文件编号：码梦工坊-SWC2020-20200182

受控状态：■受控　□非受控

保密级别：□公司级 □部门级 ■项目级 □普通级

采纳标准：CMMI DEV V1.2

智慧学术论文行文指导快应用

Smart Guiding of Academic Paper Writing

项目开发文档

Version 1.0.2.200609

2020.6.9

Written by 码梦工坊

目录

记录更改历史

序号	更改原因	版本	作者	更改日期	备　注
1	添加项目概述	0.0.1	队员 A	2019.12.16	初始编辑
2	修改项目概述	0.0.2	队员 B	2019.12.18	统一了上下文的思路，使得表达更加清晰。
3	添加可行性分析	0.1.1	队员 B	2019.12.18	初始编辑
4	修改项目概述	0.1.2	队员 A	2019.12.28	使项目概述更加贴切与专业
5	添加开发计划	0.2.1	队员 A	2019.12.28	初始编辑
6	添加需求分析	0.3.1	队员 A	2019.12.29	初始编辑
7	调整文档格式	0.3.2	队员 A	2019.12.30	
8	提高文章通顺度	0.3.3	队员 B	2019.12.30	统一了上下文的思路
9	添加用例规约	0.4.1	队员 A	2019.12.31	初始编辑
10	调整文档格式	0.4.2	队员 A	2019.12.31	
12	更新内容	0.4.3	队员 A	2020.4.19	第一、三部分
13	添加内容	0.5.1	队员 A	2020.4.20	概要设计
14	添加内容	0.6.1	队员 B	2020.4.21	数据库设计
15	添加内容	0.7.1	队员 A	2020.4.21	详细设计
16	修正内容	0.7.2	队员 A	2020.4.21	
17	修改用例规约	0.8.1	队员 B	2020.4.22	
18	错误修正	0.8.2	队员 A	2020.4.22	
19	添加并更新内容	0.9.1	队员 A	2020.6.9	
20	多次修正内容	1.0.2	队员 A	2020.6.9	

1 项目概述

1.1 项目背景

本项目涉及人工智能（AI）/自然语言处理（NLP）领域。自然语言是人类智慧的结晶，而自然语言处理是人工智能中最具挑战性的问题之一。近年来，自然语言处理处于快速发展阶段，一方面，各种词表、语义语法词典、语料库等数据资源日益丰富；另一方面，词语切分、词性标注、句法分析等技术也在快速发展。新方法、新理论、新模型不断涌现，推动了该研究以及邻域研究的繁荣发展。互联网与移动互联网市场对于自然语言处理的迫切需求，为自然语言处理研究发展提供了强大的市场动力。但相比于性能趋于饱和的计算机视觉和语音识别技术，自然语言处理因技术难度太大、应用场景过于复杂，现阶段的研究成果尚未能够达到目标的高度。

本项目立足于自然语言处理最新、最有效的模型与技术，致力于解决中文学术论文写作过程中的错别字、通顺程度与书面语规范的问题。它的核心技术基于NLP 领域中的词向量技术、Attention 机制、预训练技术与中文全词遮罩技术，并且深度利用 BERT（Bidirectional Encoder Representation from Transformers）预训练模型在文本分类任务的良好表现，辅之以 ERNIE 模型在错字检测、语句通顺程度评估任务中的优势。本项目在上述几项关于中文学术论文的行文指导问题的求解上能够达到该领域现有技术水平下较好的效果。

1.2 项目定位

1.2.1 应用场景

中国各高校本科生与研究生在初涉学术论文写作的领域时，存在着一些反复出现的问题。如何写出在语言上达标的毕业论文，提高审阅通过的概率，成为本科生的一大难题。不仅如此，对于想在学术期刊上发表自己科研成果的研究生来说，也同样面临着这样的问题。他们的毕业论文中往往存在着许多的低级的打字错误、句子不通顺以及口语化表达的问题。这类缺陷是学术论文的硬伤，削弱

了论文的专业性与严谨性，也直接影响着审阅者对于学术论文总体水平的评估。所以，这些缺乏经验的论文写作者往往会耗费大量时间与精力解决毕业论文中语言方面的错误。有些学生会求助于他们的导师，但大量的论文以及频繁出现的语言上的低级错误往往让导师们烦不胜烦。

如何快速指出并纠正此类学术论文的语言表达上的问题，让导师与学生能够将注意力放在学术论文本身，成为了一大痛点。

1.2.2 目标人群

本项目的目标人群是中国各高校的本科生、研究生以及他们的导师。

对于学生来说，他们在学术论文的写作这个领域存在着经验不足的情况，在论文写作中往往出现低级打字错误、句子表达不通顺、不自觉的口语化的问题。他们也急切的需要一种较为客观的、全面的评价来衡量他们的学术论文的文章质量，以此来引导他们在学术论文上的写作。另一方面，对于“学生党”而言，他们难以支付高昂的人工服务费来修改、润色文章，而且他们也不需要这样过于专业的服务。

对于导师而言，他们长期大量审阅学生写作的学术论文，学术论文中的语言硬伤往往让他们感到不适，而他们并没有时间与精力仔细查找并指出这类基础错误。同时，这也导致导师难以直接确定学术论文的写作档次，甚至影响到导师对于学术论文本身的理解与判断，使得他们难以将注意力集中到论文本身上。

1.3 项目方案

对论文中出现的打字错误，该项目通过自然语言处理模型中的自然语言处理基础技术来实现对于打字错误的检测与纠正。

对于文章中的句子表达不通顺的问题，该项目利用 DNN 模型在语法特征提取的优势，以句的粒度，针对这类问题进行量化处理。对于句结构中的异常量，在他们高于所设置的阈值时，给予用户警告。

对于表达口语化的问题，该项目深度基于针对项目定制的 BERT 预训练模型，

并利用中文全词遮罩技术，在预训练层次上对该模型在中文下表现进行改进。而后将此综合性问题在技术上转化为 BERT 预训练模型表现最为优秀的文本分类（Text Classification）的问题。针对文本分类问题，BERT 在读入语料之时采用了“注意力”机制，使得 BERT 能够针对语料进行整体的分析从而获得预料的特征，这恰恰适应了本项目针对表达口语化特征的提取要求。

然后利用百万级别的中文语料数据集，对预训练模型进行中文领域的学术论文语料监督训练（fine-tuning），构建并微调针对 Transformer 编码器（Encoder）进行修正的全连接层来对预训练模型进行微调（Fine-tuning），以适应问题场景对输入语料进行分类，经过对结果运算后筛选后就能找出文章中口语化部分。

对于给出学术论文综合评价这个问题，该项目将结合上述处理结果与利用函数，根据问题词句的个数、频率、分布严重性，结合学术论文的整体的口语化倾向给出综合加权评分。

1.4 项目目标

该项目将利用 AI 下的多种 NLP 研究领域的最新方法与模型，基于深度学习，最大限度地指出中文学术论文中的低级打字错误、句子不通顺与口语化表达等问题，及时地为用户指出这些问题。如果可能的话，产品将用户提供较为可行的修改建议。该项目的产品也能够依照相应的语言错误个数、频率与严重性，结合严谨性评判标准给予用户该学术论文的综合评分。

另外，该项目的作品与用户的交互将通过卡片式快应用进行，使得用户能够随取随用。并且，利用快应用卡片技术直观地告知用户其上传任务的执行情况、其论文写作水平的提升情况。最后，通过蓝牙技术让用户能够快速地将报告在其常用设备间传递，或者通过蓝牙打印机进行打印。项目作品将在界面与操作流程上推行简化设计，力求让该场景下此问题的解决过程更加标准化、快捷化、智能化。

1.5 项目价值

当下，本科生、研究生在进行毕业论文或科研论文写作时，往往存在着经验

不足、书面表达能力有限等问题，这就迫切需要一种可以解决论文写作时的一些语言表达问题，并能够提供合理性的修改建议与行文综合的评价的合理方案。

目前，对于这类人群来说，现有的解决方案往往存在着价格昂贵与使用不便的缺陷。该项目产品的出现使得用户人群，能够迅速找出论文中用词上与表达上的明显缺陷、快速得到论文的综合评分，提高论文审阅效率，以摆脱中文学术论文写作上所面临的问题的困扰，且能随取随用。

从本产品的效益上来说，一方面，本项目作品解决了中国各高校本科生、研究生的毕业论文与科研论文的写作中语言表达上的问题，极大提高了论文质量，其自动化、便捷与低收费的特点能够使得服务覆盖绝大部分目标人群。另一方面，本项目作品解决了导师审阅此类论文中的种种困扰，极大地提高了论文审阅的效率。

2 开发计划

2.1 最终呈现形式

该项目产品最终将以三部分呈现，快应用端、主服务端、分布式 GPU 计算端。

其一，产品与用户交互的部分以快应用的形式呈现，用户通过快应用界面使用产品的所有功能，快应用端将采取简化设计，尽可能对用户隐藏不必要的复杂机制，降低学习成本。另外，快应用端也将具备一定的设备间交互能力，使得用户能够在常用设备间传递运算结果。这部分在设计上也称为客户端、前端。

其二，产品的针对文本的章结构、段结构、句结构、词结构的预处理与文本的预分析功能将部署在主服务端。主服务端在逻辑上只能够有一个，且由主服务端通过 Restful API 的设计风格的接口与快应用端进行通信。一方面，主服务端将使用异步处理技术与线程池技术对快应用端发来的数据进行处理。另一方面，主服务端将通过协议将经过预处理与预分析的数据分发给一个或者多个分布式 GPU 计算端。主服务端我们也将简要的称为服务端、后端。

其三，为了解决深度学习预训练模型的运算的复杂程度高、要求 GPU 单独占用、服务器租用成本高、启动时间长的难题，本项目结合预训练模型的特点、系统整体吞吐量与运算效率的因素，经过权衡，我们创新性地为 BERT 预训练模型编写了分布式 GPU 计算端模块。用户通过快应用端上传的文本数据在主服务器端经过预处理与预分析后，通过原创的分布式计算协议与基于优先级的调度算法（使用子服务器护照与签证的概念），批量分发到多个分布式 GPU 计算端进行深度学习神经网络运算处理。多个分布式 GPU 计算端对主服务端批量分发的文本数据进行并行计算后，整个系统将在尽可能短的时间内批量输出多个用户上传文本的运算结果，而后运算结果将通过协议从分布式 GPU 计算端送回主服务端。分布式 GPU 计算端在提高系统的总吞吐量的同时，也大大降低了部署成本。分布式 GPU 计算端，我们将简称为计算端。

2.2 主要功能描述

该项目的软件产品的客户端功能主要分为三大类：启发模式、评价模式。下面将对这三个模式及其子功能进行介绍。

该项目软件产品的启发模式含有一个单向流程，而一个流程有三个步骤。每个步骤下，客户端将使用不同的方法扫描并标记出用户提交的学术论文中出现的各种问题，然后客户端辅助用户发现、修改该阶段学术论文中的特定问题。

与启发模式三个阶段想对象的三个主要子功能为：对用户的输入的中文学术论文文本进行打字错误检测与修正、对每句话的通顺程度进行评估并给出评估结果、检测句子情感倾向的问题并对非中立的句子提出修改建议。

打字错误的检测与修正是自然语言处理模型结合上下文对用户输入的中文学术论文文本中的一些错字、别字与偶尔的打字错误（如与输入法有关的拼音相近或笔画相近的输入错误）的检测，并在这些地方做出标记，而后，自然语言处理模型通过语境分析，将找出最适合当前语境的字或词，并结合模型提示并给出修改建议，用户可以根据自己的需要决定是否接受系统给出的修改建议，同时也可以在一键修改模式下将这些错字替换成建议修改的字。

句子通顺度的评估是指，在将各个段落划分成独立的句子之后，对每个句子调用 DNN 语言模型进行分析，该语言模型会通过检测词向量以及各个单独的词出现在句子中的概率对本句话进行语言通顺度的评估。对于 DNN 得分在阈值以上的模型，我们将会对这句话做出标记，并提醒用户此句话存在一定程度的通顺度问题，以帮助用户改正。

