建好的 PyTorch 模型给 Trainer 用于训练和评估任务。

分词器（tokenizer）：Trainer 使用分词器将原始文本转换为模型可接受的输入格式。需要提供与模型相匹配的分词器。

数据集（dataset）：需要准备一个包含训练和验证数据的数据集。Trainer 将使用这些数据对模型进行训练和评估。

评估函数（evaluation function）：需要定义一个评估函数，用于在每个训练步骤结束时评估模型的性能。Trainer 将根据这些评估指标来选择最佳的模型。

训练超参数（training hyperparameters）：例如学习率、批量大小、训练周期数等，将影响模型的训练过程。

在实际训练过程中，我们使用如下训练选项：

优化器(Optimizer): 使用 AdamW 优化器，同时采用学习率下降策略来提升训练的精度。在训练的前 4 个轮次，学习率余弦增长；在训练后期，学习率线性下降。

损失函数(loss function): 采用 Transformer.trainer 的默认文本模型损失函数。其数学形式如下：

$$\text{Loss} = (1-\epsilon) \times \text{NLL}_{\text{loss}} + \epsilon \times \text{Smoothed}_{\text{loss}}$$

其中，NLL_{loss}和$\text{Smoothed}_{\text{loss}}$作为交叉熵损失的变体，其形式为：

$$\ell(x, y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n, & \text{if reduction} = \text{‘mean’;} \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{‘sum’.} \end{cases}$$

NLL_{loss}和$\text{Smoothed}_{\text{loss}}$分别为 l(x, y)的平均形式和求和形式，表示局部损失和全局损失。

整个全参数微调过程中，大模型的损失 Loss 随训练轮数 Epoch 的变化趋势如图 19 所示。

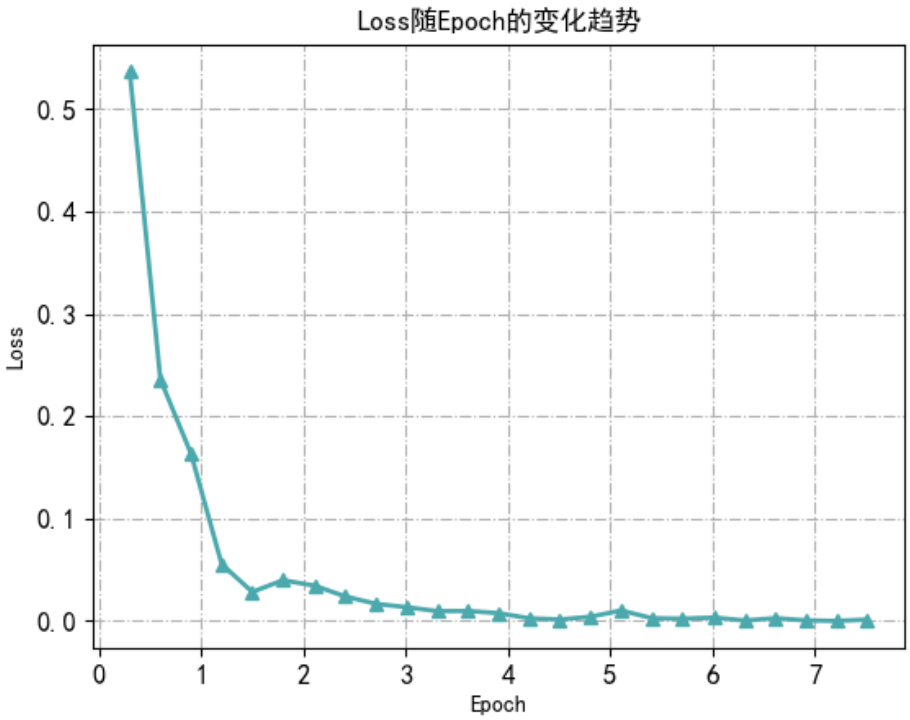


图 19　Loss 随 Epoch 的变化趋势

可以看出模型在 Epoch 为 2 附近已经收敛（全参微调所需显存约为 120G）。**这一点曾经在与项目对接的武汉通达信公司完成了实机验证。**

6.3 WEB 应用实现

为了支持前后端分离架构，本智能批改系统的前端使用 Vue 框架、node.js 运行环境和 TypeScript 语言进行开发。同时，为了便于模型的部署，后端使用 Flask 框架和 Python 语言开发。

7 系统实现

7.1 应用对象和环境

7.1.1 硬件设备

本项目计算工作主要在具有8块NVIDIA GeForce RTX 3090 GPU的运算服务器上完成，存储主要在一台磁盘空间9.1T的服务器上实现。

7.1.2 软件环境

本项目在Linux服务器上部署，依托python语言构建。使用包管理工具conda构建虚拟环境，结合服务器CUDA版本需求主要安装如下依赖。

torch == 2.0.1+cu12：项目的核心框架，主要用于支持模型架构定义、模型推理、以及数据处理和模型训练等工作。

transformers == 4.37.0：基于 pytorch 架构的大语言模型部署核心框架，为部署不同的大语言模型提供了用户友好的统一接口。

jieba & synonym：主要承担中文分词、命名实体识别和字符串批量处理等工作。

datasets：用于将病句-改句数据集处理为统一格式，便于模型的训练、微调和测试。

7.2 代码管理

本项目使用Git作为版本控制软件，使用Gitee平台进行在线代码托管，代码仓库地址为 https://gitee.com/GOODGAP/AutoThesisCorrection。代码版本管理的签入记录如图20所示。

图20　Gitee平台代码签入记录

整个仓库由五位团队成员共同贡献，项目仓库概览与仓库网络图分别如图21、图22所示，由于篇幅限制此处仅节选一部分。

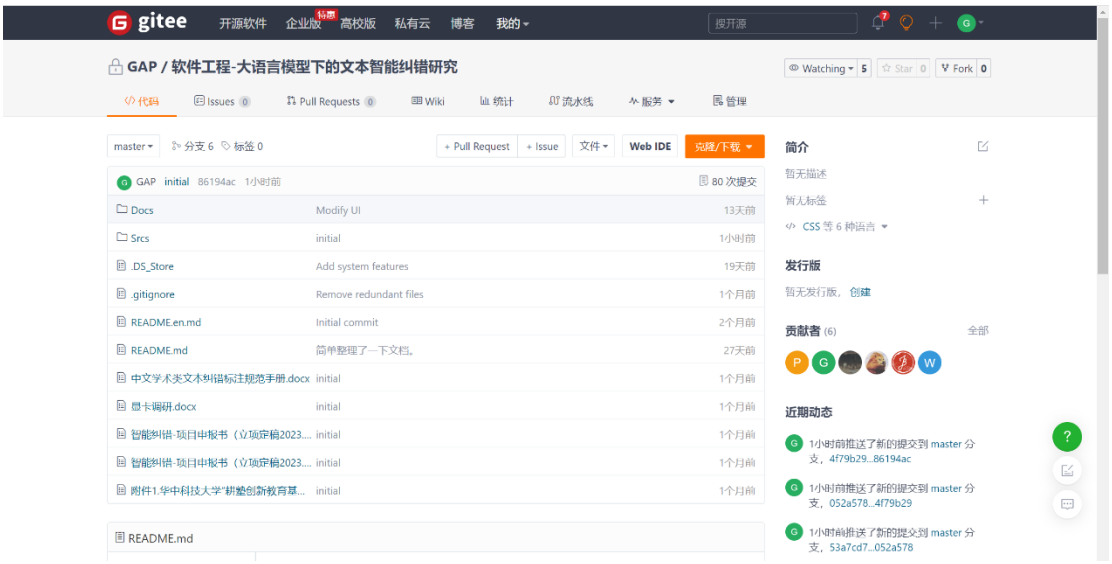


图21　项目仓库概览

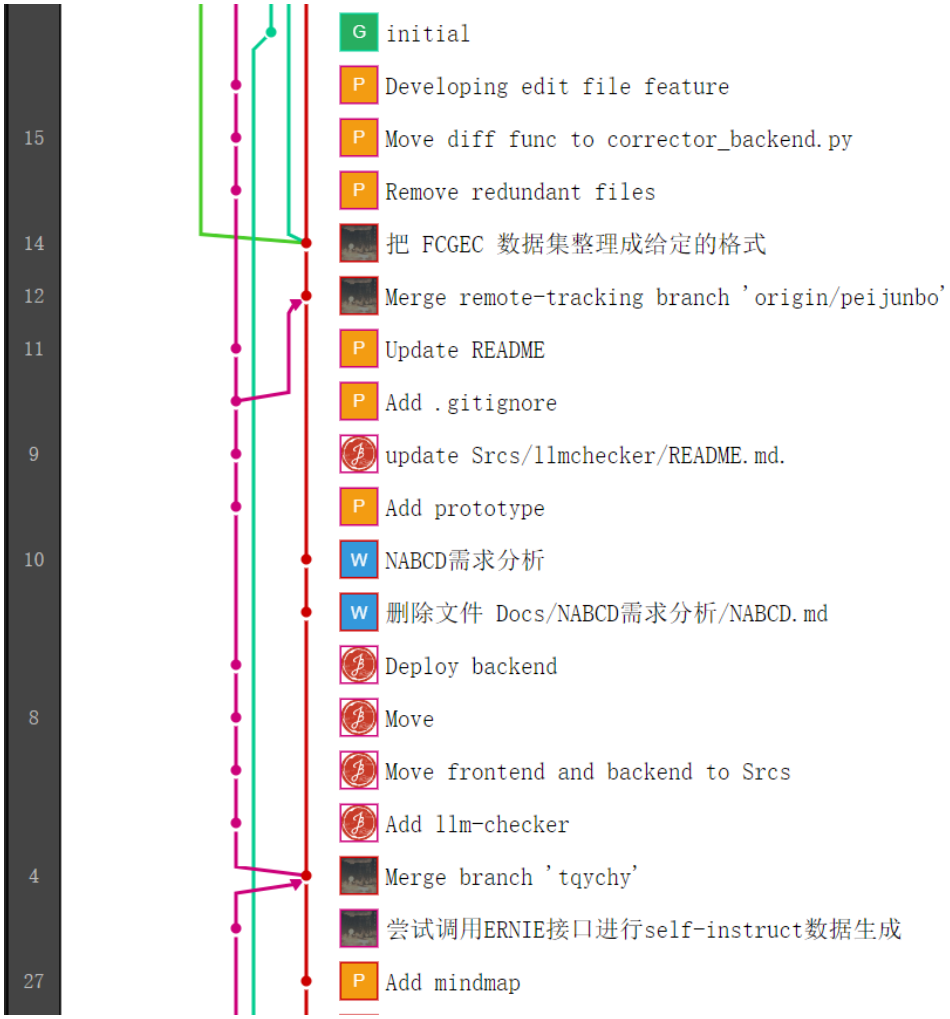


图22　仓库网络图（节选）

7.3 关键函数说明

7.3.1 M2 Scorer 评测模块关键函数说明

函数名称：edit_creator

函数功能：对两个文件(source 和 target)中的文本进行编辑距离计算和编辑操作的提取。首先对每一行文本进行预处理，并对其进行分词化（tokenize）。接下来使用 Levenshtein 算法计算距离矩阵和回溯矩阵，并合并两个编辑图，生成并输出该句的编辑序列。

函数名称：tokenizer

函数功能：将完整的句子按照语法成分拆解为多个独立的短语或词语，使用中文 jieba 分词器完成语法结构的拆分。

函数名称：m2scorer

函数功能：通过比对大模型输出的改正后的句子与原始病句 GOLD 标准之间的差异，计算在 M2 指标下的评分并输出。

整体的处理流程如图 23 所示。

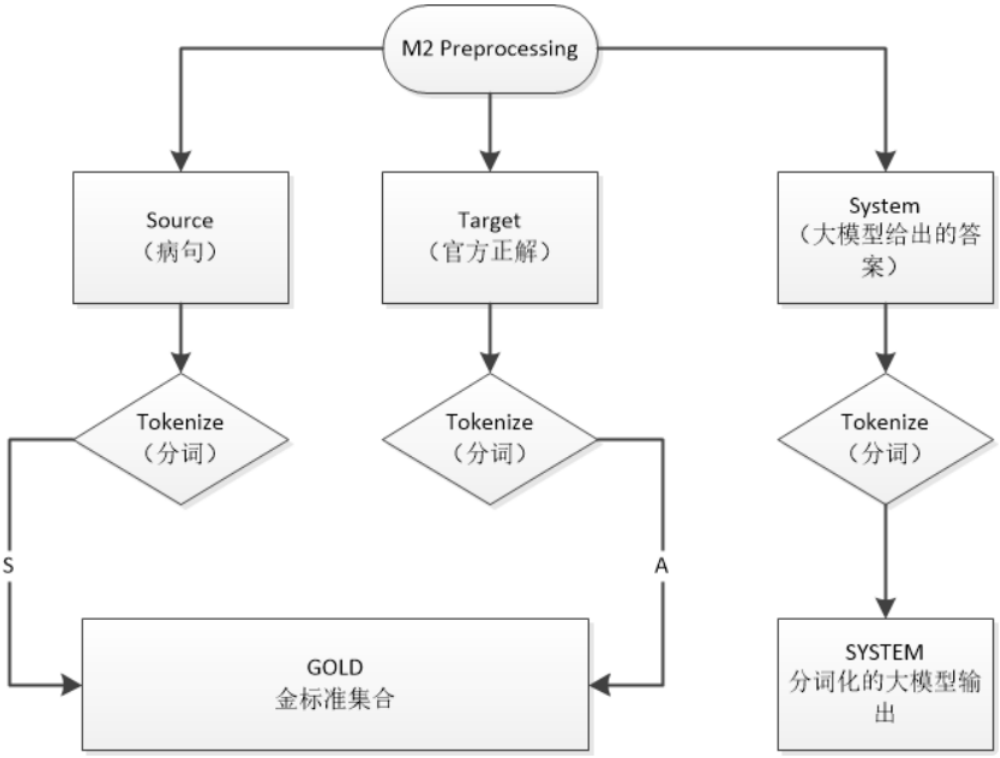


图23　M2 scorer模块处理流程

7.3.2 大语言模型关键函数说明

函数名称：LLMCorrector._TokenizerInitialize

函数功能：从参数路径中加载预训练模型的专用分词器。

函数名称：LLMCorrector._ModelInitialize

函数功能：首先检测系统环境中是否有空间充足的 GPU。如果没有则返回错误信息，否则从参数路径中加载预训练模型的参数文件。

函数名称：LLMCorrector._get_prompt

函数功能：根据模型接收的待修改文本，构建促使模型规范修改的提示词（prompt）。

函数名称：LLMCorrcector.judge

函数功能：判断是否保留修改结果。参考了 M2 scorer 的功能实现，将模型输出的序列与原始序列进行分词（tokenize），并从编辑序列构造编辑图，比较二者的交集大小。若交集占比过小则返回原句，否则返回改句。

函数名称：LLMCorrector.GEC_Correction

函数功能：首先构建提示词（prompt），之后传入 ChatGLM-6B（或 Phoenix-inst-chat-7b）模型，得到修改后的句子。最后调用 judge 函数，判断结果是否应该保留，并返回指定的输出内容。

7.3.3 前端关键对象说明

对象名称：src/router/index.ts router

对象作用：控制整个系统的页面跳转，同时为页面跳转进行保护，例如没有登录的用户不可访问个人信息界面。

对象名称：src/stores/useUserStore.ts

对象作用：存储全局的应用信息，同时管理用户登录状态等。

对象名称：src/views/EditorPages/Components/tiptap/TiptapEditor.vue editor

对象作用：编辑器实例，通过该实例对象可以在代码中进行编辑器行为的控制，包括修改内容、提取文本、插入图片、修改格式等。

7.3.4 后端关键函数说明

函数名称：user.get_all_users()

函数功能：从数据库中检索所有用户的信息。

路线：GET/users

函数名称：user.user_register()

函数功能：注册新用户。

路线：POST/register

输入：包含用户名、密码、电话、电子邮件和地址的 JSON 有效负载。

验证：检查数据库中是否已存在用户名；验证电话号码格式并检查它是否已注册；在存储密码之前对密码进行哈希处理。

函数名称：user.user_login()

函数功能：允许用户登录。

路线：POST/login

输入：带有用户名和密码的 JSON 有效负载。

验证：检查用户名是否存在并验证密码；为登录用户生成一个令牌（使用 MD5 哈希）并将其存储在 Redis 中。

函数名称：user.user_update(username)

函数功能：更新用户信息。

路线：PUT/update/user/int:id

输入：管理员用户、令牌、新密码、电话、电子邮件和地址。

验证：检查发出请求的用户是否为管理员；验证和更新用户信息（密码、电话、电子邮件、地址）。

函数名称：user.user_delete(username)

函数功能：删除用户。

路线：POST/delete/user/string:username

输入：管理员用户和令牌。

验证：检查发出请求的用户是否为管理员；从数据库中删除用户。

函数名称：predict.predict()

函数功能：将文本传入模型，调用模型纠错功能。

输入：前端用户输入文本。

函数名称：predict.differ()

函数功能：比较修改后文本和原文本区别，向前端返回错误位置和改正办法。

输入：改前和改后的句子。

返回：标注错误位置即建议更正办法。

7.4 测试计划和测试用例

对于本项目，我们主要采用黑盒测试和面向对象软件测试两种方法测试软件的性能与可靠性。

7.4.1 黑盒测试

黑盒测试是在不考虑内部结构和实现细节的情况下对软件进行测试。主要包括功能测试、用户体验测试和回归测试。

功能测试 为了测试本系统是否完成预期的功能，我们功能测试的步骤为：①输入各种文本，包括拼写错误、语法错误等；②检查软件是否能准确识别文本中的错误类型；③检查软件是否能给出正确的纠正建议，以及纠正后的文本是否合乎逻辑、语法正确。同时，还会测试软件对于特殊情况的处理，比如输入特定行业术语、缩写等，观察软件是否可以正常检错和纠错。

用户体验测试 我们将会收集同学们的使用意见，评估用户界面的友好程度，测试用户操作的直观性和便捷性。同样地，项目组成员也可以模拟用户使用过程，检查软件对于用户输入的响应是否快速，以及提供的建议是否准确、及时。

回归测试 在完成前两个测试阶段（功能测试和用户体验测试）后，我们需要修复之前发现的问题，并重新运行相同的测试用例，确保问题已被解决且没有引入新的错误。同时，在后续开发过程中，还需要定期对软件进行测试，以确保新功能的引入不会影响现有功能的稳定性。

综上所述，这些测试环节可以帮助发现智能纠错软件的各种功能和性能问题，提高软件的质量和用户体验。

在对 API 的测试中，我们使用了测试软件 Apifox。该软件可以像语雀、石墨文档一样创建团队文档，拥有相应权限的用户可在其中定义接口和数据模型等，接口定义支持添加 HTTP 请求方法、url 路径、请求体和返回格式等等信息。

7.4.2 面向对象软件测试

面向对象软件测试针对面向对象编程的软件系统进行测试，主要包括单元测试、集成测试、系统测试与安全性测试。对于本项目的“大语言模型下的智能纠错软件系统”，也可以采用面向对象的测试方法。

单元测试 主要针对各个类进行测试，确保类的方法和属性功能正确，这包括测试各个方法的输入输出、边界条件和异常情况处理。同时还需要确保测试用例覆盖了尽可能多的代码路径，包括语句覆盖、分支覆盖等。

模块集成测试 主要是将各个类和模块组合，测试它们之间的交互和集成情况。同时测试类之间的接口和数据传递，确保信息传递正确且无误，各环节都能得到确切的结果。

系统测试 模拟用户使用场景，测试系统的功能是否按照需求分析部分的要求正常工作，这包括测试系统在不同负载下的性能表现，包括响应时间、资源消耗、返回文本修改的准确性等。

安全和可靠性测试 检查系统的安全漏洞，确保用户数据和系统不受攻击。尤其是需要保护用户的隐私，比如在注册时填写的电话、邮箱、密码以及提交的文件信息等。同时，还需要测试系统在长时间运行和异常条件下的稳定性和可靠性。