句子口语化问题与严谨性问题的检测与指导方案的生成是通过 fine-tuning 好了的自然语言处理模型进行文本情感倾向判断，此模型将对划分好句子的段落逐句进行文本分类操作，找出情感倾向过于浓烈的句子，或消极或积极，并做出标记提示用户进行修改。而后，软件产品将对经过初步处理后的句子进行逐一扫描，来剥离文本中不符合学术论文一般表达习惯的句子，通过自然处理预训练模型中的机器翻译应用模型、结合语境语义、按照优秀论文的标准，将这些不严谨的句子翻译成符合学术论文表达规范且较为严谨的句子，并在此基础上生成指导

方案。软件产品将对不严谨的句子做出标记，并将指导方案呈现给用户，这样可以大大提升学术论文的整体严谨度。

本产品的评价模式可以针对论文给出综合加权分数的功能，用户将通过客户端引导来进入评价模式获取分数。该评价模式将从三个方面入手：错误文本数量、不通顺的句子的数量以及在 DNN 语言模型下的评分、情感倾向强烈的句子数量以及强烈的程度。这三者经过一定的算法之后会生成一个针对整篇文章的分数，此分数仅代表文章遣词造句的严谨性，而不代表文章内容的质量。

2.3 运行环境

主服务端采用自动构建、自动部署的方案，对服务器的硬件与网络要求较低。首先，主服务器端仅要求部署服务器带有固定的公网 IP、具备有不少于 5M 的上行带宽。在处理能力上，部署服务器的 CPU 推荐具备双核四线程、运行内存推荐具备有 4GB、剩余储存空间具备有 16GB。主服务器部署的时候推荐安装有 Jenkins 来支持主服务端的自动构建与自动部署（直接编译效果相同）。Jenkins 可以连接到项目的私有仓库地址，设置好 Web Hook 与构建状态反馈。这样，维护人员能够快速对主服务端存在的漏洞进行修补，使得主服务端能够实时保持最新版本。部署服务器推荐安装 JDK 1.8_241，JRE 1.8_241（实际上仅安装 JDK1.8 即可）。另外部署服务器推荐安装 Maven、Git，以进一步实现自动构建功能。

分布式 GPU 计算端的部署采取多计算机部署的方案，支持根据系统的事实际负载增减计算端数量。首先，在网络环境上，并不要求计算端部署在带有公网 IP 的服务器上，只要求部署的计算机能够连接到互联网。另外，计算端允许 NAT 转发的存在。在运算能力上，计算端不要求计算端具备有性能较强的 GPU。计算端推荐的 GPU 硬件的计算能力仅与 NVIDIA GTX 1050ti 相当，当然可以更高。在 CPU 能力足够的情况下，计算端甚至可以不安装 GPU 就能正常作业。在软件环境上，只要求 Python 的版本为 3.6，并且安装 TensorFlow-GPU 1.14，CUDA 10.1 以及配套的 CDNN （或者只安装 TensorFlow 1.14）。Python 只要求安装最新版本的 shutil、requests、hashlib 与 pandas。

2.4 验收标准

客户端界面简洁美观且能够在运行环境标准下正常提供开发文档中所述的所有功能以及每种功能的简要说明与使用帮助文档。另外，客户端的反馈功能与自动更新功能应该健全。同时，客户端必须有极高的用户友好性，尽量保证用户对于看到的模块和组件可以第一时间就知道其准确功能。

服务端应该符合运行环境标准，正确部署了与客户端功能相对应的运算模型，与接口实现符合标准并通过测试。服务端应该能够 7*24h 接受客户端发起的请求提供运算模型的运算服务，对于客户端的每次请求，服务端都应该在 10s 内返回给客户端对应的数据。此外，服务端软件应该能够在符合运行环境标准的机器上的进行快速部署。同时，服务端必须保证在高并发的条件下不会崩溃，且必须保证分布式计算端对于每个计算任务调用时的线程安全性。

客户端与服务端在验收前应该进行并通过相应的压力测试，测试在多个客户端进行频繁请求的情况下，服务端提供返回结果的延迟情况以及是否符合标准。

2.5 关键问题

对于服务器硬件条件不满足运行环境标准的情况，特别是在服务器显存与内存达不到标准的情况下，将影响服务端软件产品的运行时间。对于这种情况，服务端软件能够提供统一集中的配置文件来方便服务器系统管理人员对业务运算模型的计算参数（特别是神经网络节点参数个数）的实时调整，以降低服务器的运算量与并发数，适当提高客户端请求的等待时间来降低服务器压力。

对于服务器出现故障，暂时无法提供服务的情况。由于服务端软件将使用 Docker 进行封装，因此服务端软件能够获得在服务器之间快速部署的能力。这种情况下，服务端软件将自动地部署在备用服务器上，快速恢复服务能力。

对于某一个分布式计算端突然出现故障无法继续运行仍未完成的任务的问题。此情况下，该计算端会紧急执行中断方案，保留当前任务的数据，并向主服务器发送产生的故障信息，在限定时间内如果此计算端恢复正常运行，并且向主服务器索取任务，则会把此任务重新分配给此计算端，而后可以继续执行此任务。

若是护照过期则会释放任务，此任务会被调度给其他发送请求的计算端。

对于数据安全问题。在软件产品的总体设计上，一些不敏感的会话的上下文数据与临时处理结果将在其有效期内储存在客户端软件上，这样就可以大大降低服务能力恢复后用户会话的恢复难度。而对于另外的关键与敏感的用户数据将储存与云服务器分开，单独储存在云数据库上并冗余以降低数据丢失的风险。同时，所有的访问请求都需要对应的电子签名，否则无法调用系统资源。这将通过自定义的权限认证系统来实现。

对于后期高并发环境下，运算模型负载过高的问题。我们预备缓存数据库功能，将论文种常见的行文错误进行特征编码后缓存。一旦再次遇到这样的问题，将直接使用缓存中的数据，而不需要再次带入运算模型进行重复运算，降低服务器负载。同时合理利用每个分布式计算端的资源，将任务最优调配，尽力确保每个计算资源不被浪费。若是在此情况下仍然出现计算资源紧张的问题，我们会考虑增加 GPU 与 CPU 服务端。

2.6 进度安排

- 2019-12-1 ~ 2019-12-15：技术路线讨论，主要功能确定。
- 2019-12-16 ~ 2019-12-31：项目开发文档（初版）、项目测试文档（初版）、技术研究报告（初版）、作品创新性分析报告（初版）写作，软件产品主要功能及对应的技术方案敲定；客户端软件界面设计与原型制作，客户端软件基础代码编写，服务端软件基础代码编写，开发服务器设施搭建；项目视频制作。
- 2020-1-6 ~ 2020-1-31：客户端设计深入讨论，后端代码框架完善，前端设计，前端框架完善。
- 2020-2-1 ~ 2020-3-23：客户端代码与前端代码完善，单元测试
- 2020-3-24 ~ 2020-4-16：服务端业务部署，客户端与服务端对接，集中测试；软件产品用户测试与后期微调。
- 2020-4-17 ~ 2020-4-23：项目开发文档（更新版）、项目测试文档（更新版）、技术研究报告（更新版）、作品创新性分析报告（更新版）写作，应用程序源

代码整理，项目视频制作（更新版）。

- 2020-4-24 ~ 2020-5-1：服务端业务优化；客户端扩展功能讨论敲定，客户端扩展功能界面设计与原型制作。
- 2020-5-2 ~ 2020-5-7：客户端扩展功能完善，服务端扩展业务逻辑添加；扩展业务逻辑微调、测试。
- 2020-5-8 ~ 2020-5-11：项目开发文档（最终版）、项目测试文档（最终版）、技术研究报告（最终版）、作品创新性分析报告（最终版）写作，应用程序源代码整理，项目视频制作（最终版）。项目介绍 PPT 制作，使用手册制作。《商业计划书》写作。

2.7 开发预算

类别	项目	单位	预算	合计	总计
服务端/计算端	租用训练用 GPU 服务器	1 月	2000	27200	35200
	项目用服务器采购	1 台	3200		
	云服务器租用	1 年	2400		
	云服务器流量费	1T/年	1200		
	普通 GPU 服务器采购	4 台	16000		
	云数据库租用	1 年	2400		
客户端	原型设计软件授权	2	200	2400	
	启发式评估费用	\	200		
	用户测试费用	\	2000		
项目环节	CASE 类软件授权	\	3200	5600	
	开发场地费	1 年	2400		

3 可行性分析

3.1 技术可行性分析

3.1.1 类似产品功能比较

市面上关于论文润色的产品，基于人工润色的、比较有代表性的是 editSpring；基于机器润色的有如 Stylewriter、WhiteSmoke 和 Grammarly Spell Checker 等等，以下分别将针对 editSpring 和 WhiteSmoke，与我方产品进行分析比较。但这些产品或多或少都面临着如下几个问题，

editSpring：

首先是成本问题。editSpring 分了不同的套餐种类，最便宜的套餐都是近 5000 起步，更贵的将近上万元。如果是论文篇幅较长，价格递增。而我方产品完全不需要如此高的成本。主要原因在于修改途径不同，editSpring 是由专家或经验编辑人工修改；而我方产品是由自动化智能修改，可以实现批量化 修改，因此成本会小很多。

其次是效率问题。editSpring 需要的投稿周期基本上是以周为单位，没有低于十天的套餐；而受益于计算机的运算速度，我方产品解析文章的速度以分钟为单位的，远快于人工修改。

WhiteSmoke：

首先是语言差异。WhiteSmoke 等一系列市面上的产品全是针对英文论文写作的，我们暂时没有找到针对中文写作的成熟的产品。

针对这种现象，我们分析其原因如下：

1. 学生普遍思维是所写的母语的论文没有必要再专门找陌生的第三方进行辅助修改，有什么问题常常是向导师、同学寻求帮助；

2. 一次论文润色的成本太高，动辄上千，对于自己熟悉的母语而言，这样做的成本不如直接去找自己的导师。

3. 论文润色的周期较长，占用论文写作时间。对于母语写作而言，专程花时间以高代价做一次润色并非必要。由此看来，即使市面上已经有了 WhiteSmoke 此类的软件，中文市场方面仍然有极大的发展前景。

其次是服务定位的问题。WhiteSmoke 的定位是对词语拼写、语法等的错误检查和指出，对标的范围是一篇文章的词语等比较微观的部分，而我方产品最主要的是对文章整体风格的把握，最终目的是在不改变原文意思的基础上使其更具有一般论文的格式规范，可以对一篇不成熟的文章进行一次到位的整体风格修正，这样的修改方式性价比更高。

由此，综合语言定位（中文）、服务定位（论文风格整塑）、服务效率及成本、市场空缺等多方面因素，我方产品具备良好的市场竞争力以及开发价值。

3.1.2 技术风险规避方法

在技术上该项目的软件产品所使用的技术，结合 NLP 领域当前的局限，难以对论文论证本身相关的信息做出判断与预测。所以需要将模型的处理焦点集中于论文的行文方式。所以，模型的学习的重点在于文段整体的风格的特征的识别与分类，而适当忽略对于论文的论证以及论文的内容的信息。这样，要求在模型训练方面选取的训练语料的内容必须多样化，具备有较为明显的特征。

3.1.3 产品易用性

一方面，我们产品的受众主要是国内各大高校的本科生、研究生以及导师，其中本科生、研究生的人数占到用户人数比例的 80%，这类用户在该领域不具备有较强的专业性。另一方面，快应用具备有明显的轻量化的特征，注重与场景深度结合。所以在前端设计中，我们崇尚极简，注重核心功能的稳定性，只保留少量且必须的辅助功能。在 UI 设计上，风格更贴近当代青年人的喜好与习惯。对于剩余 20%的用户，我们在用户首次使用时，计划加入导航功能，一步一步指导用户如何使用本产品，以此来适应于各种用户。同时，本产品也参考了同行同类产品的设计，参考了不同行不同类的风格设计，更符合当下用户的使用习惯以及使用场景。

3.2 资源可行性分析

3.2.1 项目开发成本

类别	项目	单位	预算	合计	总计
服务端/计算端	租用训练用 GPU 服务器	1 月	2000	27200	35200
	项目用服务器采购	1 台	3200		
	云服务器租用	1 年	2400		
	云服务器流量费	1T/年	1200		
	普通 GPU 服务器采购	4 台	16000		
	云数据库租用	1 年	2400		
客户端	原型设计软件授权	2	200	2400	
	启发式评估费用	\	200		
	用户测试费用	\	2000		
项目环节	CASE 类软件授权	\	3200	5600	
	开发场地费	1 年	2400		