7.5 系统整体效果分析

我们使用上述方法对我们的系统进行了测试，测试结果如下。

7.5.1 系统主页

* 在网络正常的情况下，系统主页可以正常访问。

* 主页无乱码或排版混乱等缺陷。

* 主页上的所有跳转链接均可正常跳转。

该部分测试结果如图 24 所示。

图 24　系统主页测试（正常显示）

7.5.2 用户登录

* 已经注册过的用户可以凭借正确的账号密码成功登录。

* 账号或密码错误时，系统会正常地给予提示，并要求重新输入。

* 非法用户无法通过 SQL 注入、弱密码攻击等常用入侵手段进入系统。

* 未注册的用户可以正常进行注册，当用户填写的注册信息非法时，系统会正常的给予提示。

该部分测试结果如图 25~图 27 所示。

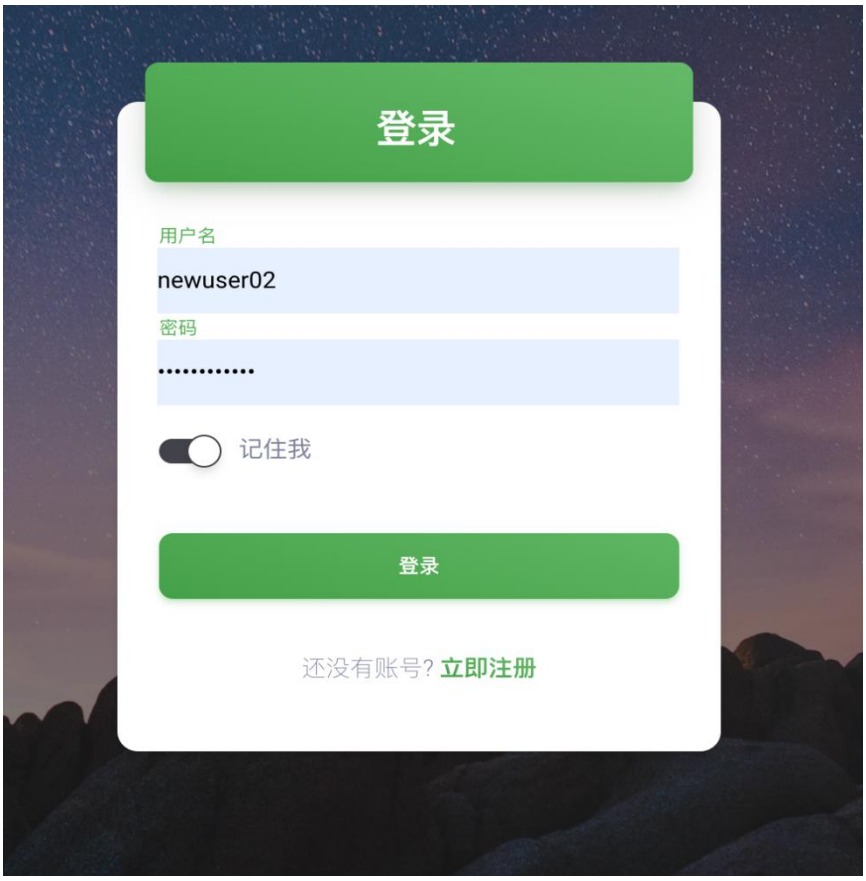


图 25　用户登录界面测试（正常显示）

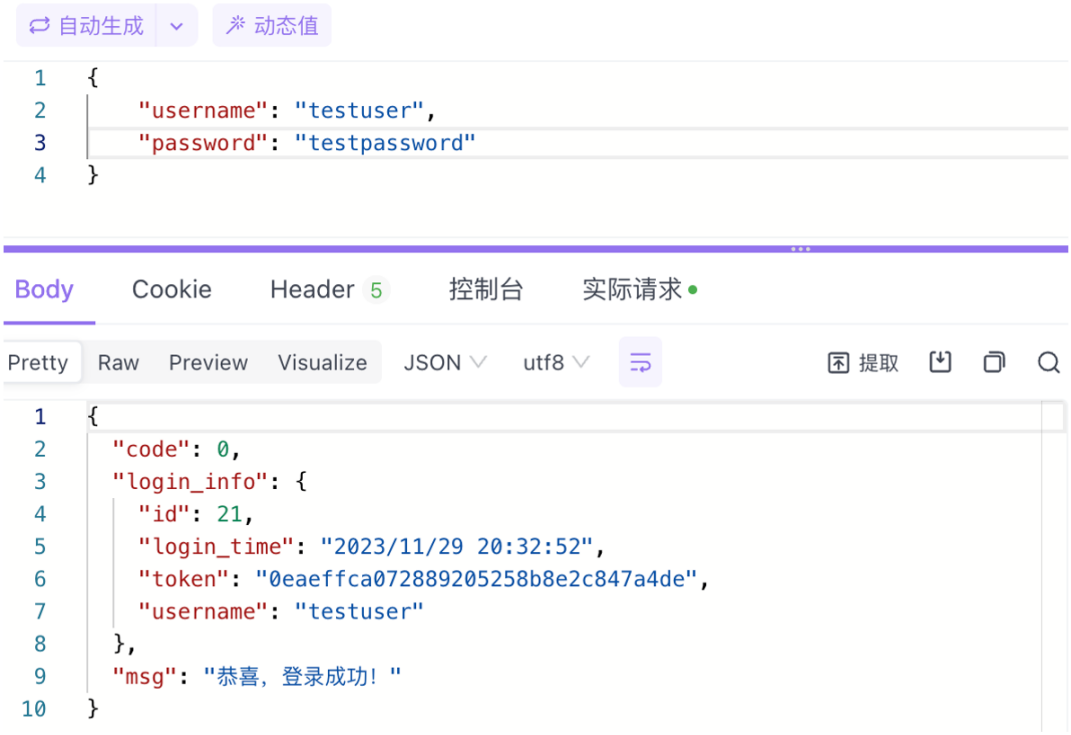


图 26　Apifox 测试正确登录

图 27　Apifox 测试错误登录

7.5.3 文件管理

* 已登录的用户可以正常进入文件上传页面。

* 登录的用户可以正常上传文件格式受支持的文件。

* 用户无法选择文件格式不受支持的文件。

* 文件上传出现错误时，系统会给予提示，并终止上传过程。

* 用户上传成功的文件可以正常打开，文件格式解析正常，不会出现乱码、格式错误等异常现象。

该部分测试结果如图 28、图 29 所示。

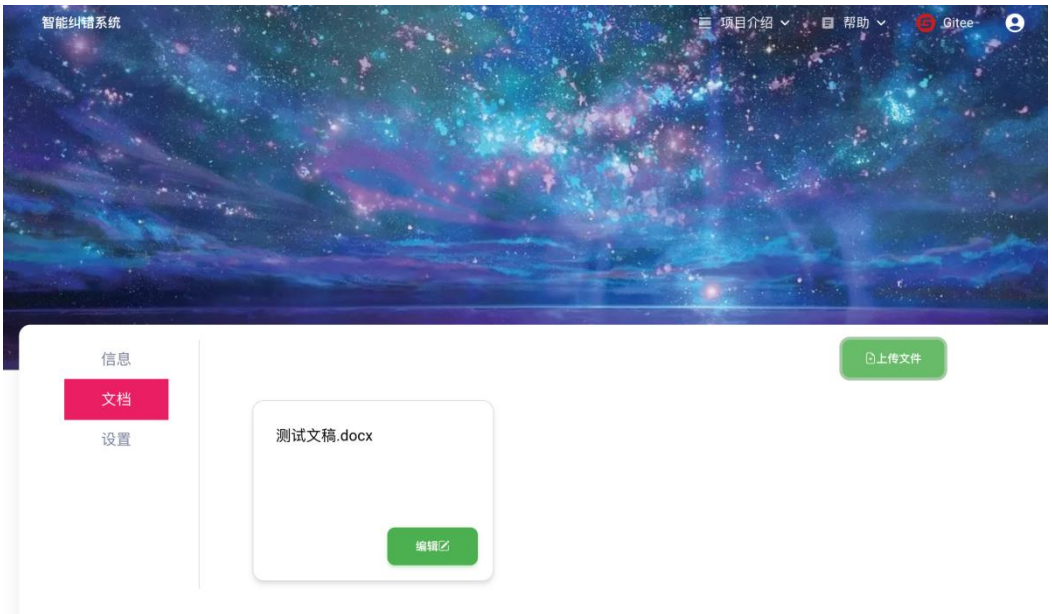


图 28　成功获取文件列表

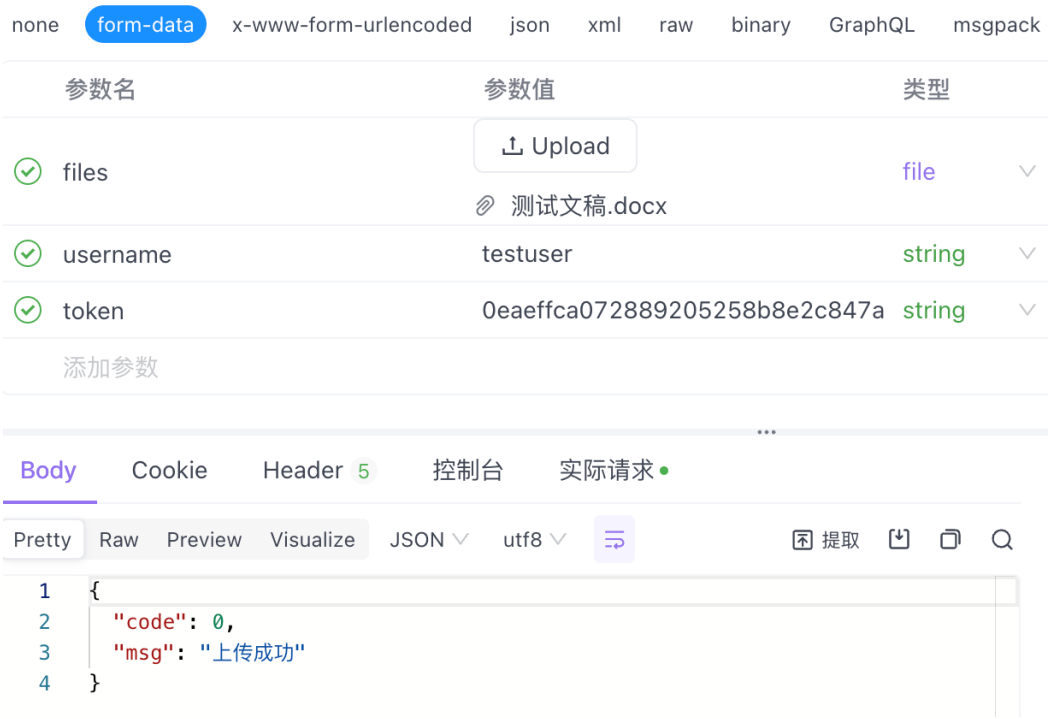


图 29　Apifox 测试文件上传成功

7.5.4 文本纠错

* 当用户打开上传的文件时，系统纠错功能运行正常。

* 系统的错误提示、修改按钮可以正常使用。

* 系统对各种拼写、语法、语义错误的检出率、改正率较高，且误检率较低。

该部分测试结果如图 30 所示。

图 30　纠错结果测试

7.5.5 在线编辑

* 在线编辑的编辑器、各种功能按钮等显示正常。

* 编辑器的功能按钮可以正常使用。

* 用户编辑后的修改记录可以正常保存至数据库，不会丢失。

综上所述，经过测试可知，本系统的显示内容和各种功能均正常，可以上线运行。

7.6 用户反馈

本项目系统开发完成后，我们将其介绍给一些同学进行试用，使用反馈与照片如下。

7.6.1 试用案例一

使用感受 某同学：“我觉得这个智能文本纠错系统对于我的 word 文档编辑有很大的帮助，可以很方便的帮我修正文本的拼写错误，语法错误，节省了我自己检查纠错的时间。同时，界面也设计的简洁明了，容易上手。美中不足的是有个别错误会检查不出来，可以继续改进算法和模型，进一步优化准确性。同时由于特定行业的术语可能与通用规则有所不同，考虑扩展支持更多的语种和专业领域，以满足不同用户的多样化需求。”

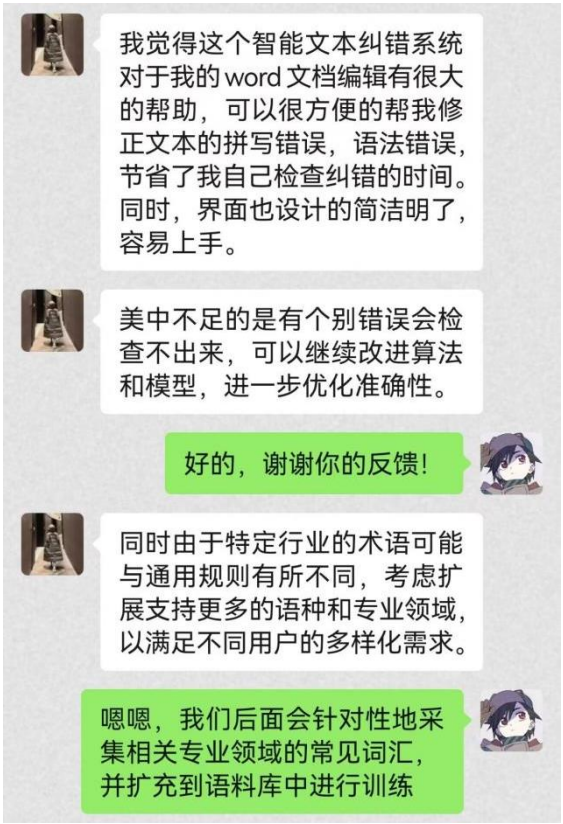


图 31　用户反馈一

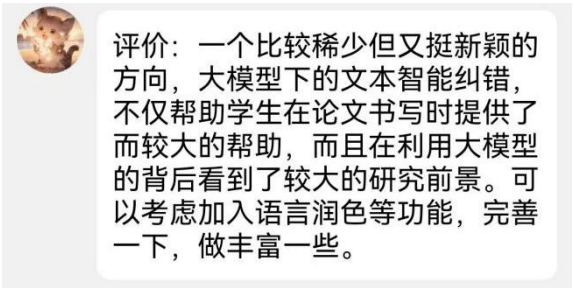


图 32 用户反馈二

7.6.2 试用案例二

使用感受 某同学：“这是一个比较稀少但又挺新颖的方向，大模型下的文本智能纠错，不仅帮助学生在论文书写时提供了较大的帮助，而且在利用大模型的背后看到了较大的应用前景。可以考虑加入语言润色等功能，完善一下，做丰富一些。”

8 结语

8.1 核心问题解决程度的定论

本项目研发了一种基于大语言模型（LLM）的中文论文批改修正虚拟助教系统，针对学术论文领域中常见的语法错误和拼写错误问题进行智能批改，提出专业的论文修改意见。我们基于相关理论[1]，采用近义词字典 synonym 和调用大模型 API 两种方式生成了高质量的教学场景下的语病数据集，填补了当前学术界该领域数据的空白；通过将 Phoenix-inst-chat-7b 模型作为基座模型并进行全参数微调，在多个公开测试集上达到接近 SOTA 的性能表现，智能检错、智能纠错的精度高达 60%以上，达到本领域领先水平。

我们还将微调好的大模型嵌入虚拟助教系统，并基于此在网页端设计了一个论文智能批改应用，基于大语言模型实现了学术论文的自动检错和纠错功能，还加入了师生注册与鉴别、一键改错、一键忽略、导出专业修改意见报告（PDF 文件）、上传下载文件、在线文本编辑等多种便捷功能。

综上所述，我们的项目不仅解决了教育领域中文语病修改数据方面空白的问题，同时还为高等教育领域论文批改慢、批改难的问题提供了“AIGC as Teaching Assistant”——即“以 AI 生成内容作为虚拟助教”的独特解决方案，在多个方面均展现了优秀的性能。

8.2 未来展望

本项目虽然已经有较高的完成度，但尚未大规模投入使用。结合用户反馈和团队成员的建议，我们将该中文论文智能批改虚拟助教系统在未来需要进一步完善的地方总结如下：

提高纠错准确度 我们将进一步提高虚拟助教系统在拼写错误和语法错误纠正方面的准确度，从而更好地适应学术论文中复杂的逻辑性要求。为了达到这一目标，我们接下来将会更深入调研相关数据构造模式和微调算法（如近期较火的 Qlora），并定期前往与项目对接的公司（武汉通达信公司）进行 A100 全参数微调。

引入专业术语知识库 我们计划扩展系统的领域范围，涵盖更多专业学科领域，争取让“虚拟助教”在高校的各个专业得到运用，减轻教师和学生的负担。无论是科学研究、工程技术还是人文社科等领域，我们都将采集更丰富的背景语料知识，增强大模型 AI 虚拟助教对于特定类型学术文本的理解能力。

提供更多辅助功能 我们逐渐计划为系统添加更多的辅助功能，如虚拟助教陪伴下的论文写作、虚拟助教的论文评估和论文风格指导、虚拟助教改写论文以及中文结构优化等。为此，我们将不断拓宽团队技术栈，以做代学，进一步学习并应用风格迁移等相关技术，**让虚拟助教更友好、更高效、更可靠，让 AI 赋能教育新篇章，实现论文智能批改，精准提升！**

参考文献

[1] Zeng, Aohan.etc (2022). GLM-130B: An Open Bilingual Pre-trained Model.

[2] Wang, Yizhong.etc (2022). Self-Instruct: Aligning Language Model with Self Generated Instructions. 10.48550/arXiv.2212.10560.

[3] Ma Shirong.etc (2022). Linguistic Rules-Based Corpus Generation for Native Chinese Grammatical Error Correction. arXiv.2210.10442.

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[5] Chollampatt, S., Ng, H. T., & Luo, Z. (2018). Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. arXiv preprint arXiv:1810.06799.3. Zhang, Y., & Sun, M. (2018). Interpretability for language learners. arXiv preprint arXiv:1810.03675.

[6] 王辰成, 陈嘉琪, 张瑞琦. 基于 Transformer 增强架构的中文语法纠错方法[J]. 计算机学报, 2020, 43(6): 106-118.

[7] Chollampatt, S., Ng, H. T., & Luo, Z. (2019). Adjusting the precision-recall trade-off with align-and-predict decoding for grammatical error correction. arXiv preprint arXiv:1901.09528.

[8] Chollampatt, S., Ng, H. T., & Luo, Z. (2019). Reusing a multi-lingual setup to bootstrap a grammar checker for a very low resource language without data. arXiv preprint arXiv:1901.09528.

[9] Liu, X., He, Y., & Sun, M. (2019). “Is Whole Word Masking Always Better for Chinese BERT?”: Probing on Chinese Grammatical Error Correction.

[10] Liang, C., Liang, X., Sun, M., & Huang, J.-T. (2019). MDCSpell: A Multi-task Detector-Corrector Framework for Chinese Spelling Correction

[11] Fan, Yaxin et al. “GrammarGPT: Exploring Open-Source LLMs for Native Chinese Grammatical Error Correction with Supervised Fine-Tuning.” *ArXiv* abs/2307.13923 (2023): n. pag.

[12] Fan, Y., Jiang, F., Li, P., & Li, H. (2023). GrammarGPT: Exploring Open-Source LLMs for Native Chinese Grammatical Error Correction with Supervised Fine-Tuning.

[13] Yang, H., & Quan, X. (n.d.). Alirector: Alignment-Enhanced Chinese Grammatical Error Corrector.

[14] Qu, Fanyi and Yunfang Wu. “Evaluating the Capability of Large-scale Language Models on Chinese Grammatical Error Correction Task.” ArXiv abs/2307.03972 (2023): n. pag.