3.2.2 后期维护成本

类别	项目	单位	预算	合计	总计
服务端/计算端	测试用服务器维护	1 台/年	200	6200	7400
	云服务器租用	1 年	2400		
	云服务器流量费	1T/年	1200		
	云数据库租用	1 年	2400		
客户端	软件维护费	1 年	1200	1200	

3.2.3 其他成本

待验证。

3.2.4 产品收益分析

待验证。

3.3 市场可行性分析

从当下市场行情发展来看,我们将对产品进行纵向比较与横向比较。具体上,本产品也将与其他能够实现类似功能的产品进行比较。

纵向比较

当下存在许多论文润色服务，但是或多或少都面临着以下几个问题：

1.只涉及句子为单位的查重类服务。

2.对基本句子结构只是简单的套用而不是在原句的基础上进行匹配。

3.基本无法实现对文章的语言风格的把握，进行行文上的有针对性的修改。

4.过于专业，导致成本过高。

横向比较

国外确实涌现了一大批类似文章润色的软件，如 Stylewriter、WhiteSmoke 和 Grammarly Spell Checker 等等，但是此类国外的软件只能实现对英文论文的修改和润色，对中文缺乏支持。另外，缺乏对于行文上的检测，专注于语法错误的纠正。

而对于中国广大的高校本科生、研究生以及所有的初涉论文写作的人群，他们的毕业论文中往往存在着许多的打字错误、句子不通顺、口语化表达的问题。这类问题不仅大大削弱了论文的专业性与严谨性,也直接影响着审阅者对于学术论文总体水平的评估。所以，他们对于一个可以对中文论文的行文进行评估，并给出合适的修改建议的的软件有着迫切需求。

从市场背景来看，当前尚未存在一款可以对用户提供的中文论文行文风格进行全面的书面化的指导的软件。而缺乏经验的用户在撰写论文时，在该领域，又面临着诸多明显的问题。所以这一款可以实现基本的错误检测与文章语言的书面化、正式化的软件宛如寒中送暖般地满足市场的需求。

4 需求分析

4.1 数据需求

4.1.1 静态数据

本项目的 BERT 预训练模型在预训练阶段需要使用大量的语料进行。而在微调（Fine Tuning）阶段，根据文本分类的应用场景，模型训练需要维基百科和亚马逊书评的去除英文字母、公式后的纯中文文本作为语料，我们将维基百科的数据集作为中立句子的数据集，将亚马逊书评文段作为消极和积极的数据集来进行监督训练。

而本项目的词语搭配的辅助 DNN 神经网络需要使用大量语料的所有语句的分词结果作为语料进行监督训练。

4.1.2 动态数据

本项目客户端的行文综合评分功能数据需要根据在 BERT 模型基础上 Fine Turing 的计算模型得出的不同句子情感倾向、文章的错字字数、不同顺的句子数量及其不通顺的程度来进行计算。

本项目服务端的缓存数据库需要文本预处理模块提供的分词分句以及预处理结果，此预处理模块是先对文章各个段落实施自定义的分句算法，然后对每个句子调用百度的自然语言处理模型进行文本纠错以及句子通顺度判断。数据库会存储经过预处理的文本结果。对于调用 BERT 进行处理的语料数据库也会保存以加快相同语料的处理速度。

4.1.3 数据词典

- 纯中文文本=[中文学术论文，维基百科（中文）文本]
- 纯中文语句=句子+语气
- 分类标签=1..3

- 分词结果=词语+词性+（成分）
- 词语=2{汉字}8
- 词性=0..23
- 成分=0..5
- 数据整理功能模块的最终结果数据=句子编号+错误类型+（错误描述）+（修改建议）
- 句子编号=000…999
- 错误类型=1…3
- 错误描述=4{字母}12
- 修改建议=1{汉字}32
- 分句模块处理后生成的中间数据=1{词语}32
- 启发式句式处理模块最终数据=句子编号+错误类型
- BERT 预训练模型处理最终数据=句子编号+口语化程度+严谨性程度+（修改建议）
- 口语化程度=0.00..1.00
- 严谨程度=0.00..1.00

4.1.4 数据采集

维基百科（中文）文本与中文学术论文将通过互联网获取，其中去除英文字母、中文的纯中文文本语料数据将由使用 Python 编写的工具自动处理后批量生成。纯中文文本按句子拆分的结果数据将由服务端分句预处理模块自动处理后批量生成。而对于纯中文文本在文本分类应用场景下的监督学习的语料的分类标签由人工编写。

维基百科（中文）的文本以及亚马逊书评的文本将通过互联网获取，其中去除英文字母、中文的纯中文文本语料数据将由使用 Python 编写的工具自动处理后批量生成。每篇文章的分句结果将在预处理时通过自定义的分句算法完成，DNN 语言模型的运算结果以及错误文本的检测结果同样会在预处理时获取。句子情感倾向将会通过 Fine-Tuning 后的 BERT 模型获取。

4.2 功能需求

4.2.1 功能模块

表 核心功能模块描述

功能模块	功能	功能描述	优先级
权限管理	账号认证	认证账号是否通过	高
	账号权限管理	分配、撤销账号权限	高
	身份验证	访问系统资源时验证身份	高
	账号角色管理	管理系统内所有账号及相应角色	中
任务管理模块	子任务创建	创建一个子任务，将之放入等待队列	高
	子任务自动预处理	创建子任务后自动开始拆分文章结构	高
	子任务自动预分析	预处理之后自动开始预分析	高
	子任务结果查询	根据任务 id 获取任务结果	高
	子任务自动批处理任务转化	1. 将等待超时的任务加入批处理任务 2. 等待任务数超过阈值后加入批处理任务	高
子服务器管理	子服务器自动注册	注册新加入的子服务器	中
	子服务器签证更新	定期更新子服务器的签证	中
	子服务器自动护照管理	管理子服务器的护照	中
批处理任务模块	批处理任务自动调度	对等待队列中的批处理任务调度至对应子服务器	高
	批处理任务计算量估量	估计批处理任务的计算量，并据此设立优先级	高
	批处理任务自动拆分	将批处理任务拆分为细粒度的子任务	高

	批处理任务结果处理	使用 BERT 模型计算任务并返回结果	高

4.2.2 用例规约

表 1 句子划分模块用例规约

用例名称	句子划分模块
功能简述	此模块用于把一个段落划分为多个独立完整的句子
用例编号	UT-001
执行者	后端
前置条件	无
后置条件	1. 预处好的句子
涉众利益	用户：划分是否出现偏差导致最终呈现形式与原文有异
基本路径	1. 权限认证 2. 上传文件 3. 结果获取请求
扩展路径	
字段列表	
设计规则	1. 保证检测的准确度 2. 保证在段落分句完成后进行检测
未解决的问题	
备注	

表 2 段落划分模块用例规约

用例名称	段落划分模块
功能简述	此模块用于把此模块用于把一篇文章划分为多个段落
用例编号	UT-002
执行者	后端
前置条件	无
后置条件	1. 划分好的段落的文章
涉众利益	用户：划分是否出现偏差导致最终呈现形式与原文有异
基本路径	1. 权限认证 2. 上传文件 3. 结果获取请求
扩展路径	
字段列表	
设计规则	1. 保证检测的准确度
未解决的问题	

备注	

表 3　DNN 语言模型处理模块用例规约

用例名称	DNN 语言模型处理
功能简述	对于用户预处理文本进行句子粒度的分析，获取通顺程度
用例编号	UC004
执行者	用户、后端
前置条件	1. 权限认证通过 2. 文件已经上传 3. 已完成段落分句
后置条件	1. 已经获取句子通顺程度 2. 返回处理结果
涉众利益	用户：软件系统能够检测出出句子中的通顺问题
基本路径	1. 权限认证 2. 上传文件 3. 结果获取请求
扩展路径	
字段列表	
设计规则	2. 保证检测的准确度 3. 保证在段落分句完成后进行检测
未解决的问题	
备注	

表 4　文本纠错模块用例规约

用例名称	文本纠错
功能简述	对用户输入文本的中的错字（打字错误、错别字）进行检测
用例编号	UT-004
执行者	后端
前置条件	1. 前端已经上传了处理文本 2. 前端针对文本发出了处理请求
后置条件	1. 文本纠结果成功保存
涉众利益	用户：软件系统能够检测出出论文中的打字错误与错别字的情况
基本路径	1. 输入文本 2. 模块调用
扩展路径	

字段列表	
设计规则	1. 确保异常能否正确处理
未解决的问题	
备注	

表 5 任务监听模块用例规约

用例名称	任务监听
功能简述	监听一个 task 在队列中的等待时间，当超出阈值后立即加入一个批处理任务并等待计算端索取
用例编号	UT-005
执行者	后端
前置条件	1. 任务已经创建 2. 任务在队列中等待超时
后置条件	1. 任务成功加入批处理任务
涉众利益	用户：软件系统能够检测出出论文中的打字错误与错别字的情况
基本路径	1. 权限认证 2. 上传文件 3. 新建任务 4. 等待至任务超时
扩展路径	
字段列表	
设计规则	保证超时机制能被测试
未解决的问题	
备注	

表 6 批处理任务分配模块用例规约

用例名称	批处理任务分配
功能简述	把一个批处理任务序列线程安全地分配给计算端
用例编号	UT-006
执行者	后端
前置条件	1. 批处理任务已经创建 2. 计算端发起调用请求
后置条件	1. 任务成功分配
涉众利益	计算端：能否正确获取一个批处理任务
基本路径	1. 权限认证

	2. 上传文件 3. 新建任务 4. 等待至任务超时
扩展路径	
字段列表	
设计规则	保证超时机制能被测试
未解决的问题	
备注	

表 7 批处理任务监听模块用例规约

用例名称	批处理任务监听
功能简述	监听每个被前端调用的批处理任务的处理时间，在超过阈值后解锁，并重新放回就绪队列等待处理
用例编号	UT-007
执行者	后端
前置条件	1. 批处理任务已经创建 2. 计算端发起调用请求 3. 批处理任务等待超时
后置条件	1. 任务成功解锁并重新就绪
涉众利益	用户：任务能否按时解析
基本路径	1. 权限认证 2. 上传文件 3. 新建任务 4. 新建批处理任务 5. 锁住批处理任务 6. 等待至超时
扩展路径	
字段列表	
设计规则	保证超时机制能被测试
未解决的问题	
备注	

表 8 模块用例规约

用例名称	结果构造
功能简述	根据计算结果，生成用于返回前端的结果

用例编号	UT-008
执行者	后端
前置条件	1. 所有计算已经执行完毕
后置条件	1. 结果成功生成
涉众利益	用户：能否获取结果
基本路径	1. 权限认证 2. 上传文件 3. 新建任务 4. 新建批处理任务 5. 计算端发送完成标记信号 6. 发送结果获取请求
扩展路径	
字段列表	
设计规则	保证所有结果都被测试
未解决的问题	
备注	

表 9 模块用例规约

用例名称	文件接收与解析
功能简述	此模块用于接收用户上传的文件并将其解析以及持久化
用例编号	UT-009
执行者	后端
前置条件	无
后置条件	1. 文件成功解析 2. 文件成功持久化
涉众利益	用户：能否顺利上传文件
基本路径	1. 权限认证 2. 上传文件
扩展路径	
字段列表	
设计规则	
未解决的问题	
备注	

表 10 任务创建模块用例规约

用例名称	任务创建
功能简述	在用户提交创建任务请求之后进行预处理
用例编号	UT-010
执行者	后端
前置条件	1. 所有计算已经执行完毕
后置条件	1. 结果成功生成
涉众利益	用户：能否顺利创建任务
基本路径	1. 权限认证 2. 上传文件 3. 新建任务
扩展路径	
字段列表	
设计规则	
未解决的问题	
备注	

表 11 BERT 计算模块用例规约

用例名称	查看修改结果
功能简述	在 Fine Tuning 后的 BERT 模型上计算句子的口语化倾向、情感倾向、严谨程度
用例编号	UT-011
执行者	后端
前置条件	无
后置条件	1. 正确处理句子并返回相应结果
涉众利益	用户：能否得到结果
基本路径	1. 向模型输入 100 个句子
扩展路径	
字段列表	
设计规则	
未解决的问题	
备注	