附录

附录 A 关键代码

我们在此给出模型端、前端、后端和 M2 scorer 的部分关键代码。项目完整源代码请见【项目其他材料】。

模型端关键代码

```
    class LLMCorrector():
        def __init__(self,model=None,tokenizer=None):
            """
            :param model:模型对象
            :param tokenizer: 模型分词器对象
            """
            pynvml.nvmlInit()
            self.model=model if model!=None else
self.__ModelInitialize('/home/grammar/Paper_Corrector/backend_interact/mymodel','/home/gram
mar/Paper_Corrector/backend_interact/CGEC-chatglm2-6b-pt-128-2e-2/checkpoint-3000/pytorch_m
odel.bin')#如果没有指定模型，此函数重新部署模型，在这里修改参数。
            self.tokenizer=tokenizer if tokenizer!=None else
self.__TokenizerInitialize('/home/grammar/Paper_Corrector/backend_interact/mymodel')#如果没
有加载分词器，则重新加载分词器，路径在此处修改。
            self.his=[("因为污染也很高，我们要回收垃圾。为什么?塑料、电池、垃圾是我们做的东西。 ->
","因为污染很严重，所以我们要回收垃圾。为什么?因为塑料、电池等垃圾是我们人类造出来的东西。"),
                      ("最近吸烟者率是越来越多，而且吸烟者的年龄层是越来越少。这些问题是现在最重要
的事。 -> ","最近吸烟率是越来越高，而且吸烟者的年龄层是越来越低。这些问题是现在最重要的事。"),
                      ("不仅他学习某种习惯，他们性格也开始养成。 -> ","他们不仅开始学习某种习惯，
他们的性格也开始形成。"),
                      ("这世界上是很残酷的。 -> ","这世界是很残酷的。"),
                      ("每一刻都用得无可挑剔。 -> ","每一刻都用得无可挑剔。")]#人为构造的模型的历
史记录，不用更改

        def __TokenizerInitialize(self,model_params_path):
            """
            :param model_params_path:模型文件路径，里面也包含了分词器
            :return: 一个重新部署的分词器对象
            """
            # 载入 Tokenizer
            tokenizer = AutoTokenizer.from_pretrained(model_params_path,
trust_remote_code=True)
            return tokenizer

        def __ModelInitialize(self,model_params_path,finetune_params_path):
            """
            :param model_params_path:本地下载的模型参数地址
            :param finetune_params_path: 微调参数地址，需要精确到文件名
            :return: 重新部署到 GPU 的模型对象
            """
            # 检查是否有 GPU 以及 GPU 是否有足够内存加载模型。
            available_gpu_exists=False
            if torch.cuda.is_available():
                device_count=torch.cuda.device_count()
                for i in range(device_count):# 这里遍历所有的 GPU 找能用的
                    handle = pynvml.nvmlDeviceGetHandleByIndex(i)
                    meminfo = pynvml.nvmlDeviceGetMemoryInfo(handle)#这些都是模板
                    if(meminfo.free/1024**3>=16):    #chatglm-6b 占用显存 11-12G，这里找至少 13G
可用显存的 GPU，要是资源实在紧张可以把这个改成'>=12'.
                        print(f'Device {i} is available!')
```

```
                        device = torch.device(f"cuda:{i}")  # 获取当前 GPU 设备#这里用完之后记得改回 i
                        available_gpu_exists=True
                        break
                if available_gpu_exists==False:
                    raise MemoryError('No GPU available to load model.Please check whether the model is already loaded.')
            else:
                raise ModuleNotFoundError('No GPU found on this device,please check if you are running this program on GPU.')

            # 加载 chatglm2-6b 模型
            config = AutoConfig.from_pretrained(model_params_path, trust_remote_code=True, pre_seq_len=128)
            #config 是用来配置一部分模型参数的，不用改动。
            model = AutoModel.from_pretrained(model_params_path, trust_remote_code=True)
            #加载微调参数文件

            #-------------------------------------------------------------#
            prefix_state_dict = torch.load(
                finetune_params_path)#根据实际情况修改这一行为微调参数文件路径，要精确到.bin 文件
            new_prefix_state_dict = {}
            for k, v in prefix_state_dict.items():
                if k.startswith("transformer.prefix_encoder."):
                    new_prefix_state_dict[k[len("transformer.prefix_encoder."):]] = v
            model.transformer.prefix_encoder.load_state_dict(new_prefix_state_dict)
            #-------------------------------------------------------------#

            # 量化操作，减少模型的大小和计算量，不影响性能。
            model = model.quantize(4)  # 参数映射为 8 位整数，4bit 的量化精度，也可以设定为 8.
            model = model.half().cuda()
            model.transformer.prefix_encoder.float()
            # 设置为评估模式，推理速度更快。

            model = model.to(device)  # 将模型移动到当前 GPU 设备上
            model = model.eval()

            print('Successfully built model ChatGLM2-6B!')
            return model

        def __get_prompt(self,text):
            """
            :param text:待纠错的句子
            :return: 适合模型识别的提示词，跟历史 his 一起使用
            """
            prompt = """中文语法纠错任务：将一段中文句子进行修正，如果句子没有明显语病则返回原句。

            下面是一些范例：

            学生大概做飞机去北京。 -> 学生大概坐飞机去北京。\n
            我觉得他很大胆，他不去外国。他觉得如果要改转变中国，所以就要从中国做的研究。 -> 我觉得他很大胆，他不去外国。他觉得如果要改变中国，就要在中国做研究。\n
            吸烟者反对这样的措施。 -> 吸烟者反对这样的措施。\n

            请对下述句子进行修正。返回你修改后的句子，无需其它说明和解释。

            xxxxxx ->

            """
            return prompt.replace('xxxxxx',text)

        def GEC_Classification(self):
            """
            :return: 暂时还没写。
            """
            pass
```

```
        def GEC_Correction(self,text):
            """
            :param text: 输入模型的句子。
            :return:修改后返回的编辑序列列表。
            """
            prompt = self.__get_prompt(text)
            response, _ = self.model.chat(self.tokenizer, prompt, history=self.his)
            # edits=self.__Resp_to_Edit(response,text)
            response=self.judge(response,text,threshold=4)
            return response

        def judge(self,response,text,max_unchanged_words=0,threshold=10):
            """
            :param response
            :param text
            :param max_unchanged_words:
            """
            beta = 0.5
            ignore_whitespace_casing = False  # -  Ignore edits that only affect whitespace and
caseing. Default no.
            verbose = False  # -  print verbose output"
            very_verbose = False  # -  print lots of verbose output
            if(response==text):
                return response
            text = text.strip()
            response = response.strip()
            text_tok = ' '.join(jieba.cut(text, cut_all=False)).split()
            response_tok = ' '.join(jieba.cut(response, cut_all=False)).split()
            lmatrix1, backpointers1 = levenshtein_matrix(response_tok, text_tok, 1, 1, 1)
            lmatrix2, backpointers2 = levenshtein_matrix(response_tok, text_tok, 1, 1, 2)
            V1, E1, dist1, edits1 = edit_graph(lmatrix1, backpointers1)
            V2, E2, dist2, edits2 = edit_graph(lmatrix2, backpointers2)
            V, E, dist, edits = merge_graph(V1, V2, E1, E2, dist1, dist2, edits1, edits2)
            V, E, dist, edits = transitive_arcs(V, E, dist, edits, max_unchanged_words,
very_verbose)
            # Find the shortest path with an empty gold set
            gold = []
            localdist = levenshtein.set_weights(E, dist, edits, gold, verbose, very_verbose)
            editSeq = levenshtein.best_edit_seq_bf(V, E, localdist, edits, very_verbose)
            if ignore_whitespace_casing:
                editSeq = filter(lambda x: not equals_ignore_whitespace_casing(x[2], x[3]),
editSeq)
            if abs(editSeq[0][1] - editSeq[0][0]) >= threshold:
                return text
            return response
```

前端关键代码

```
# 前端主代码，负责启动前端 app
import { createApp } from "vue";
import { createPinia } from "pinia";
import App from "./App.vue";
import router from "./router";
// Nucleo Icons
import "./assets/css/nucleo-icons.css";
import "./assets/css/nucleo-svg.css";
import "bootstrap-icons/font/bootstrap-icons.css"
import "bootstrap/dist/css/bootstrap.css";
import materialKit from "./material-kit";
import "highlight.js/styles/github-dark.css";
import hljs from "highlight.js/lib/core";
import hljsVuePlugin from "@highlightjs/vue-plugin";
import bash from "highlight.js/lib/languages/bash";
import sql from "highlight.js/lib/languages/sql";
import shell from "highlight.js/lib/languages/shell";
hljs.registerLanguage("bash", bash);
hljs.registerLanguage("sql", sql);
hljs.registerLanguage("shell", shell);
const app = createApp(App);
app.use(createPinia());
app.use(router);
app.use(hljsVuePlugin);
app.use(materialKit);
app.mount("#app");

# 创建编辑器实例
const editor = useEditor({
  extensions: [
    StarterKit,
    Highlight,
    CodeBlock,
    TaskList,
    TaskItem.configure({
      nested: true
    }),
    LanguageTool.configure({
      automaticMode: true,
      documentId: '1',
      apiUrl: 'http://222.20.97.104:8081/correct/',
    }),
    Table as any,
    TableCell,
    TableHeader,
    TableRow,
    Image.configure({
      inline: true,
      allowBase64: true,
    })
  ],
  onUpdate({ editor }) {
    setTimeout(() => updateMatch(editor as any))
  },
  onSelectionUpdate({ editor }) {
    setTimeout(() => updateMatch(editor as any))
  },
  onTransaction({ transaction: tr }) {
    if (tr.getMeta(LanguageToolHelpingWords.LoadingTransactionName)) {
      console.log('loading')
      loading.value = true
    }
    else {
      console.log('not loading')
      loading.value = false
```

```
    }
  },
})

创建用户信息存储实例
export const useUserStore = defineStore("userDetail", () => {
    const mUserDetail = ref<UserDetail>(getSessionUserDetail() ?? emptyUserDetail());
    const userDetail = computed<UserDetail>({
        get() {
            return mUserDetail.value;
        },
        set(newUserDetail: UserDetail) {
            console.log("set userDetail")
            setSessionUserDetail(newUserDetail);
            mUserDetail.value = newUserDetail
        }
    })
    const isLogin = computed<boolean>(() => {
        return userDetail.value.token !== undefined && userDetail.value.token.length > 0
    })
    return {
        userDetail,
        isLogin,
    };
});
```

后端关键代码

```
    def diff(a, b):
        """
        比较两个字符串的差异，输出需要变换的操作序列
        :param a: 字符串 a     :param b: 字符串 b
        """
        m, n = len(a), len(b)
        dp = [[0] * (n + 1) for _ in range(m + 1)]
        for i in range(m + 1):
            dp[i][0] = i
        for j in range(n + 1):
            dp[0][j] = j
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if a[i - 1] == b[j - 1]:
                    dp[i][j] = dp[i - 1][j - 1]
                else:
                    dp[i][j] = 1 + min(dp[i][j - 1], dp[i - 1][j], dp[i - 1][j - 1])
        # 记录操作序列
        res = []
        """
        Operation:
        0, not change    1. add     2. delete     3. modify
        """
        i, j = m, n
        while i != 0 or j != 0:
            if i > 0 and j > 0 and a[i - 1] == b[j - 1]:
                res.append([0, ""])
                i -= 1
                j -= 1
            else:
                if i > 0 and dp[i][j] == dp[i - 1][j] + 1:
                    res.append([2, a[i - 1]])
                    i -= 1
                elif j > 0 and dp[i][j] == dp[i][j - 1] + 1:
                    res.append([1, b[j - 1]])
                    j -= 1
                else:
                    res.append([3, b[j - 1]])
                    i -= 1
                    j -= 1
        res.reverse()
        return res


    if __name__ == "__main__":
        a = "I ad a boy"
        b = "I dont am a boy"
    print(diff(a, b))

    #coding utf-8
    parser = argparse.ArgumentParser()
    parser.add_argument('-m', '--model_path',default=r'Simple Flask
Backend\Models\exps\seq2seq_lang8+hsk')
    parser.add_argument('-i', '--input_path',default=r'Simple Flask Backend\Models\test.txt')
    parser.add_argument('-o', '--output_path',default=r'Simple Flask Backend\Models\res.txt')
    parser.add_argument('-b', '--batch_size', default=50)
    args = parser.parse_args()
    cc = OpenCC("t2s")
    tokenizer=BertTokenizer.from_pretrained(args.model_path)
    model=BartForConditionalGeneration.from_pretrained(args.model_path)
    model.eval()
    model.half()
    model.cuda()

    def split_sentence(document: str, flag: str = "all", limit: int = 510):
        """
```

```
        Args:
            document:
            flag: Type:str, "all" 中英文标点分句, "zh" 中文标点分句, "en" 英文标点分句
            limit: 默认单句最大长度为 510 个字符
        Returns: Type:list
        """
        sent_list = []
        try:
            if flag == "zh":
                document = re.sub('(?P<quotation_mark>([。？！](?![”’"\'])))',
r'\g<quotation_mark>\n', document)  # 单字符断句符
                document = re.sub('(?P<quotation_mark>([。？！])[”’"\'])',
r'\g<quotation_mark>\n', document)  # 特殊引号

            elif flag == "en":
                document = re.sub('(?P<quotation_mark>([.?!](?![”’"\'])))',
r'\g<quotation_mark>\n', document)  # 英文单字符断句符
                document = re.sub('(?P<quotation_mark>([?!.]["\']))', r'\g<quotation_mark>\n',
document)  # 特殊引号

            else:
                document = re.sub('(?P<quotation_mark>([。？！….?!](?![”’"\'])))',
r'\g<quotation_mark>\n', document)  # 单字符断句符
                document = re.sub('(?P<quotation_mark>(([。？！.!?]|…{1,2})[”’"\']))',
r'\g<quotation_mark>\n', document)  # 特殊引号

            sent_list_ori = document.splitlines()
            for sent in sent_list_ori:
                sent = sent.strip()
                if not sent:
                    continue
                else:
                    while len(sent) > limit:
                        temp = sent[0:limit]
                        sent_list.append(temp)
                        sent = sent[limit:]
                    sent_list.append(sent)
        except:
            sent_list.clear()
            sent_list.append(document)
        return sent_list

    def run_model(sents):
        num_ret_seqs = 1
        beam = 5
        inp_max_len = 100
        batch = [tokenizer(s, return_tensors='pt', padding='max_length',
max_length=inp_max_len) for s in sents]
        oidx2bidx = {} #original index to final batch index
        final_batch = []
        for oidx, elm in enumerate(batch):
            if elm['input_ids'].size(1) <= inp_max_len:
                oidx2bidx[oidx] = len(final_batch)
                final_batch.append(elm)
        batch = {key: torch.cat([elm[key] for elm in final_batch], dim=0) for key in
final_batch[0]}
        with torch.no_grad():
            generated_ids = model.generate(batch['input_ids'].cuda(),
                                           attention_mask=batch['attention_mask'].cuda(),
                                           num_beams=beam, num_return_sequences=num_ret_seqs,
max_length=inp_max_len)
        _out = tokenizer.batch_decode(generated_ids.detach().cpu(), skip_special_tokens=True)
        outs = []
        for i in range(0, len(_out), num_ret_seqs):
            outs.append(_out[i:i+num_ret_seqs])
        final_outs = [[sents[oidx]] if oidx not in oidx2bidx else outs[oidx2bidx[oidx]] for oidx
in range(len(sents))]
```

```
    return final_outs

def predict():
    sents = [l.strip() for l in open(args.input_path,encoding='utf-8'
)]  # 分句
    subsents = []
    s_map = []
    for i, sent in enumerate(sents):  # 将篇章划分为子句，分句预测再合并
        subsent_list = split_sentence(sent, "zh")
        s_map.extend([i for _ in range(len(subsent_list))])
        subsents.extend(subsent_list)
    assert len(subsents) == len(s_map)
    b_size = args.batch_size
    outs = []
    for j in tqdm(range(0, len(subsents), b_size)):
        sents_batch = subsents[j:j+b_size]
        outs_batch = run_model(sents_batch)
        for sent, preds in zip(sents_batch, outs_batch):
            outs.append({'src': sent, 'preds': preds})
    results = ["" for _ in range(len(sents))]
    with open(args.output_path, 'w') as outf:
        for i, out in enumerate(outs):
            results[s_map[i]] += cc.convert(out['preds'][0])
        for res in results:
            outf.write(res + "\n")

predict()
```