表 12 任务结果查询模块用例规约

用例名称	查看修改结果
功能简述	可视化软件系统对于论文的所有的检测记录、自动修改记录与用户的手

	工修改记录，并附上可视化报告
用例编号	UT-012
执行者	用户、后端
前置条件	1. 成功执行了错字修改、句式分析、语义分析过程
后置条件	1. 可视化报告已生成 2. 修改完成的论文文本已保存
涉众利益	用户：软件系统能够给出该篇论文所有的修改过程的记录以及修改总结报告
基本路径	1. 权限认证 2. 文件上传 3. 文件处理 4. 获取处理结果 5. 查看修改过程记录 6. 查看修改报告 7. 保存本次会话的所有信息
扩展路径	1. 删除本次会话的所有信息
字段列表	
设计规则	
未解决的问题	
备注	

表 13 查看评价详情用例规约

用例名称	查看评价详情
功能简述	用户查看软件系统对论文行文打出的综合分数、对应档次以及各个指标的详情
用例编号	UT-013
执行者	用户
前置条件	1. 用户已经上传了有效的论文文本 2. 软件系统成功对论文文本进行所有分析并计算出论文行文的综合分数
后置条件	1. 保存评价结果
涉众利益	用户：软件系统能够对上传的文本进行打分，给出论文在行文维度上的综合分数
基本路径	1. 权限认证 2. 文件上传

	3. 文件分析 4. 查看评价结果详情 5. 退出查看评价结果界面、进入主页
扩展路径	1. 查看软件系统的分析报告
字段列表	
设计规则	
未解决的问题	
备注	

表 14 权限管理用例规约

用例名称	权限管理
功能简述	控制用户对系统资源的访问
用例编号	SST-001
执行者	后端
前置条件	
后置条件	
涉众利益	用户：是否正确访问系统
基本路径	1. 账号注册 2. 账号登录 3. 访问系统资源 4. 账号登出 5. 访问系统资源
扩展路径	
字段列表	
设计规则	对每个系统模块在不同权限下都要访问
未解决的问题	
备注	

表 15 任务管理用例规约

用例名称	任务管理
功能简述	管理任务与批处理任务，包括其创建和调度
用例编号	SST-002
执行者	后端
前置条件	
后置条件	

涉众利益	用户：是否顺利创建任务、得到正确处理结果
基本路径	1. 登录系统 2. 上传文件 3. 新建任务 4. 获取结果
扩展路径	
字段列表	
设计规则	保证可以测试到超时机制
未解决的问题	
备注	

表 16 分布式计算用例规约

用例名称	分布式计算
功能简述	对用户输入文本的中的口语化成分进行检测；对句子的严谨性进行评估；对不严谨的句子自动生成修改指导方案
用例编号	SST-003
执行者	后端、计算端
前置条件	1. 文件上传成功 2. 处理请求发送成功 3. 预处理成功 4. GPU 服务器发送了任务索取请求
后置条件	1. 完成了语义检测并返回结果
涉众利益	用户：软件系统能够检测出论文中的口语化成分并标记；能够检测出不严谨的句子并生成合理的指导方案
基本路径	1. 权限认证 2. 前端发送处理请求 3. 文本预处理 4. 分布式 GPU 计算端发送任务索取请求 5. 分布式 GPU 计算端返回任务处理结果 6. 主服务端整合包装处理结果 7. 前端发送结果查询请求 8. 主服务端返回结果
扩展路径	
字段列表	
设计规则	保证高并发的情况被

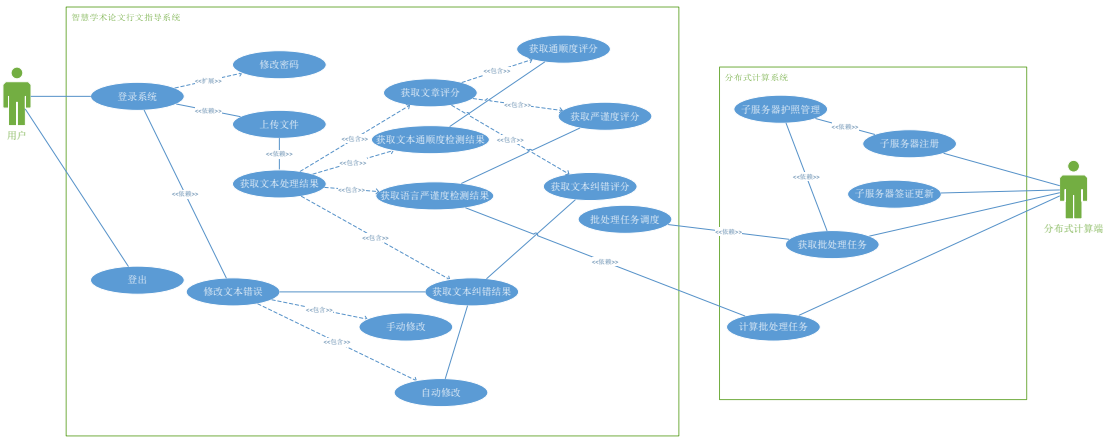
未解决的问题	
备注	

表 17　文件管理用例规约

用例名称	文件管理
功能简述	准确接收前端上传的文本并保存至数据库等待处理
用例编号	SST-004
执行者	后端
前置条件	无
后置条件	已经持久化文件
涉众利益	用户：文件可以被记录 前端：能够获取该文件的 id
基本路径	1. 权限认证 2. 文本上传
扩展路径	
字段列表	
设计规则	1. 文本按照 doc 或 docx 的形式整个上传
未解决的问题	
备注	

4.2.3 功能用例图

图 功能用例



4.3 性能需求

4.3.1 时间特性

- 对于登录登出、上传文件的操作要求在 3s 内响应
- 每篇文章的处理时间不得超过 40s
- GPU 计算端向主服务器索取任务的过程耗时不应超过 2s
- 主服务器对每个任务在 GPU 计算端的等待时间不应超过 15s
- 每个批处理任务在就绪队列中的等待时间不应超过 10s
- 服务器支持 24h 不间断请求响应

4.3.2 适应性

- 当某一服务器发生故障时需保证没有因此丢失的任务
- 要保证良好的功能接口拓展性以便后续功能的添加
- 确保新加入的计算端同样可以最大限度地索取任务
- 对于更高运算能力的 GPU 保证分配更难以处理的任务

4.4 界面需求

- 坚持以用户体验为中心设计原则，界面直观、简洁，操作方便快捷，用户接触软件后对界面上对应的功能一目了然、不需要太多培训就可以方便使用本应用系统。
- 保持字体及颜色一致，避免一套主题出现多个字体。
- 不可修改的字段，统一用灰色文字显示。
- 保持页面内元素对齐方式的一致，如无特殊情况应避免同一页面出现多种数据对齐方式。
- 使用一致的标记、标准缩写和颜色，显示信息的含义应该非常明确，用户不必再参考其它信息源。
- 在进行 UI 设计时需要充分考虑布局的合理化问题，遵循用户的浏览、操作习惯，避免常用功能按钮排列过于分散。多做“减法”运算，将不常用的功能区块隐藏，以保持界面的简洁，使用户专注于主要业务操作流程，有利于提高软件的易用性及可用性。
- 系统响应时间应该适中，响应时间过长，用户就会感到不安和沮丧，而响应时间过快也会影响到用户的操作节奏，并可能导致错误。

4.5 接口需求

4.5.1 硬件接口

目前核心功能暂无硬件接口要求。

4.5.2 软件接口

主服务端对于百度 NLP 自然语言处理平台 API 接口调用，接口以及函数如下：

接口名称	简要描述	调用函数
词法分析	分词、词性标注、专名	lexer(text,options)

	识别	
词向量表示	查询词汇的词向量，实现文本的可计算	wordEmbedding(word,options)
DNN 语言模型	判断一句话是否符合语言表达习惯，输出分词结果并给出每个词在句子中的概率值	dnnlmCn(text, options)
文本纠错	识别文本中有错误的片段，进行错误提示并给出正确的建议文本内容	ecnet(text, options)