M2 Scorer 关键代码

```
# m2scorer 代码
import re
import sys

class DummyTokenizer(object):

    def tokenize(self, text):
        return text.split()

class PTBTokenizer(object):

    def __init__(self, language="en"):
        self.language = language
        self.nonbreaking_prefixes = {}
        self.nonbreaking_prefixes_numeric = {}
        self.nonbreaking_prefixes["en"] = ''' A B C D E F G H I J K L M N O P Q R S T U V
W X Y Z
            Adj Adm Adv Asst Bart Bldg Brig Bros Capt Cmdr Col Comdr Con Corp Cpl DR Dr Drs
Ens
            Gen Gov Hon Hr Hosp Insp Lt MM MR MRS MS Maj Messrs Mlle Mme Mr Mrs Ms Msgr Op
Ord
            Pfc Ph Prof Pvt Rep Reps Res Rev Rt Sen Sens Sfc Sgt Sr St Supt Surg
            v vs i.e rev e.g Nos Nr'''.split()
        self.nonbreaking_prefixes_numeric["en"] = '''No Art pp'''.split()
        self.special_chars = re.compile(r"([^\w\s\.\'\`\,\-\"\|\/])", flags=re.UNICODE)

    def tokenize(self, text, ptb=False):
        text = text.strip()
        text = " " + text + " "

        text = re.sub(self.special_chars, r' \1 ', text)
        text = re.sub(r";", r' ; ', text)
        text = re.sub(r":", r' : ', text)
        text = re.sub(r"\|", r' -PIPE- ', text)
        text = re.sub(r"(\S)/(\S)", r'\1 / \2', text)

        # 分词化
        if ptb:
            text = re.sub(r"\(", r' -LRB- ', text)
            text = re.sub(r"\)", r' -RRB- ', text)
            text = re.sub(r"\[", r' -LSB- ', text)
            text = re.sub(r"\]", r' -RSB- ', text)
            text = re.sub(r"\{", r' -LCB- ', text)
            text = re.sub(r"\}", r' -RCB- ', text)

            text = re.sub(r"\"\s*$", r" '' ", text)
            text = re.sub(r"^\s*\"", r' `` ', text)
            text = re.sub(r"(\S)\"\s", r"\1 '' ", text)
            text = re.sub(r"\s\"(\S)", r" `` \1", text)
            text = re.sub(r"(\S)\"", r"\1 '' ", text)
            text = re.sub(r"\"(\S)", r" `` \1", text)
            text = re.sub(r"'\s*$", r" ' ", text)
            text = re.sub(r"^\s*'", r" ` ", text)
            text = re.sub(r"(\S)'\s", r"\1 ' ", text)
            text = re.sub(r"\s'(\S)", r" ` \1", text)

            text = re.sub(r"'ll", r" -CONTRACT-ll", text)
            text = re.sub(r"'re", r" -CONTRACT-re", text)
            text = re.sub(r"'ve", r" -CONTRACT-ve", text)
            text = re.sub(r"n't", r" n-CONTRACT-t", text)
            text = re.sub(r"'LL", r" -CONTRACT-LL", text)
            text = re.sub(r"'RE", r" -CONTRACT-RE", text)
            text = re.sub(r"'VE", r" -CONTRACT-VE", text)
            text = re.sub(r"N'T", r" N-CONTRACT-T", text)
            text = re.sub(r"cannot", r"can not", text)
            text = re.sub(r"Cannot", r"Can not", text)
```

```
        text = re.sub(r"\.([\.]+)", r" DOTMULTI\1", text)
        while re.search("DOTMULTI\.", text):
            text = re.sub(r"DOTMULTI\.([^\.])", r"DOTDOTMULTI \1", text)
            text = re.sub(r"DOTMULTI\.", r"DOTDOTMULTI", text)

        text = re.sub(r"\-([\-]+)", r" DASHMULTI\1", text)
        while re.search("DASHMULTI\-", text):
            text = re.sub(r"DASHMULTI\-([^\-])", r"DASHDASHMULTI \1", text)
            text = re.sub(r"DASHMULTI\-", r"DASHDASHMULTI", text)

        text = re.sub(r"(\D),(\D)", r'\1 , \2', text)
        text = re.sub(r"(\d),(\D)", r'\1 , \2', text)
        text = re.sub(r"(\D),(\d)", r'\1 , \2', text)

        if self.language == "en":
            text = re.sub(r"([^a-zA-Z])'([^a-zA-Z])", r"\1 ' \2", text)
            text = re.sub(r"(\W)'([a-zA-Z])", r"\1 ' \2", text)
            text = re.sub(r"([a-zA-Z])'([^a-zA-Z])", r"\1 ' \2", text)
            text = re.sub(r"([a-zA-Z])'([a-zA-Z])", r"\1 '\2", text)
            text = re.sub(r"(\d)'(s)", r"\1 '\2", text)
            text = re.sub(r" '\s+s ", r" 's ", text)
            text = re.sub(r" '\s+s ", r" 's ", text)
        else:
            text = re.sub(r"'", r" ' ")
            text = re.sub(r"' '", r"''", text)
        words = text.split()
        text = ''
        for i, word in enumerate(words):
            m = re.match("^(\S+)\.$", word)
            if m:
                pre = m.group(1)
                if ((re.search("\.", pre) and re.search("[a-zA-Z]", pre)) or \
                    (pre in self.nonbreaking_prefixes[self.language]) or \
                    ((i < len(words)-1) and re.match("^\d+", words[i+1]))):
                    pass  # do nothing
                elif ((pre in self.nonbreaking_prefixes_numeric[self.language] ) and \
                      (i < len(words)-1) and re.match("\d+", words[i+1])):
                    pass  # do nothing
                else:
                    word = pre + " ."

            text += word + " "
        text = re.sub(r"'\s+'", r"''", text)

        while re.search("DOTDOTMULTI", text):
            text = re.sub(r"DOTDOTMULTI", r"DOTMULTI.", text)
        text = re.sub(r"DOTMULTI", r".", text)

        while re.search("DASHDASHMULTI", text):
            text = re.sub(r"DASHDASHMULTI", r"DASHMULTI-", text)
        text = re.sub(r"DASHMULTI", r"-", text)
        text = re.sub(r"-CONTRACT-", r"'", text)

        return text.split()


    def tokenize_all(self,sentences, ptb=False):
        return [self.tokenize(t, ptb) for t in sentences]

if __name__ == "__main__":
    tokenizer = PTBTokenizer()
    for line in sys.stdin:
        line = line.decode("utf8")
        tokens = tokenizer.tokenize(line.strip())
        out = ' '.join(tokens)
        print(out.encode("utf8"))
```

```
# levenshtein 矩阵图算法代码
def levenshtein_distance(first, second):
    lmatrix, backpointers = levenshtein_matrix(first, second)
    return lmatrix[-1][-1]


# levenshtein matrix
def levenshtein_matrix(first, second, cost_ins=1, cost_del=1, cost_sub=2):
    #if len(second) == 0 or len(second) == 0:
    #    return len(first) + len(second)
    first_length = len(first) + 1
    second_length = len(second) + 1

    distance_matrix = [[None] * second_length for x in range(first_length)]
    backpointers = {}
    distance_matrix[0][0] = 0
    for i in range(1, first_length):
        distance_matrix[i][0] = i
        edit = ("del", i-1, i, first[i-1], '', 0)
        backpointers[(i, 0)] = [((i-1,0), edit)]
    for j in range(1, second_length):
        distance_matrix[0][j]=j
        edit = ("ins", 0, 0, '', second[j-1], 0) # always insert from the beginning
        #edit = ("ins", j-1, j-1, '', second[j-1], 0)
        backpointers[(0, j)] = [((0,j-1), edit)]

    for i in range(1, first_length):
        for j in range(1, second_length):
            deletion = distance_matrix[i-1][j] + cost_del
            insertion = distance_matrix[i][j-1] + cost_ins
            if first[i-1] == second[j-1]:
                substitution = distance_matrix[i-1][j-1]
            else:
                substitution = distance_matrix[i-1][j-1] + cost_sub
            if substitution == min(substitution, deletion, insertion):
                distance_matrix[i][j] = substitution
                if first[i-1] != second[j-1]:
                    edit = ("sub", i-1, i, first[i-1], second[j-1], 0)
                else:
                    edit = ("noop", i-1, i, first[i-1], second[j-1], 1)
                try:
                    backpointers[(i, j)].append(((i-1,j-1), edit))
                except KeyError:
                    backpointers[(i, j)] = [((i-1,j-1), edit)]
            if deletion == min(substitution, deletion, insertion):
                distance_matrix[i][j] = deletion
```

```
                edit = ("del", i-1, i, first[i-1], '', 0)
                try:
                    backpointers[(i, j)].append(((i-1,j), edit))
                except KeyError:
                    backpointers[(i, j)] = [((i-1,j), edit)]
            if insertion == min(substitution, deletion, insertion):
                distance_matrix[i][j] = insertion
                edit = ("ins", i, i, '', second[j-1], 0)
                try:
                    backpointers[(i, j)].append(((i,j-1), edit))
                except KeyError:
                    backpointers[(i, j)] = [((i,j-1), edit)]
    return (distance_matrix, backpointers)


# edit_creator 编辑序列创建算法
for candidate, source in zip(system_read, source_read):
    if not count % 1000:
        print >> sys.stderr, count,
    count += 1
    candidate = candidate.strip()
    source = source.strip()

    candidate_tok = candidate.split()
    source_tok = source.split()
    #lmatrix, backpointers = levenshtein_matrix(source_tok, candidate_tok)
    lmatrix1, backpointers1 = levenshtein_matrix(source_tok, candidate_tok, 1, 1, 1)
    lmatrix2, backpointers2 = levenshtein_matrix(source_tok, candidate_tok, 1, 1, 2)

    #V, E, dist, edits = edit_graph(lmatrix, backpointers)
    V1, E1, dist1, edits1 = edit_graph(lmatrix1, backpointers1)
    V2, E2, dist2, edits2 = edit_graph(lmatrix2, backpointers2)

    V, E, dist, edits = merge_graph(V1, V2, E1, E2, dist1, dist2, edits1, edits2)
    V, E, dist, edits = transitive_arcs(V, E, dist, edits, max_unchanged_words, very_verbose)

    # S = source, T = target
    print >> write_output, "S {0}".format(source)
    if verbose:
      print >> write_output, "T {0}".format(candidate)

    # 最短路径搜索
    gold = []
    localdist = levenshtein.set_weights(E, dist, edits, gold, verbose, very_verbose)
    editSeq = levenshtein.best_edit_seq_bf(V, E, localdist, edits, very_verbose)
    if ignore_whitespace_casing:
```

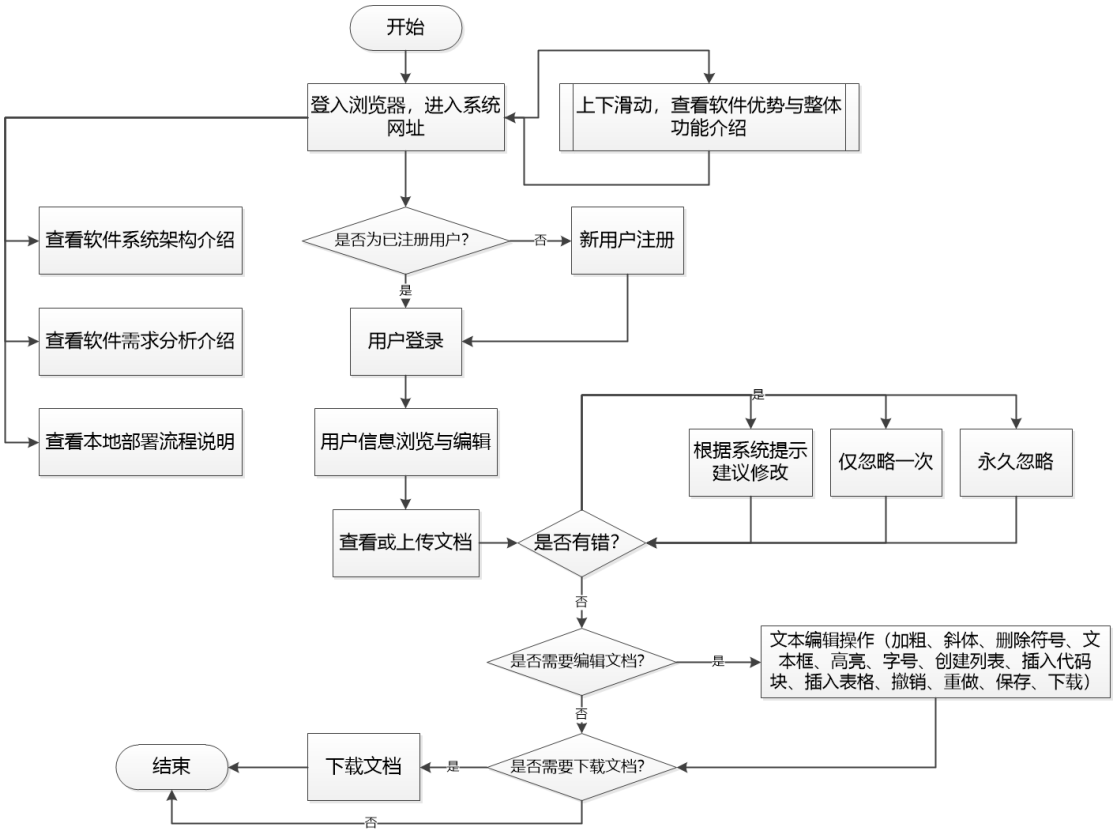
```
            editSeq = filter(lambda x : not equals_ignore_whitespace_casing(x[2], x[3]),
editSeq)

        for ed in list(reversed(editSeq)):
            if ed[2] != ed[3]:
                print >> write_output, "A {0} {1}|||{2}|||{3}|||{4}|||{5}|||{6}".format(ed[0],
ed[1], "UNK", ed[3], 'REQUIRED', '-NONE-', 0)
        print >> write_output,""
    system_read.close()
    source_read.close()
    print >> sys.stderr, "Done!"
```

附录 B 用户使用说明书与知识产权相关信息

一、软件使用流程图

下图展示了本软件的使用全流程。



二、软件运行系统要求

本软件要求用户设备具有 Android 5.0 以上、iOS 10.0 以上或 Windows 7 以上操作系统，内存 4GB 以上。

三、软件使用步骤详解

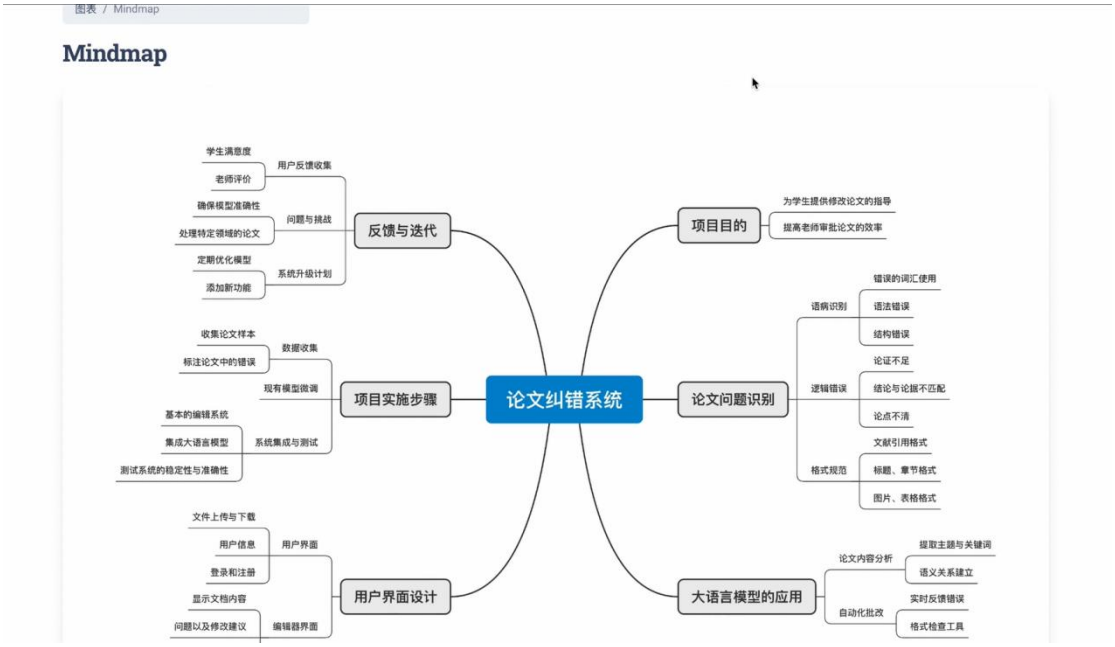
（1）进入软件首页。

（2）在首页中上下滑动，可以查看软件优势与整体功能介绍。

用户可借此快速上手本软件的智能纠错功能，了解软件特色及支持的文本格式。

（3）依次点击上方的 项目介绍 项目图表 项目相关图表 系统架构图 按钮，进入软件系统架构介绍页面。

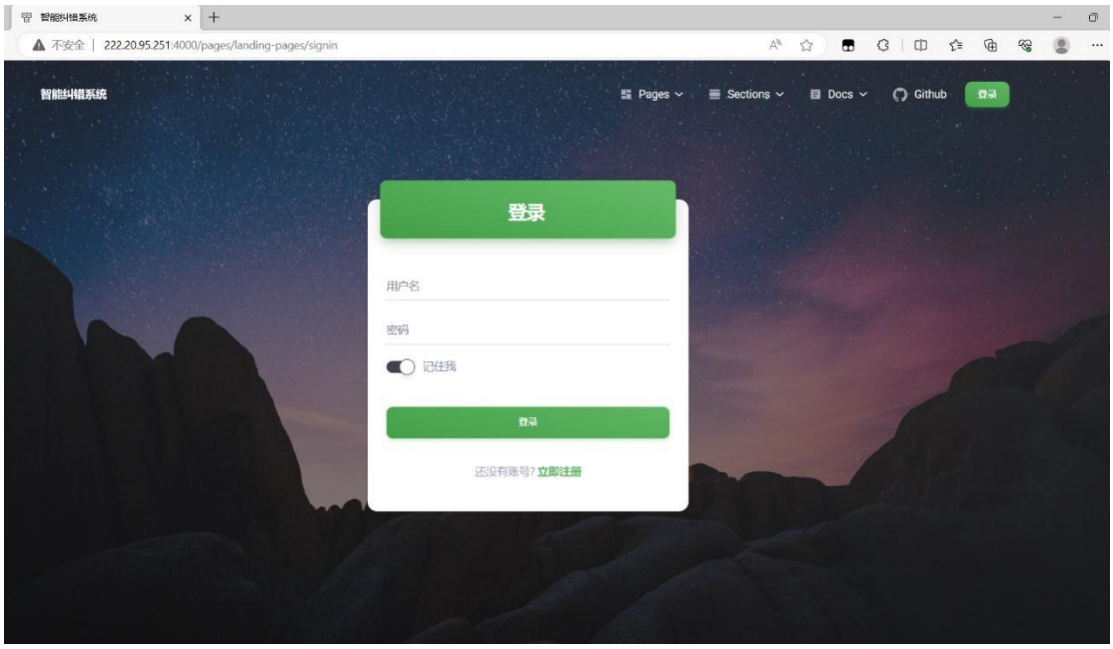
（4）依次点击上方的 项目介绍 项目图表 项目相关图表 Mindmap 按钮，进入软件需求分析介绍页面。



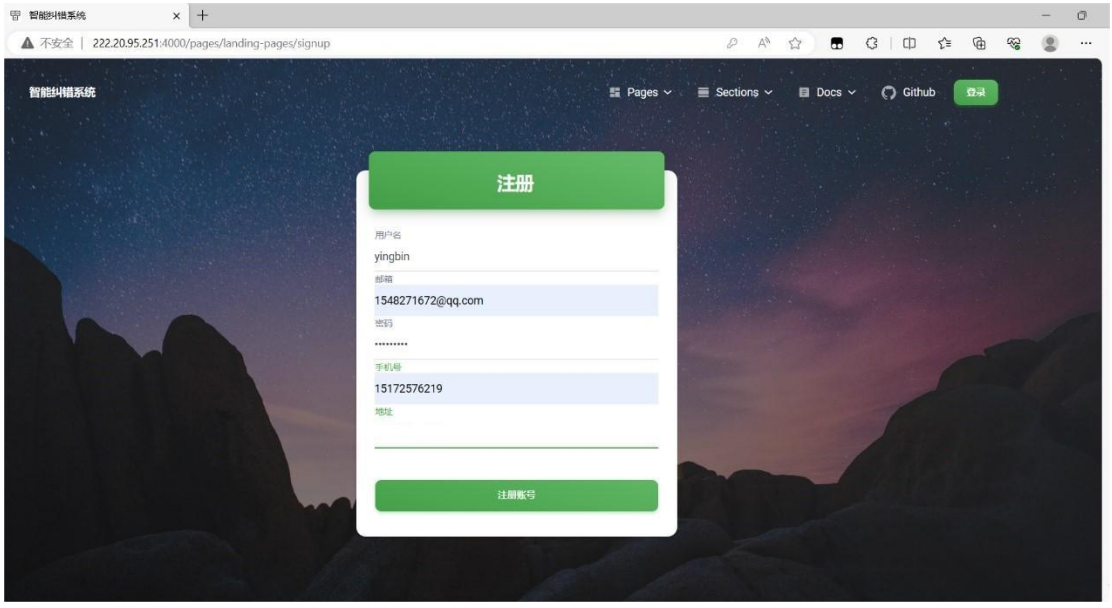
（5）依次点击上方的 帮助 部署指南 部署在你的服务器上 按钮，进入用户本地部署流程说明页面。

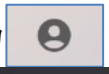
用户可依据本软件给出的部署流程提示，将项目克隆至本地环境并进行个性化迁移。

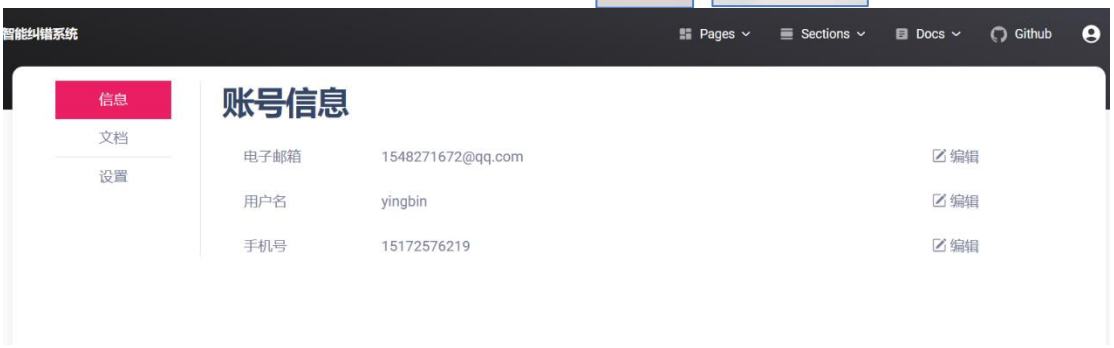
（6）点击 登录 按钮，进入用户登录界面。



（7）第一次登录时，依次填写用户名、邮箱、密码、手机号、地址等信息，填写完后点击 注册账号 按钮完成新用户注册流程。

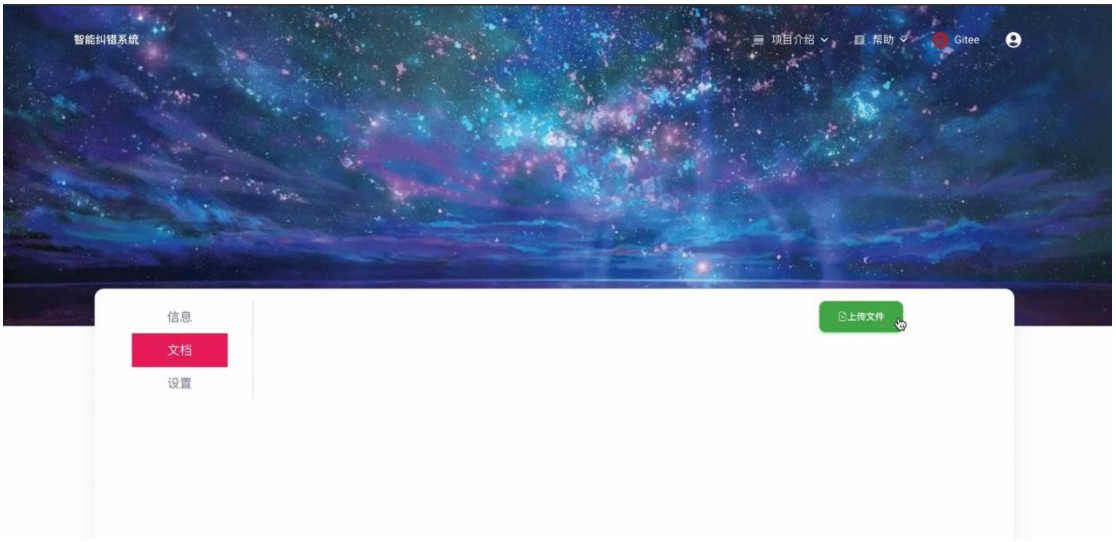


（8）登录并进入系统后，依次点击上方的  我的信息 按钮进入个人信息页面。

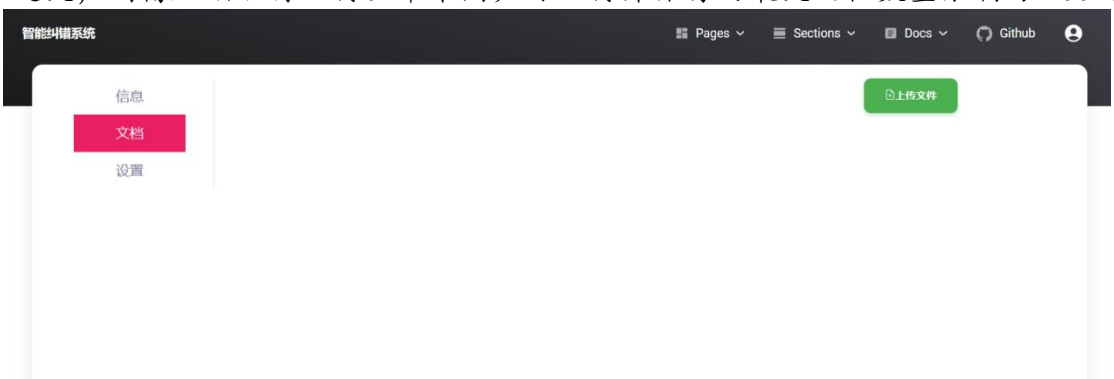


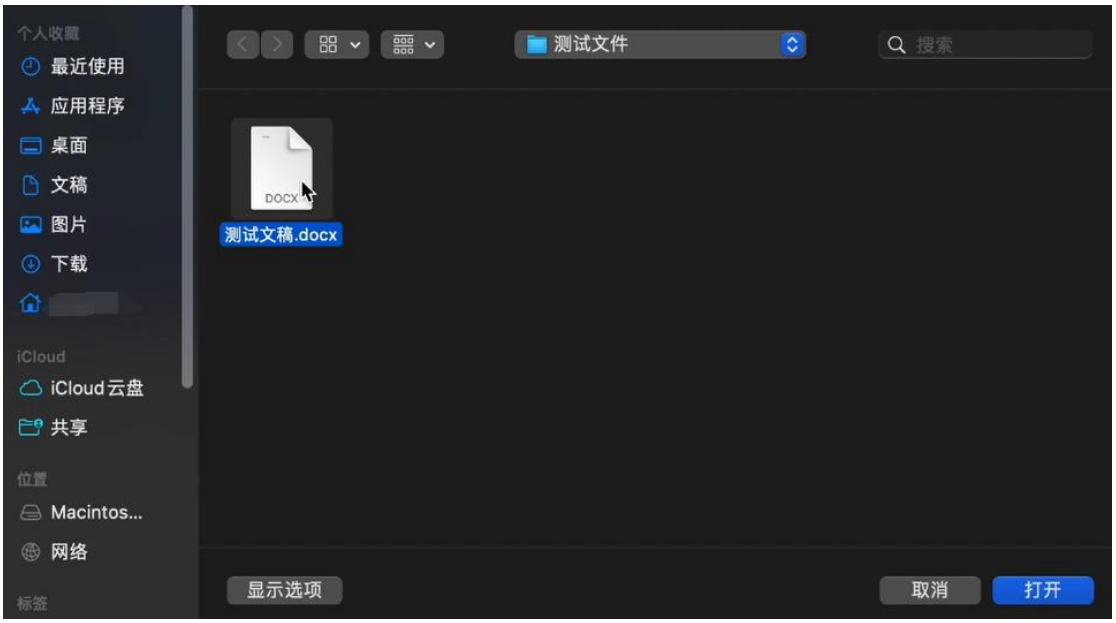
用户个人信息包含电子邮箱、用户名、手机号三项，点击右侧的 编辑 按钮即可选择性修改某一项信息并保存。