4.6 其他需求

- 保证操作的简便性以及合理性
- 保证系统是便于维护的
- 保证服务器迁移时数据的完整性
- 保证用户信息的安全性
- 保证调用系统接口时权限管理的严格性

5 概要设计

5.1 处理流程

处理流程分为核心功能逻辑流程图与数据流程图展示。其中，为了直观、准确，核心功能逻辑流程图采用 UML 活动图的形式呈现。

5.1.1 核心功能逻辑流程图

图 5-1 认证阶段

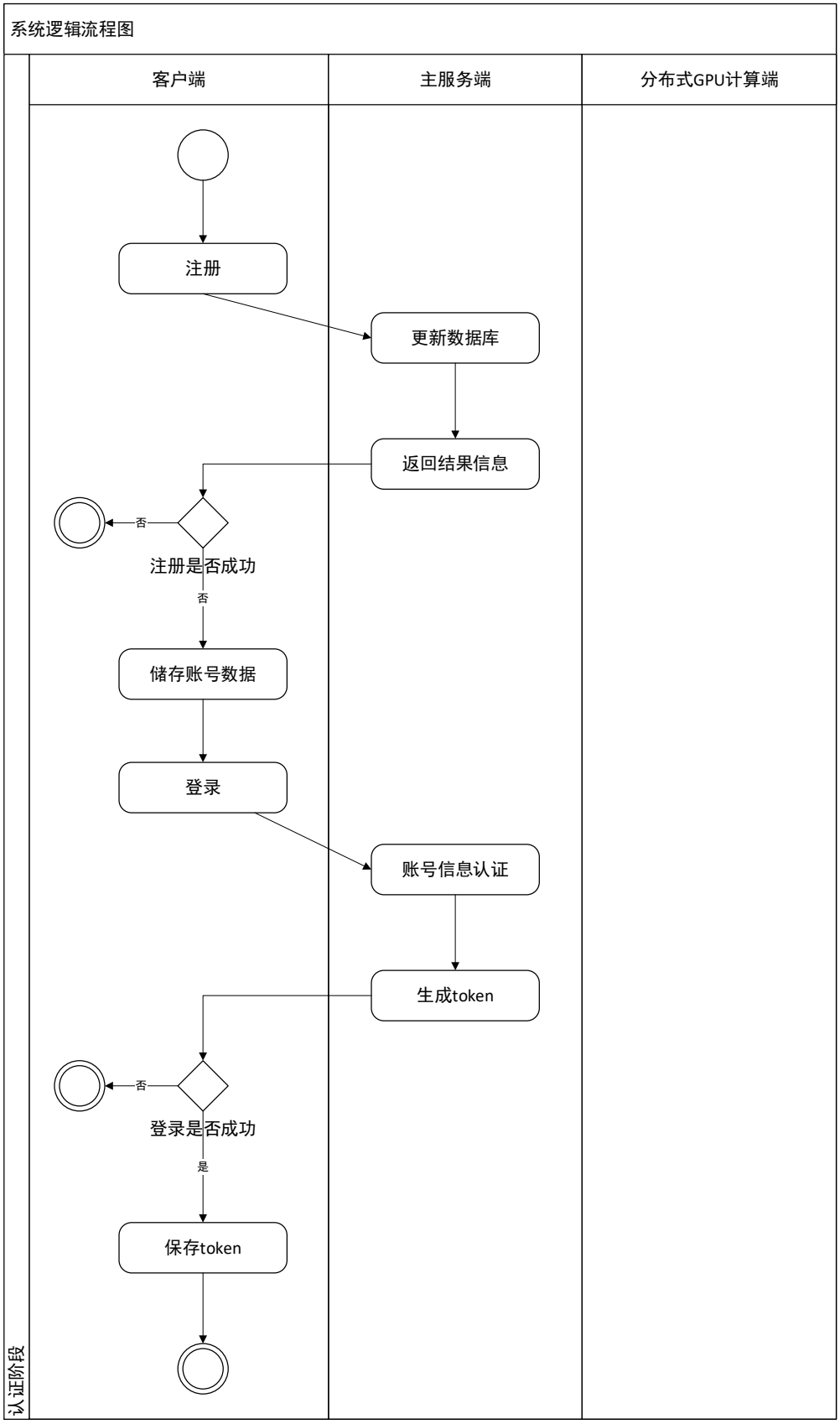


图 5-2 验证阶段

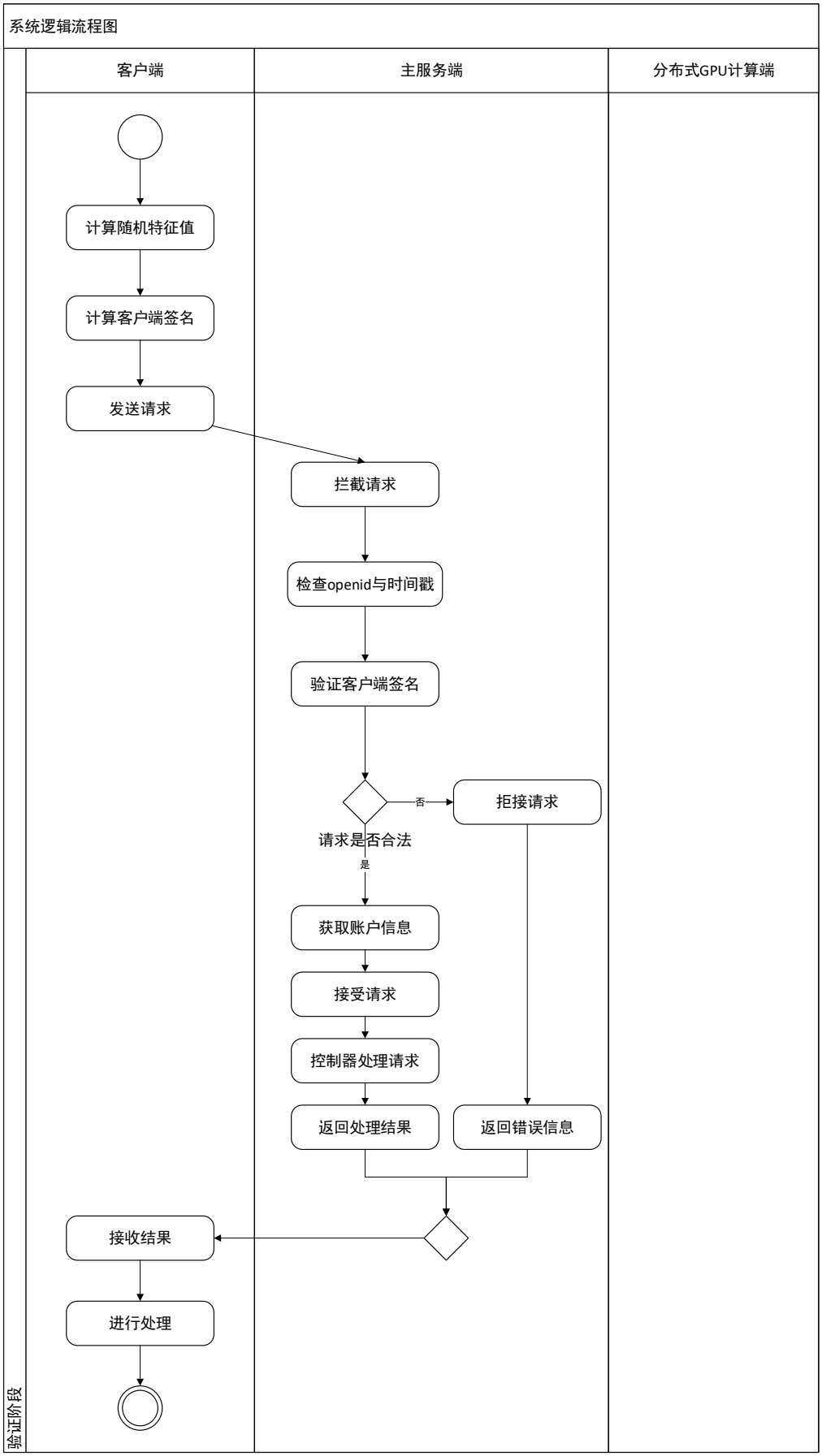

系统逻辑流程图
客户端
主服务端
分布式GPU计算端
计算随机特征值
计算客户端签名
发送请求
拦截请求
检查openid与时间戳
验证客户端签名
请求是否合法
否
拒接请求
是
获取账户信息
接受请求
控制器处理请求
返回处理结果
返回错误信息
接收结果
进行处理
验证阶段