点击左侧的 文档 按钮（或在主页面依次点击上方的  我的文档 按钮）进入用户文档页面。新用户的文档页面为空。

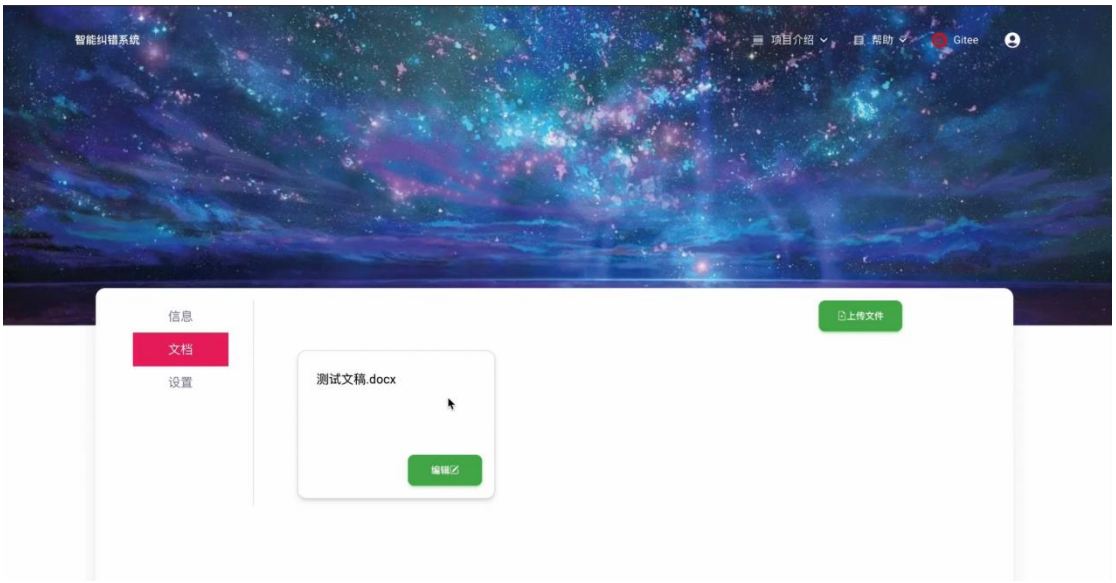


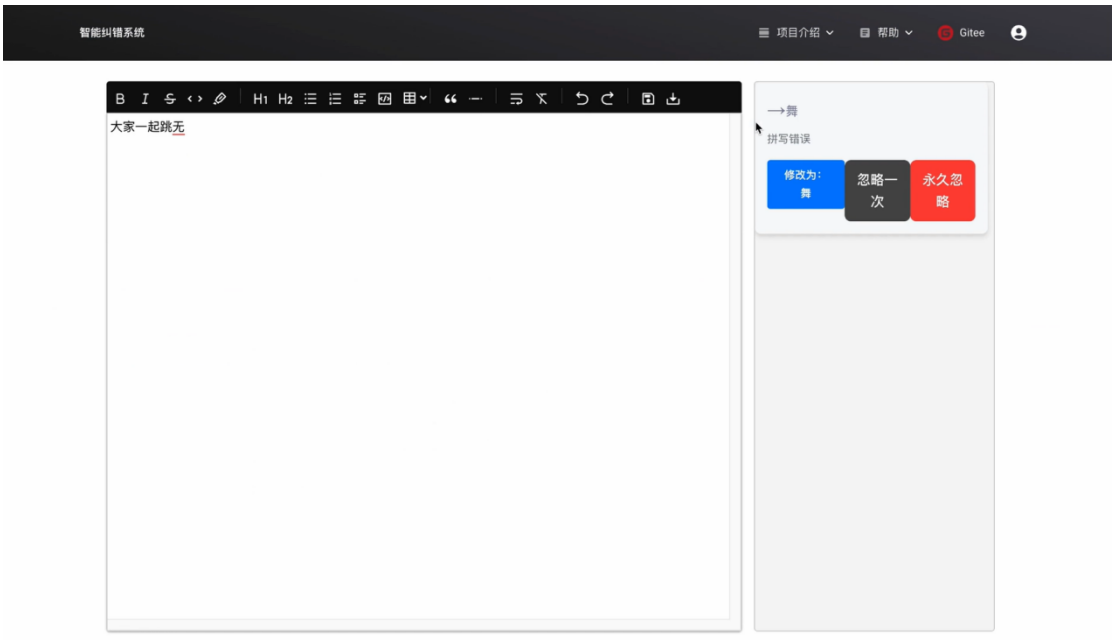
（9）点击 上传文件 按钮，用户上传想要修改的文件（此处以“测试文稿.docx”为例）。系统接受的文档格式包括 DOC、DOCX、PDF、TXT 格式，文档最大大小限制为 100M，若文档过大，则需压缩后再上传。单个用户可上传并保存的最大文档数量限制为 100 个。





（10）上传成功后，相应文件显示在用户文档页面中。点击 编辑 按钮进入文本编辑页面。





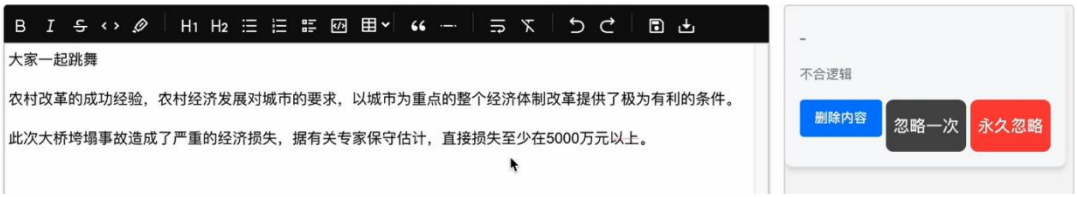
（11）将鼠标移至文本中出错位置，系统提示具体错误，并指明错误类型与修改意见，用户可自行选择是否根据系统建议修改，或者选择“仅忽略一次”或“永久忽略”。

①根据系统建议修改：点击 修改为：舞 按钮，将标注出来的出错位置按照大模型的判断进行修改；

②仅忽略一次：点击 仅忽略一次 按钮，系统仅会将此处错误忽略，对于文本中其余出现同样错误的位置，则仍然会提示错误；

③永久忽略：点击 永久忽略 按钮，系统将忽略文本中所有出现同样错误的位置。

（12）界面右侧也会同步显示当前文本中出现的全部错误，鼠标移至右侧并点击相应按钮也可完成（11）中所述功能。



若当前文本中不存在任何错误，则右侧提示“没有错误”。

（13）文本编辑页面上方黑色条带提供大量文本在线编辑操作，如下所示。用户可在软件使用过程中根据个人需要进行文档的实时编辑，只需将鼠标移至需要修改的位置即可。

下面详细介绍各按钮对应的功能。

B：文本加粗

效果：**此次**

I：文本改为斜体

效果：*桥垮塌*

S：文本删除符号

效果：~~造成子严重~~

‹›：添加文本框

效果：`经济损失，`

：添加文本高亮效果

效果：据有关专家

H1：将文本字号调整为最高级

效果：# 大家一起跳舞

H2：将文本字号调整为次高级

效果：## 大家一起跳舞

：创建一个新列表，且每项前用小圆点标明

效果：

- 列表
- 2
- 3
- 4

：创建一个新列表，且用数字标明各项顺序

效果：

1. 列表
2. 2
3. 3
4. 4

：创建一个新列表，且每项前用方框标明

效果：☐ 223213

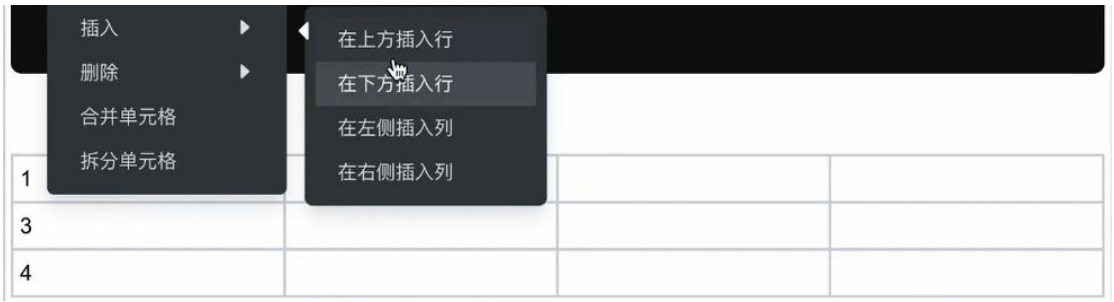
：插入代码块

效果：

```
code block
```

：插入表格（包括表格的插入与删除、单元格的合并与拆分操作。用户可选择“在上方插入行”“在下方插入行”“在左侧插入列”“在右侧插入列”等选项满足个性化要求）

效果：



：撤销上一步操作

：重做上一步操作

：保存在当前文件中所作的修改。用户可返回用户文档页面，下一次打开本文件时，文件版本为最新版本。

：下载当前文件。软件会弹出下载请求的提示框。

附：本软件常用快捷键包括：

剪切文本：Ctrl+X

复制文本：Ctrl+C

粘贴文本：Ctrl+V

全选：Ctrl+A

四、项目知识产权相关信息

本项目已于 2024 年 4 月申请国家软件著作权，流水号为：2024R11L0174476

受理签字：_____　　审查签字：_____

流水号	2024R11L0174476

计算机软件著作权登记申请表

软件基本信息	软件全称	大语言模型下的文本智能纠错软件	版本号	V1.0
	软件简称		软件分类	应用软件
	软件作品说明	◉ 原创　○ 修改（含翻译软件、合成软件） □ 修改软件须经原权利人授权 □ 原有软件已经登记 原登记号： 修改（翻译或合成）软件作品说明：		
开发完成日期		2024年04月08日		
发表状态		○ 已发表 首次发表日期： 首次发表地点： ◉ 未发表		
软件功能和技术特点	硬件环境	开发：PC电脑，16G内存，Intel i7处理器2.60GHz 运行：PC电脑、安卓手机或iOS手机		
	软件环境	开发：Windows 11 64位；node.js环境；PyCharm、Visual Studio Code工具 运行：Android 5.0以上、iOS 10.0以上或Windows 7以上；node.js环境，MySQL 5.5以上版本		
	编程语言	C++ ； Python ； SQL ； TypeScript	源程序量	2500行
	主要功能和技术特点	开发目的：帮助用户更好地检测并改正文本（尤其是学术类文本）中的语法错误和拼写错误，使用大语言模型增强改错性能。　面向领域 / 行业：本软件主要面向教育领域，主要用户包括教师、科研人员、研究生等中文论文写作人士。　主要功能：本软件的功能包括：（1）文本的自动检错和纠错功能，文档中出现的错误将用高亮形式展示，并在屏幕右侧显示大模型提供的修改建议，用户可自行选择是否接受或忽略；（2）完善安全的用户账号功能，包括注册登录、上传下载个人文件、更新个人信息等；（3）在线文本编辑功能，如一键纠错、插入高级格式等。　技术特点：教育软件;人工智能软件；本软件依托于预训练大语言模型技术，将人工智能技术赋能教育行业，属于教育软件和人工智能软件。		

附录 C 文本纠错标注手册

中文学术类文本纠错标注规范手册

1 前言

中文文本纠错是利用自然语言处理技术自动识别并纠正中文语料中包含的各类错误，包括标点错误、拼写错误、语法错误等多方面的内容。

当前中文语病语料库存在的主要问题有二：总体样本数量较小、数据集精细化程度不高。目前公开的中文文本语病数据集有 MuCGEC、NacGEC 等，它们大多聚焦于通用文本类型，没有针对学术类文本进行准确标注和深入研究。而对于文本纠错研究，数据集的质量好坏将在很大程度上决定研究的效果。

为了解决这一短板，本项目旨在生成一份高质量的学术论文数据集，研发并完善一种主要基于大语言模型（LLM）的智能文本纠错系统，针对学术论文领域中常见的拼写错误和语法错误问题提出解决方案。

为了帮助标注人员更高效准确地标注中文学术类文本纠错数据，我们编写了下面的标注规范。本规范将定义常见的中文学术类文本错误分类体系，并给出相应的例句；介绍各类文本错误的纠正方法，并给出纠错示例；介绍文本纠错语料的标注准则和技巧。

值得注意的是，尽管本研究的最终目的是在中文学术类文本上表现出出色的纠错性能，但在目前的标注任务中主要针对的是文本中出现的明显错误，而尽量不修改文本的表述风格。在取得高质量数据集后，我们将首先确保实现基本的检错和改错功能。此后，在条件允许的情况下，则会进一步根据学术类文本的特点尝试对文本的叙述视角、语气等要素进行优化。

2 标注原则与形式

2.1 标注原则

由于本次标注的语料全部为学术论文，与以往的通用文本内容有所不同，我们在这里简单地规定汉语学术类文本纠错数据标注的准则。

（1）标注者标注的数据绝大多数为纯中文数据，但如果遇到了中英文混搭的句子，那么对于中文需要纠正所有类型的错误，而对于英文只纠正拼写错误。

（2）对于标点符号出错，需要将其视为一个语病并加以改正。

（3）对于句中出现的重复现象，如果不是写作者有意为之，则需要进行纠正。

（4）若句子的含义较清晰，但表述混乱，则需要进行纠正。

（5）若单个句子中出现多种类型的语言错误（尽管这种情况发生的可能性不高），请选取最合适的一种方式进行纠正。

标注过程中，对于拼写类型和词语误用类型的错误，要善于利用已有的词典资源，先充分理解再标注。请尽量不要贸然标注，影响质量。可以利用的资源如：百度汉语（hanyu.baidu.com），在线新华字典（https://zidian.aies.cn）等。若遇到人名、地名、机构名等命名实体类型的拼写错误，可以借助互联网搜索引擎判别是否正确。