图 5-3 子任务创建与预处理阶段

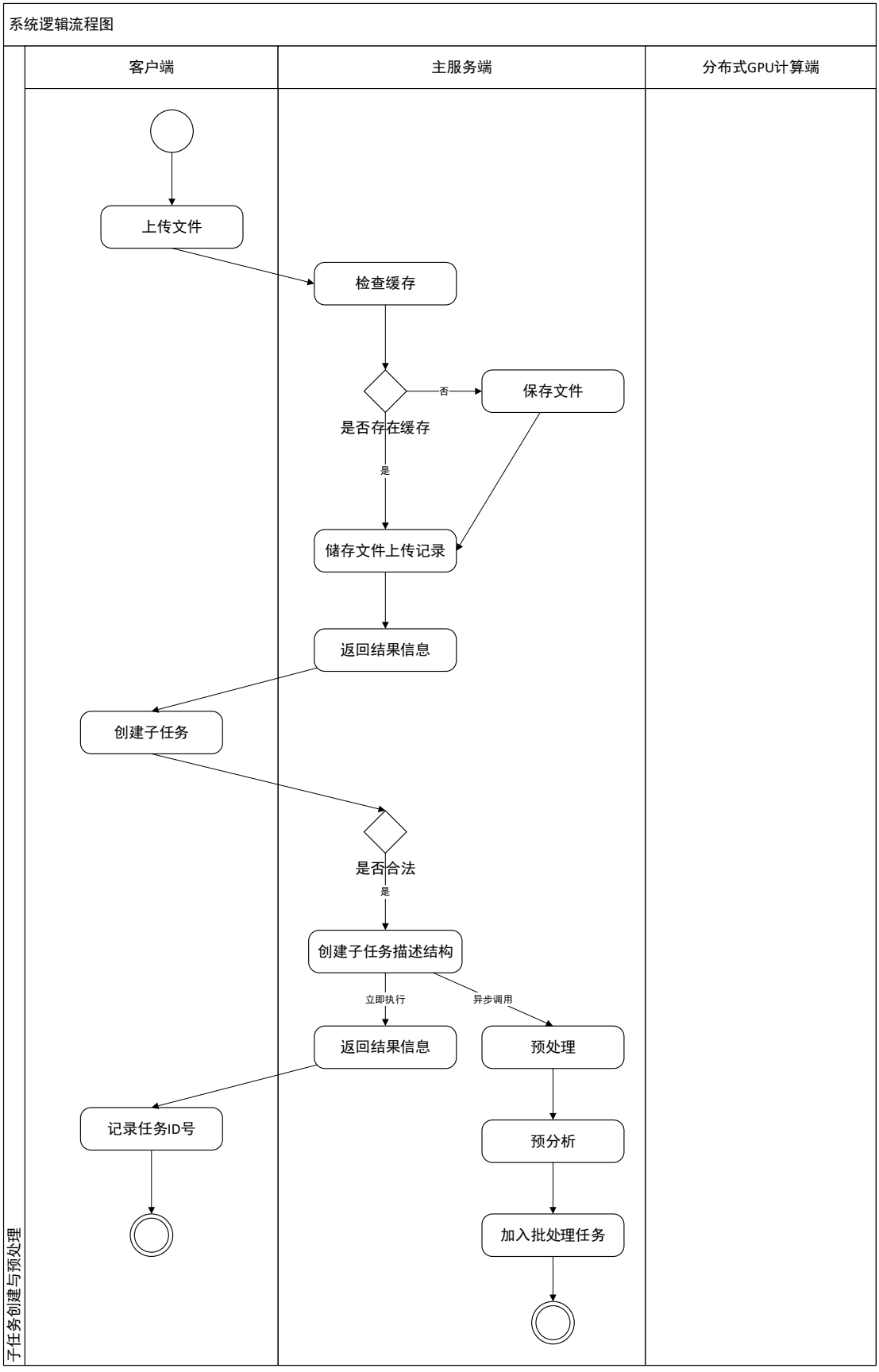


图 5-4 批处理阶段

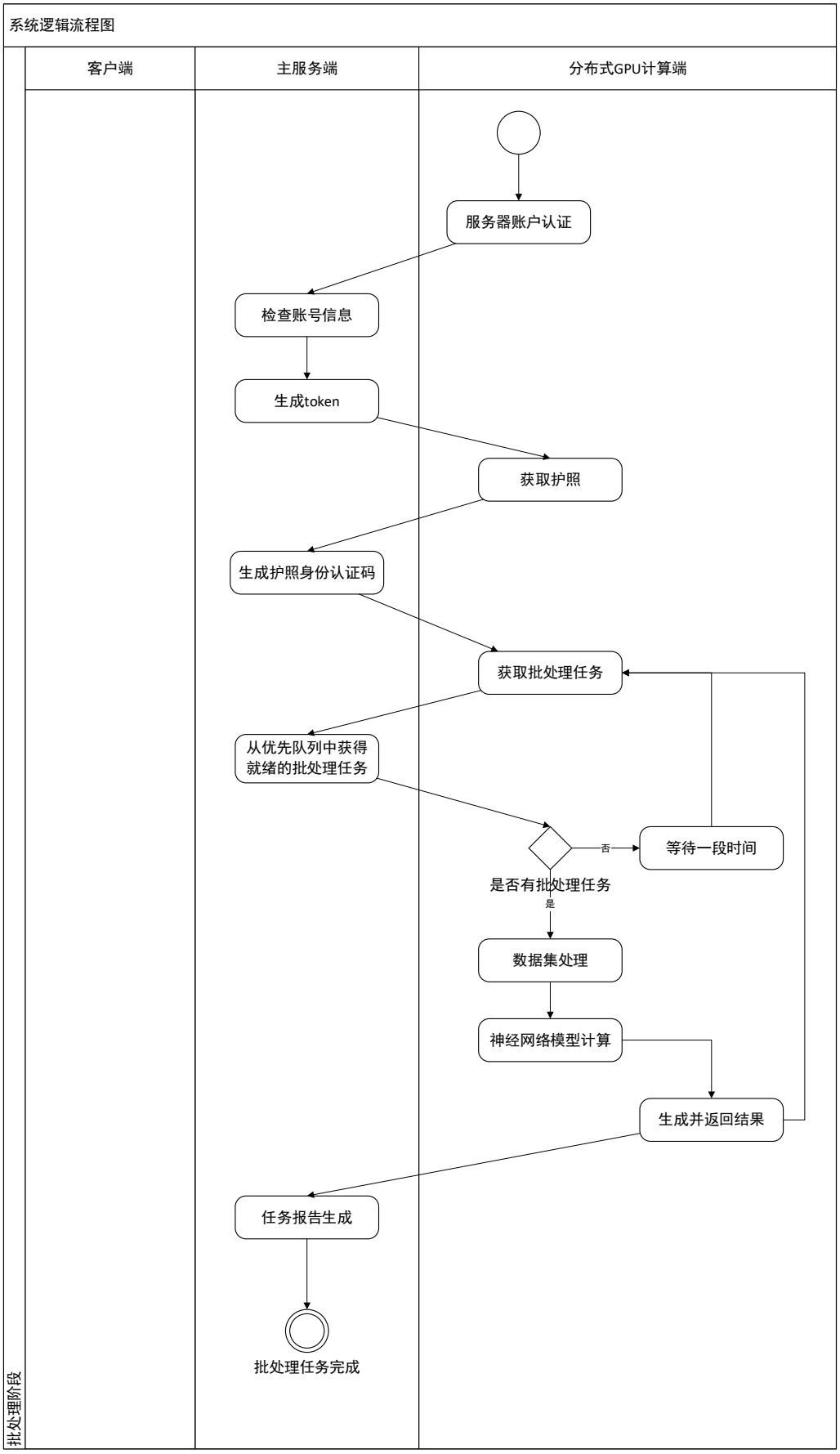
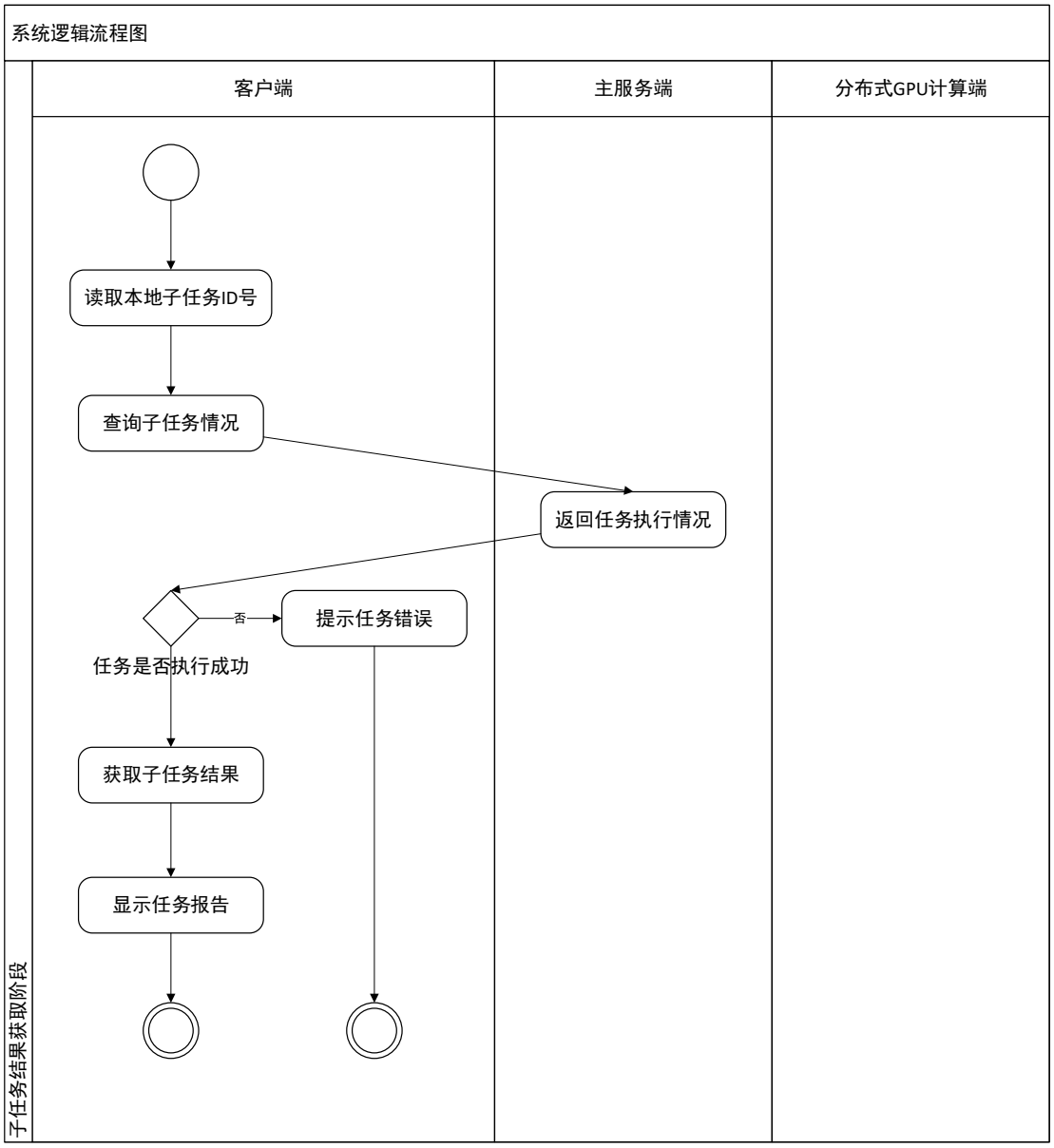

系统逻辑流程图
客户端
主服务端
分布式GPU计算端
批处理阶段
服务器账户认证
检查账号信息
生成token
获取护照
生成护照身份认证码
获取批处理任务
从优先队列中获得
就绪的批处理任务
否
等待一段时间
是否有批处理任务
是
数据集处理
神经网络模型计算
生成并返回结果
任务报告生成
批处理任务完成

图 5-5 子任务结果获取阶段



5.1.2 核心功能数据流程图

图 5-6 子任务创建阶段

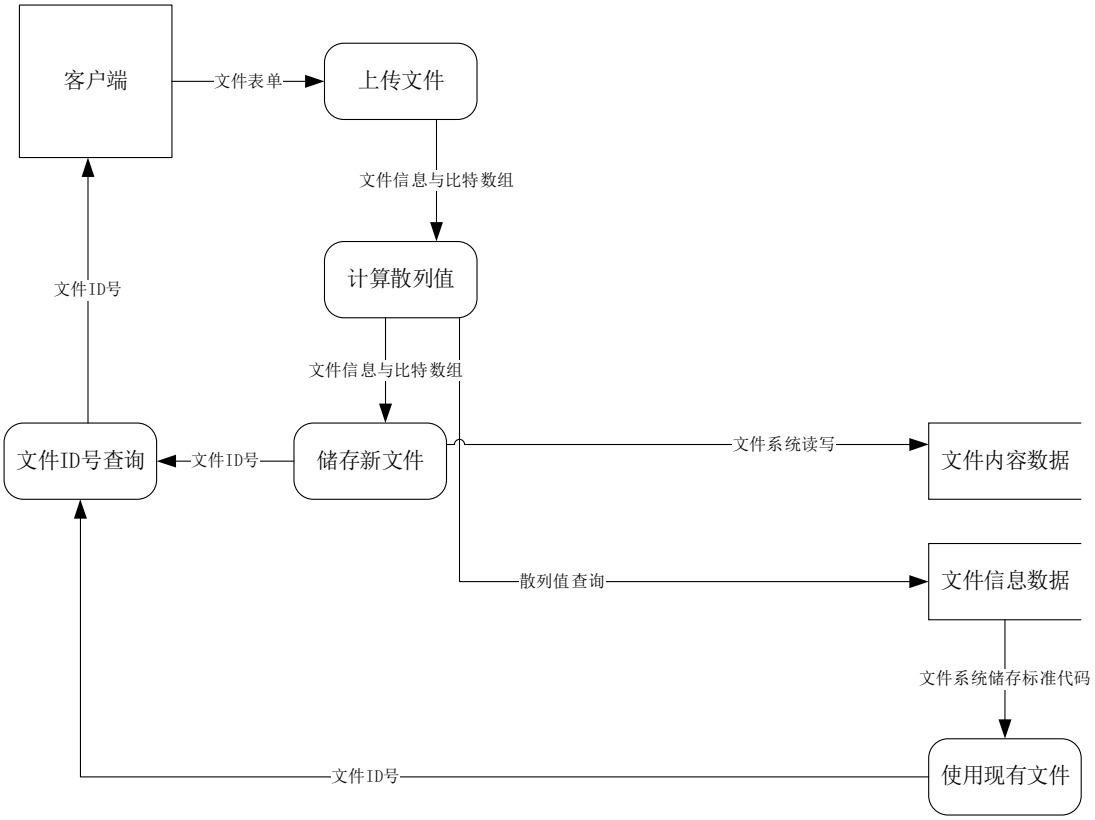


图 5-7 子任务预处理阶段

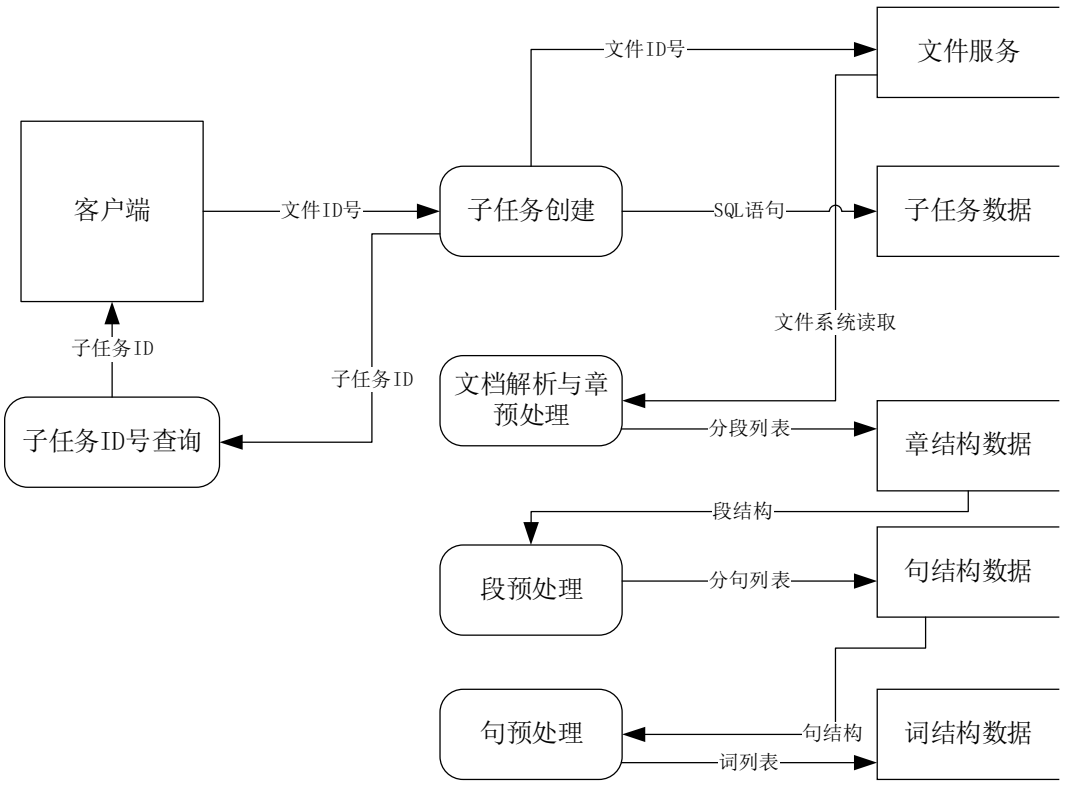


图 5-8 子任务预分析与批处理阶段

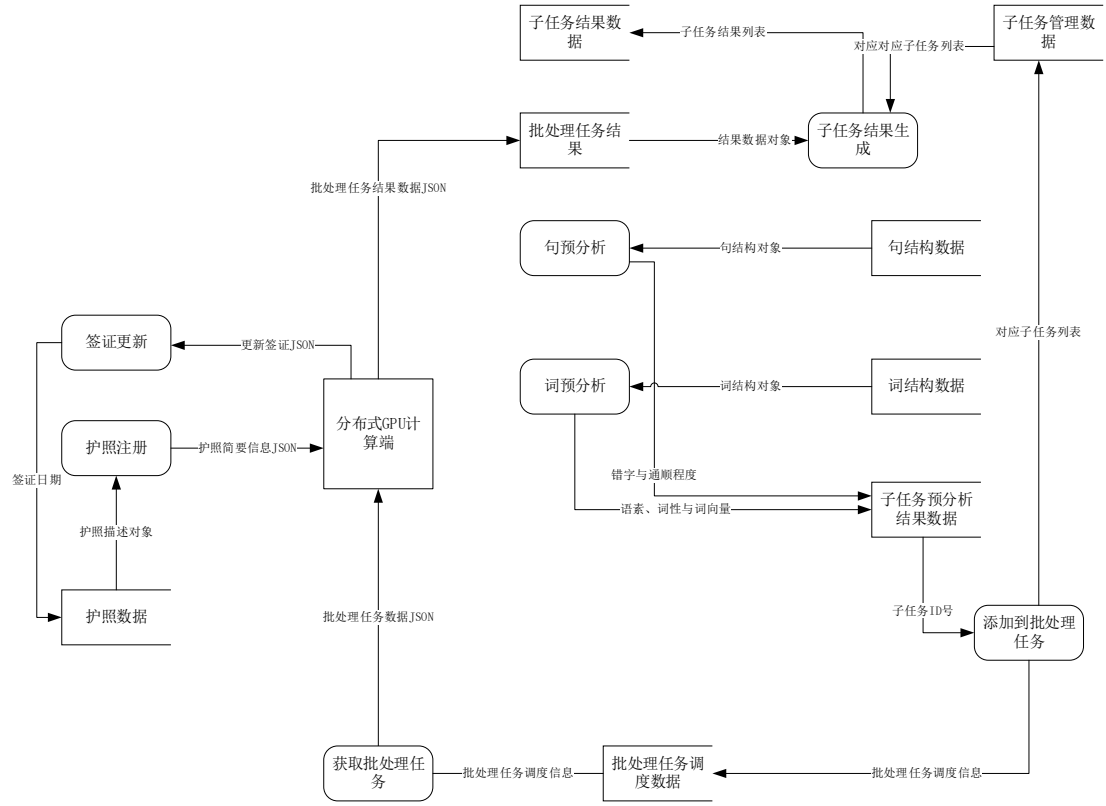
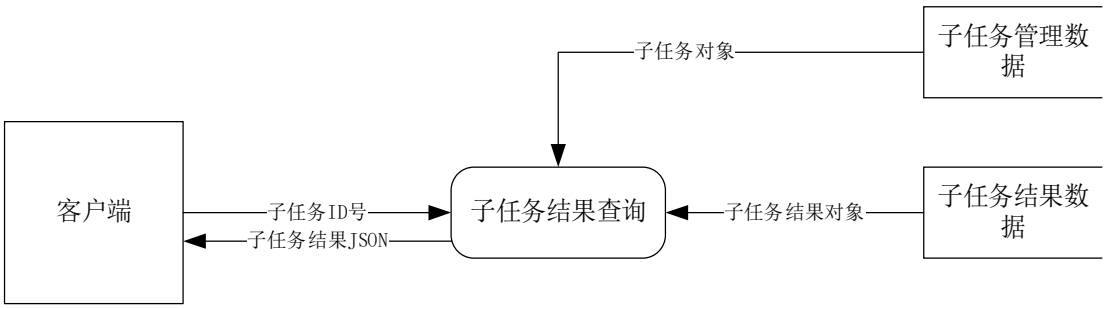


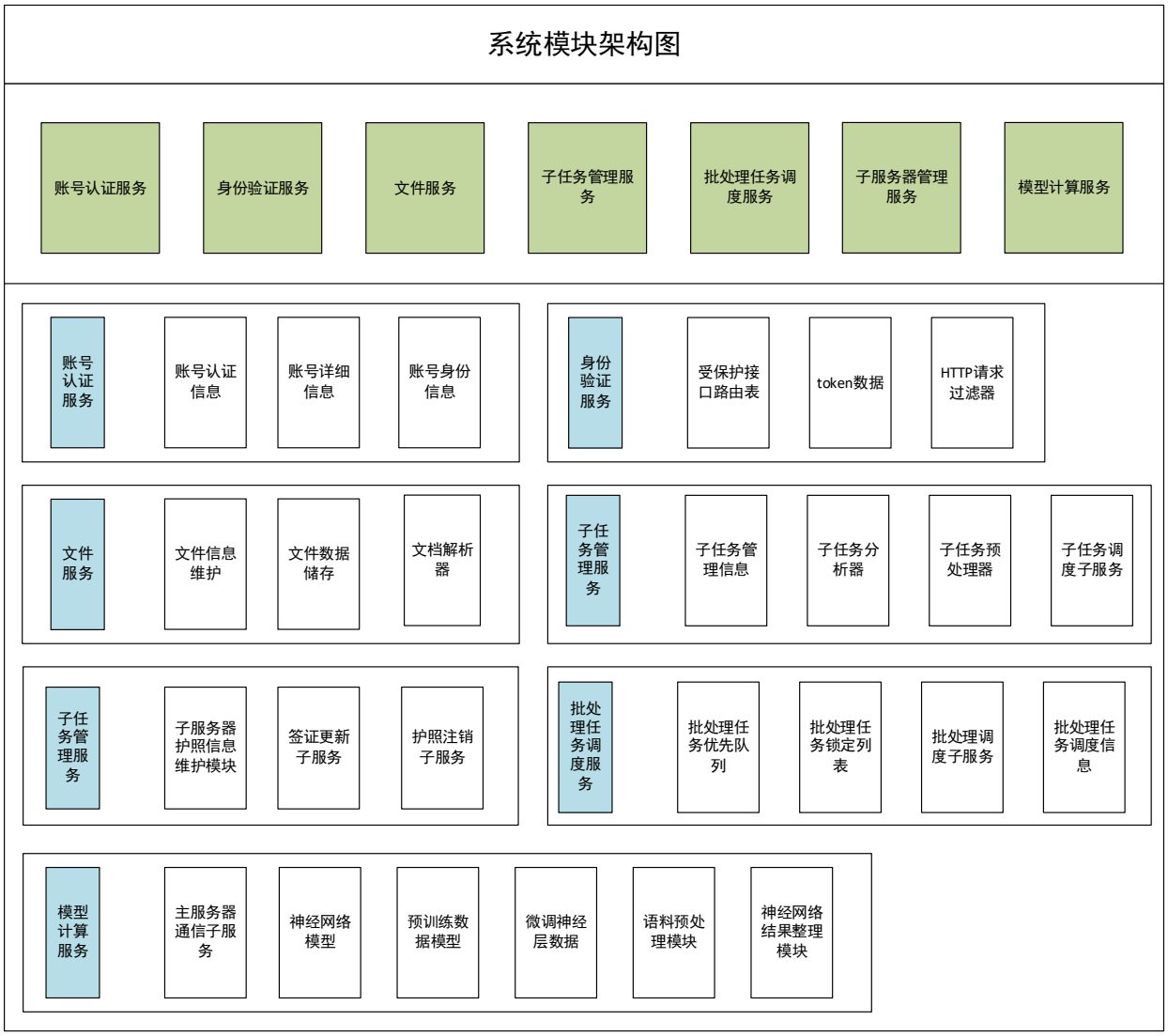
图 5-9 子任务结果查询阶段



5.2 总体结构设计

5.2.1 系统模块架构图

图 5-10 系统模块架构图



5.2.2 系统整体概述

软件系统（主服务端、计算端、客户端）由账号认证服务、身份验证服务、文件服务、子任务管理服务、批处理任务调度服务、子服务器管理服务、模型计算服务组成。

其中，主服务端与客户端互为表里，相辅相成。

客户端（快应用端）是系统的主要输入来源，而主服务端是系统的主要服务承担者。而计算端是系统的深度学习模型的计算任务的承担者，它从主服务端获

得批处理任务，并启动模型对批处理任务的数据进行计算，并为主服务端提供计算结果。

计算端具备分布式的特点，系统的所有深度学习模型的计算任务可以由多个正确部署的计算端并行分担。

5.2.3 系统服务概述

1. 账号认证服务：承担系统的账号注册与认证任务，包括依据 openid 与预设密码创建新的账号、检验 openid 与密码是否正确、记录登录是的客户端代码（clientCode）给身份验证服务提供凭据、生成 token 用于客户端计算客户端签名。该服务掌管着数据库中关于账号认证信息、账号详细信息、账号身份信息的创建与更新。

2. 身份验证服务：承担系统的对于受保护的 API 接口的请求的身份检验任务，这些任务细分可包括对于记录受保护的 API 接口的 URL 地址（路由表）、拦截对于受保护的 API 接口的访问请求并进行过滤、对于客户端依据 token 计算出的客户端签名（signed）进行核验、为接口控制器提供用户身份信息。

3. 文件服务：承担系统对于文件相关的任务，该服务维护着软件系统所掌握的文件的信息与文件数据。除却文件上传与文件下载的这些基本功能外，该服务也承担者对于 Word、Excel 文档的解析任务。另外，该服务具备有计算文件散列值来防止重复上传的功能。

4. 子任务管理服务：子任务管理服务是系统较为核心的服务，用户创建的子任务（对于论文进行检查）由该服务接收。该服务在接收到合法的子任务创建请求后，将创建子任务管理对象与子任务结果对象。在子任务管理对象成功创建、文档正确解析的前提下，子任务管理服务将对子任务的数据进行预处理与预分析。该服务也负责将经过预处理的子任务交给批处理调度服务。

5. 批处理任务调度服务：该服务是系统的核心服务，并且具备由高度自动化的特点。该服务的承担的核心任务是将子任务整合成批处理任务，并估计批处理任务的计算量，为进行模型计算服务的对于批处理任务的计算做铺垫。另

外，批处理任务调度服务维护着批处理任务的优先队列与锁定队列，决定批处理任务的调度优先级与调度锁定状态。该服务为模型计算服务提供关键的接口。

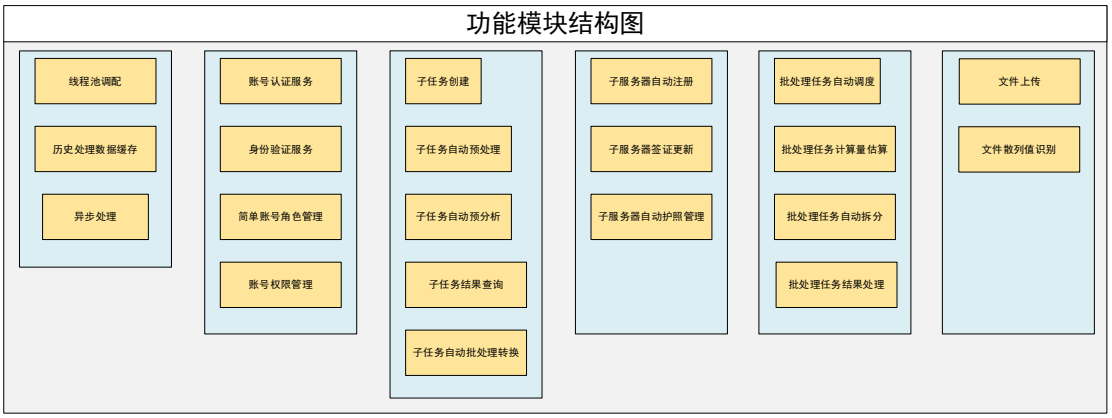
6. 子服务器管理服务：该服务维系着主服务端与分布式 GPU 计算端的关系。它实际上管理着计算端的任务分配与状态，并为计算端提供注册并分配护照的功能。该服务也负责确认计算端的在线状态，为计算端更新签证（刷新在线状态），及时注销过期的子服务器护照、解锁被计算端占用且被其未完成的批处理任务。另外，该服务也承担由于计算端的批处理任务的传输功能。

7. 模型计算服务组成：模型计算服务承担基于神经网络的计算任务，模型计算服务主要部署在计算端。它安装了神经网络模型，也是预训练数据与模型微调数据的实际使用者。该服务输入批处理任务数据，输出深度学习神经网络预训练模型的预测结果。另外，模型计算服务承担者计算端与主服务端的全部通讯功能。

5.3 功能设计

5.3.1 功能模块结构图

图 5-11 功能模块架构图



5.3.2 系统功能模块概述

1. 线程池调配功能：创建与维护主服务端线程池，并为预处理与预分析任务分配线程，使得预处理器与预分析器具备有多线程并行处理的能力。（该功能集

成在 Spring 框架中）

2. 历史处理数据缓存：缓存预处理器、预分析器、模型计算服务与文件服务的处理结果，防止计算浪费。其中，针对预处理器提供词粒度的缓存服务，针对预分析器与模型计算服务提供句粒度的缓存服务。历史数据缓存功能极大地加速了主服务端与计算端对于相同数据的二次处理速度，大大减少了主服务端与计算端的计算浪费现象。另外，针对文件服务的缓存功能，有效地减少了主服务器的空间占用。（该功能模块分布在系统的各个服务中，不以单独的模块出现）

3. 异步处理功能提供创建子任务时的异步处理机制的支持，使得快应用端在创建子任务请求后能够快速的到主服务端快速反馈，减少了快应用端的卡顿，杜绝创建任务请求被预处理与预分析阻塞。而与之相对应，预处理器与分析器得以在异步运行的条件下的多线程优势得到充分的发挥。（该功能模块集成在 Spring 框架中）