2.2 标注形式

在 Word 文档中进行标注时，请将原文中出现错误的句子的头尾两处用字符串“[[[”“]]]”

进行框选。请注意包含一整个句子，避免丢失句首、句末的标点符号。然后，在原始错误句子的“]]]”字符串后，用字符串“[[[”“]]]”框选改正后的句子的头部和尾部。示例如下。

原文：本文综述了近年来在计算机科学领域的一些重要研究进展。首先，我们回顾了机器学习领域的发展，包括深度学习、强化学习和迁移学习等方面的研究。其次，我们探讨了计算机视觉领域的最新进展，如目标检测、图像分割和人脸识别等技术。然而，我们还讨论了自然语言处理领域的研究现状，包括机器翻译、情感分析和问答系统……等方面的研究。最后，我们总结了当前计算机科学领域的挑战和未来的研究方向，包括解释性 AI、可解释性和隐私保护等问题。

标注后的文档：本文综述了近年来在计算机科学领域的一些重要研究进展。首先，我们回顾了机器学习领域的发展，包括深度学习、强化学习和迁移学习等方面的研究。其次，我们探讨了计算机视觉领域的最新进展，如目标检测、图像分割和人脸识别等技术。[[[然而，我们还讨论了自然语言处理领域的研究现状，包括机器翻译、情感分析和问答系统……等方面的研究。]]][[[此外，我们还讨论了自然语言处理领域的研究现状，包括机器翻译、情感分析和问答系统等方面的研究。]]]最后，我们总结了当前计算机科学领域的挑战和未来的研究方向，包括解释性 AI、可解释性和隐私保护等问题。

3 中文文本错误分类

为了便于理解，下面将依次列举学术类文本中常见的中文文本错误，并给出一定的例子。请标注者在进行标注之前首先阅读下面的示例。

3.1 标点错误

标点错误主要是在语句中多用、少用或误用了标点符号，根据实际情况比较容易判断。

3.1.1 标点冗余

标点冗余是指在句子中不必要的地方插入了标点符号。

【例 1】

错误句子：根据调查结果，我们发现：大多数受访者对这个问题持有积极的态度。

正确句子：根据调查结果，我们发现大多数受访者对这个问题持有积极的态度。

【例 2】

错误句子：这个理论在实践中有着丰富的应用，和广阔的前景。

正确句子：这个理论在实践中有着丰富的应用和广阔的前景。

【例 3】

错误句子：在这个研究中，我们使用了多种方法，包括实地观察、问卷调查、深度访谈……等等。

正确句子：在这个研究中，我们使用了多种方法，包括实地观察、问卷调查、深度访谈等等。

3.1.2 标点缺失

标点丢失主要是指在句中、句末漏写了本应存在的标点。

【例 1】

错误句子：我们使用了多种统计方法，例如 t 检验方差分析回归分析等。

正确句子：我们使用了多种统计方法，例如 t 检验、方差分析、回归分析等。
说明：这里在多条列举时，没有使用顿号进行分隔。

【例 2】
错误句子：研究结果表明这个理论有广泛的应用领域比如人工智能数据挖掘等。
正确句子：研究结果表明，这个理论有广泛的应用领域，比如人工智能、数据挖掘等。
说明：这里整个长句未使用标点符号。

【例 3】
错误句子：根据计算机视觉算法与应用一书中的方法，我们进行了图像分割的研究。
正确句子：根据《计算机视觉算法与应用》一书中的方法，我们进行了图像分割的研究。
说明：这里引用了书籍的名称，但未使用书名号。

3.1.3 标点误用

标点误用主要是指混淆了某些标点符号的用法，并在句中错误使用。
【例 1】
错误句子：我们得出了一个重要的结论:这个新方法非常有效,可以提高产品质量和生产效率.
正确句子：我们得出了一个重要的结论：这个新方法非常有效，可以提高产品质量和生产效率。
说明：这里将中文标点符号误用为英文标点符号。

【例 2】
错误句子：我们的研究目标是解决一个关键问题，如何提高系统的性能，稳定性，实用性和便捷性。
正确句子：我们的研究目标是解决一个关键问题，如何提高系统的性能、稳定性、实用性和便捷性。
说明：这里将顿号误用为逗号，导致“一逗到底”。

【例 3】
错误句子：根据《大数据分析方法》的理论，我们进行了《数据挖掘》的实验研究。
正确句子：根据《大数据分析方法》的理论，我们进行了“数据挖掘”的实验研究。
说明：书名号和引号发生混淆。

3.2 拼写错误

主要指由于作者疏忽导致句子中出现明显不符合原意的词语，即错别字。学术类文本中，要特别关注专业术语是否出现此类错误。

3.2.1 中文术语拼写错误

【例 1】
错误句子：统计学中的“方程”分析方法可以用于数据处理。
正确句子：统计学中的“方差”分析方法可以用于数据处理。

【例 2】
错误句子：在人工智能领域，我们研究了机械学习和深入学习的算法。
正确句子：在人工智能领域，我们研究了机器学习和深度学习的算法。

3.2.2 英文术语拼写错误

【例】
错误句子：在人工智能领域，我们研究了 natural network（神经网络）的训练方法。
正确句子：在人工智能领域，我们研究了 neural network（神经网络）的训练方法。

3.2.3 其它拼写错误

【例 1】
错误句子：在生物学研究中，我们使用了“基因饭组”进行分析。
正确句子：在生物学研究中，我们使用了“基因组”进行分析。

【例 2】
错误句子：在本文中，我们引用了分子构造和反应机理理论。
正确句子：在本文中，我们应用了分子构造和反应机理理论。

3.3 语法错误

3.3.1 成分冗余

主要指相同或相近意思的词语在句子中同时出现，往往是两个词相邻出现，导致语义重复。

【例 1】
错误句子：城镇化攸关到亿万人民的生活质量，它不是简单的城镇人口比例增加和城市面积扩张，而是在人居环境、社会保障、生活方式等方面实现由“乡”到“城”的转变。
正确句子：城镇化攸关亿万人民的生活质量，它不是简单的城镇人口比例增加和城市面积扩张，而是在人居环境、社会保障、生活方式等方面实现由“乡”到“城”的转变。
说明：句中“攸关”本身包含“到”的含义。

【例 2】
错误句子：根据我们的初步估计，这项研究的样本处理时间大约在 2 个月左右。
正确句子：根据我们的初步估计，这项研究的样本处理时间大约在 2 个月。
说明：“大约”和“左右”同义。

【例 3】
错误句子：通过对文本的深入研究和批评分析，我们逐渐培养起从经典作品中理解和感知美学价值的能力，产生出来了解读文学作品的本领。
正确句子：通过对文本的深入研究和批评分析，我们逐渐培养起从经典作品中理解和感知美学价值的能力，产生了解读文学作品的本领。

【例 4】
错误句子：这款新一代芯片具有多达到 64 个核心，能够同时处理复杂的计算任务。
正确句子：这款新一代芯片具有多达 64 个核心，能够同时处理复杂的计算任务。

3.3.2 成分残缺

主要指句中必要的词语成分缺失，导致表意不明或歧义。比较常见的情况有主语缺失、宾语缺失、谓语缺失等。

【例 1】
错误句子：在过去几十年的研究中，发现了许多与遗传相关的疾病。

正确句子：在过去几十年的研究中，研究者发现了许多与遗传相关的疾病。
说明：句中缺少主语。

【例 2】
错误句子：自 2000 年起就掀起了“互联网+”的新浪潮。
正确句子：自 2000 年起全社会就掀起了“互联网+”的新浪潮。
说明：句中缺少主语。

【例 3】
错误句子：只有按照实验设计，才能确保结果的有效性。
正确句子：只有按照实验设计操作，才能确保结果的有效性。
说明：句中缺少谓语。

【例 4】
错误句子：在治疗方案后，研究团队对患者的病情进行了跟踪观察。
正确句子：在实施治疗方案后，研究团队对患者的病情进行了跟踪观察。
说明：句中缺少谓语。

【例 5】
错误句子：本文介绍了 LK-99 超导体的制备过程与制备方法，提出了该材料成为跨时代的介质。
正确句子：本文介绍了 LK-99 超导体的制备过程与制备方法，提出了该材料成为跨时代的介质的结论。
说明：句中缺少宾语。

【例 6】
错误句子：针对“AI 机器人会抢美国工人饭碗”，新版路线图指出，过去五六年里美国新增约 60 万个制造业岗位。
正确句子：针对“AI 机器人会抢美国工人饭碗”的观点，新版路线图指出，过去五六年里美国新增约 60 万个制造业岗位。
说明：句中缺少宾语。

【例 7】
错误句子：对于第一组实验数据，第二组实验数据，表现出相当的稳定性与鲁棒性。
正确句子：对于第一组实验数据，以及第二组实验数据，都表现出相当的稳定性与鲁棒性。
说明：句中缺少关联词。

【例 7】
错误句子：当技术进步提出了人性的考验，就可能引发伦理困境。
正确句子：当技术进步提出了对人性的考验，就可能引发伦理困境。
说明：句子缺少介词，导致表意不明。

3.3.3 搭配不当

主要指对某个词语的词义、词性运用不当。
【例 1】
错误句子：研究人员通过实地考察和文献综述，给出了一个诡异的观点。

正确句子：研究人员通过实地考察和文献综述，给出了一个奇异的观点。

【例 2】
错误句子：在这个研究领域，有很多前人的经验可供我们借鉴和抄袭。
正确句子：在这个研究领域，有很多前人的经验可供我们借鉴和参考。

【例 3】
错误句子：我们通过对样本的深入分析，得出了一个不可能的发现和结论。
正确句子：我们通过对样本的深入分析，得出了一个此前未出现过的发现和结论。

3.3.4 语序不当

主要指句子中词语或分句的顺序不合理，即不符合语法习惯或不符合事理。
【例 1】
错误句子：在这篇论文中，作者探讨了基因编辑技术在治疗遗传性疾病上的详细应用。
正确句子：在这篇论文中，作者详细探讨了基因编辑技术在治疗遗传性疾病上的应用。
说明：“详细”应置于“探讨”之前。

【例 2】
错误句子：研究发现，慢性疲劳综合征患者的免疫系统存在异常，主要表现为淋巴细胞亚群的比例失调和细胞因子的分泌异常。此外，该综合征还与神经内分泌和代谢紊乱有关。
正确句子：研究发现，慢性疲劳综合征不仅与神经内分泌和代谢紊乱有关，还与免疫系统异常有关，主要表现为淋巴细胞亚群的比例失调和细胞因子的分泌异常。
说明：两个“有关”的句子应放在一起。

【例 3】
错误句子：该论文主要探讨了人工智能在医疗保健领域的应用，首先详细阐述了人工智能在医学影像诊断、疾病预测、药物研发等方面的应用，接着介绍了人工智能的基本原理和分类，最后讨论了其未来的发展方向和挑战。
正确句子：该论文主要探讨了人工智能在医疗保健领域的应用，首先介绍了人工智能的基本原理和分类，接着详细阐述了人工智能在医学影像诊断、疾病预测、药物研发等方面的应用，最后讨论了其未来的发展方向和挑战。
说明：应该先介绍基本原理和分类，再介绍应用情况。

3.3.5 不合逻辑

主要指句子在语法上虽然没有问题，但在表达事理、语言习惯等方面不符合逻辑。主要表现在概念使用、判断推理等方面的错误。
【例 1】
错误句子：在这篇论文中，作者探讨了电子商务在古代社会中的应用和影响，详细分析了其优缺点、市场现状和发展趋势等问题。
正确句子：在这篇论文中，作者探讨了电子商务在当今社会中的应用和影响，详细分析了其优缺点、市场现状和发展趋势等问题。
说明：电子商务不可能在古代出现。

【例 2】
错误句子：研究发现，长期饮酒会导致肝脏损伤和肝硬化，但过量饮酒有益于预防心血管疾病。

正确句子：研究发现，长期饮酒会导致肝脏损伤和肝硬化，但适量饮酒有益于预防心血管疾病。

说明：“过量饮酒有益于预防心血管疾病”不合常理。

【例 3】

错误句子：在这篇论文中，作者调查了近期发生的网络攻击事件，分析了被攻击者的动机和目的，并提出了预防未来攻击的建议。

正确句子：在这篇论文中，作者调查了近期发生的网络攻击事件，分析了攻击者的动机和目的，并提出了预防未来攻击的建议。

说明：“被攻击者”不会有“目的和动机”。

3.3.6 句式杂糅

主要指把两种不同的句法结构混杂在一个句子中，结果造成语句结构混乱、语义纠缠。

【例 1】

错误句子：曾被计划经济思想束缚下的人们也觉悟起来。

正确句子：在计划经济思想束缚下的人们也觉悟起来。

说明：举棋不定。此句应该在“曾经被……束缚的……”和“在……束缚下的……”两种格式中选用一个。

【例 2】

错误句子：反动分子的阴谋活动是应当加以揭露，而且能够把它揭露的。

正确句子：反动分子的阴谋活动我们是应当加以揭露，而且能够把它揭露的。

说明：反客为主，就上半句说，谁“加以揭露”，当然是“我们”，但这个词隐而未现，正式主语应当是受揭露的“反动分子的阴谋活动”。可是下半句的“能够把它揭露的”主语就不可能还是“反动分子的阴谋活动”，而只能是“我们”。这一句应该在“是应当”前加“我们”。

【例 3】

错误句子：这家工厂虽然规模不大，但曾两次荣获省科学大会奖，三次被授予省优质产品称号，产品远销全国各地和东南亚地区。

正确句子：这家工厂虽然规模不大，但曾两次荣获省科学大会奖，其产品远销全国各地和东南亚地区，三次被授予省优质产品称号。

说明：中途易辙，这句话的主语是“工厂”，但“三次被授予省优质产品称号”的主语只能是这家工厂的产品，而不能是“工厂”。