4. 账号认证（功能模块）：提供系统的账号注册与认证功能，包括依据 openid 与预设密码创建新的账号、检验 openid 与密码是否正确、记录登录是的客户端代码（clientCode）给身份验证服务提供凭据、生成 token 用于客户端计算客户端签名。该服务掌管着数据库中关于账号认证信息、账号详细信息、账号身份信息的创建与更新。（核心功能）

5. 身份验证（功能模块）：提供系统的对于受保护的 API 接口的请求的身份检验功能，这些功能细分可包括对于记录受保护的 API 接口的 URL 地址（路由表）、拦截对于受保护的 API 接口的访问请求并进行过滤、对于客户端依据 token 计算出的客户端签名（signed）进行核验、为接口控制器提供用户身份信息。

6. 账号角色管理功能：提供账号的角色的管理与区分功能，目前主要用于区分子服务器账号与普通用户账号，为子服务器管理功能模块提供依据。（核心功能，但是该功能模块分布在系统的各个服务中）

7. 账号权限管理功能：使得系统具备用对不同的账号能够授予不同的 API 接口使用权的能力。（非核心功能）

8. 子任务创建功能模块：接受客户端的发来的子任务创建请求，并创建子任务管理结构。（核心功能模块）

9. 子任务自动预处理功能模块：自动对已经创建成功的子任务进行章结构、段结构、句结构的预处理，涉及在异步条件下在利用线程池获得空闲线程并启动预处理器。（核心功能模块）

10. 子任务自动预分析功能模块：自动对已经经过与处理的子任务进行基于词性、词向量、语句通顺程度的分析，涉及在异步条件下在利用线程池获得空闲线程并启动预分析器。（核心功能模块）

11. 子任务结果查询功能：为客户端（快应用端）提供子任务的状态（是否完成），并返回相应已完成的子任务的处理结果。（核心功能，集成在子任务管理服务中，并非单独的模块）

12. 子任务自动批处理转换功能：由于同一时间段可以有多个子任务预处理与预分析完成，该功能自动地将这短时间内的多个子任务合并成为批处理任务提供给计算端处理。（核心功能，集成在子任务管理服务中，并非单独的模块）

13. 子服务器自动注册功能：提供分布式 GPU 计算端（以下称为计算端）的注册功能，并给相应的计算端创建并分配护照。（核心功能，集成在子服务器管理服务中，并非单独模块）

14. 子服务器签证更新功能：提供计算端的签证的更新功能，以此确认计算端的在线状态，防止计算端护照因为过久未更新签证而被注销。（核心功能，集成在子服务器管理服务中，并非单独模块）

15. 子服务器自动护照管理功能模块：提供子服务器的护照的管理功能，查找签证超过一段时间为更新的护照并注销。对于绑定了批处理任务的护照，该功能解锁相应的批处理任务并重新将该批处理任务加入优先级队列。（核心功

能模块）

16. 批处理任务自动调度服务（功能模块）：提供批处理任务的自动调度，解决分布式并行环境下的调度冲突问题。为计算端分配合适的批处理任务并且使得高优先级的批处理任务尽可能快的处理完成。（核心功能模块）

17. 批处理任务工作量估算功能：以句的粒度估算批处理任务的工作量，为批处理任务的拆分、调度与分配提供重要依据。（核心功能，集成在批处理任务自动调度服务中，并非单独模块）

18. 批处理任务拆分功能：按照计算端反馈自动拆分工作量过大的批处理任务，直到不可拆分为止。（核心功能，集成在批处理任务自动调度服务中，并非单独模块）

19. 批处理任务结果处理功能模块：结合计算端返回的模型计算结果与预分析处理结果，生成与批处理任务结果相适应的数据结构，使得数据能够储存到数据库中。并且，该功能模块负责根据批处理任务结果生成对应的子任务的结果，使得客户端能够获得子任务的处理结果。（核心模块）

20. 文件上传功能模块：使得主服务端能够保存客户端能够上传文件数据及相应信息，并能够对文档文件（Word）进行解析。

21. 文件散列值识别功能：主服务端能够计算文件数据的散列值，使得文件服务能够找到并引用文件缓存。（核心功能，集成在文件上传功能模块中）

5.4 用户界面设计

5.4.1 用户登录与注册界面

图 5-12 登录界面

该界面由账号输入框，密码输入框和登录按钮以及账号注册选项组成，采用 input 文本框接受用户的数字、字母与特殊字符，其中密码将会在本地进行 sha1 哈希加密，完成之后将其发送至后台。

5.4.2 我的提交历史记录界面

图 5-13“我的”界面

该模块采用抽屉式布局，按更新时间展示当前登录用户的历史提交处理结果，同时给出快捷评分预览。

5.4.3 启发模式界面

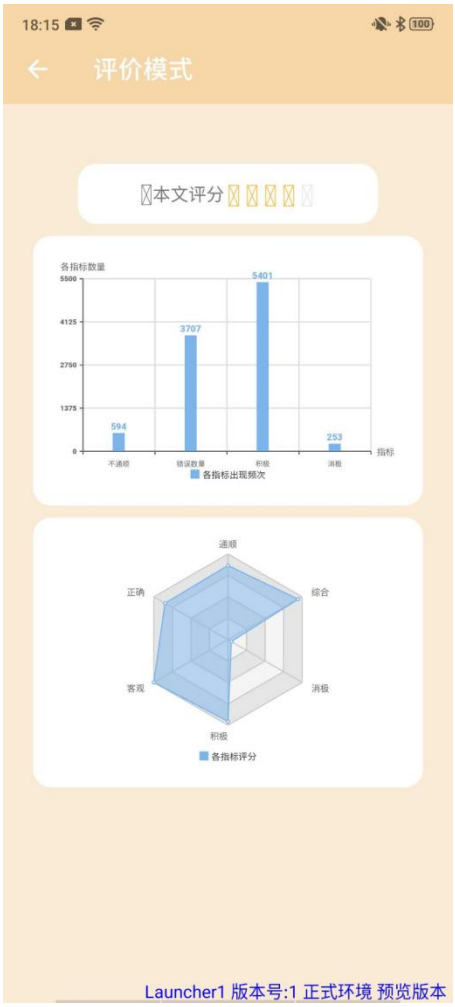
图 5-14 启发模式界面

该模块采用抽屉式布局，其中每个独立的选项卡按照不同的错误类别进行文本标注，点击“提示手”将会出现该处问题的详尽描述与修改建议。

5.4.4 评价模式界面

该模块采用 canvas 绘图，给出文章各个指标的可视化标图，其中指标有：文章通顺程度、错误数量、积极情感分数，消极情感分数，综合评分等数据。

图 5-15 评价模式界面



5.4.5 快应用卡片界面

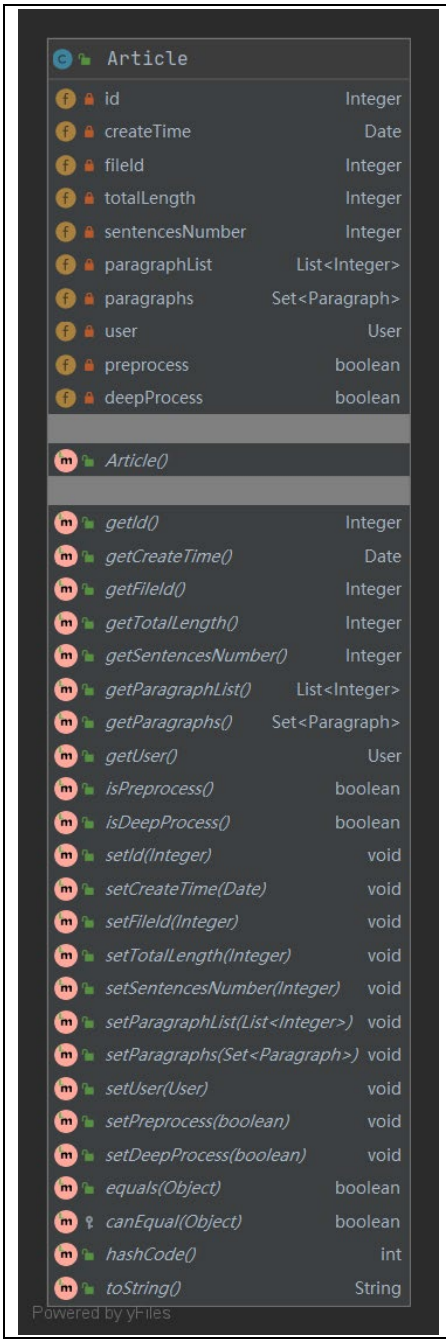
展示用户最近任务处理情况

图 5-16 快应用卡片功能调试

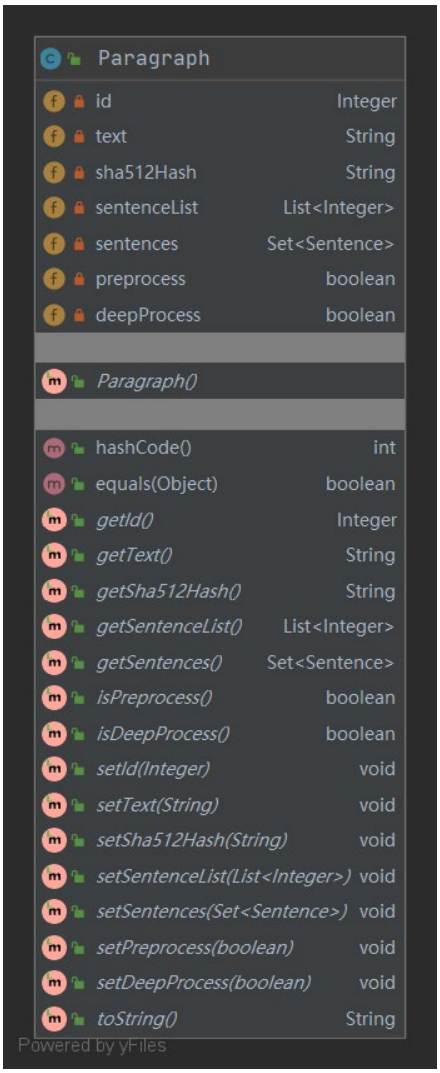
5.5 数据结构设计

服务端数据结构设计主要展示模型层的 UML 类图，目前只包括核心功能使用到的数据结构，这些数据结构绝大部分与数据库中的数据表一一对应。另外，下面所示的数据结构一般在创建后就被保存到数据库中，并且内存中的数据与数据库中的数据会实时更新。

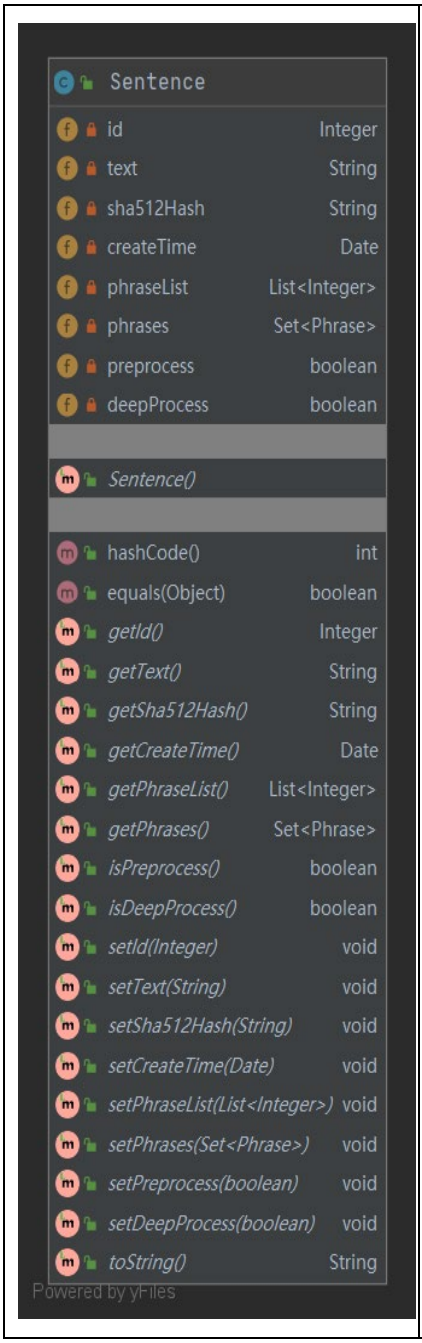
5.5.1 文本处理类数据结构



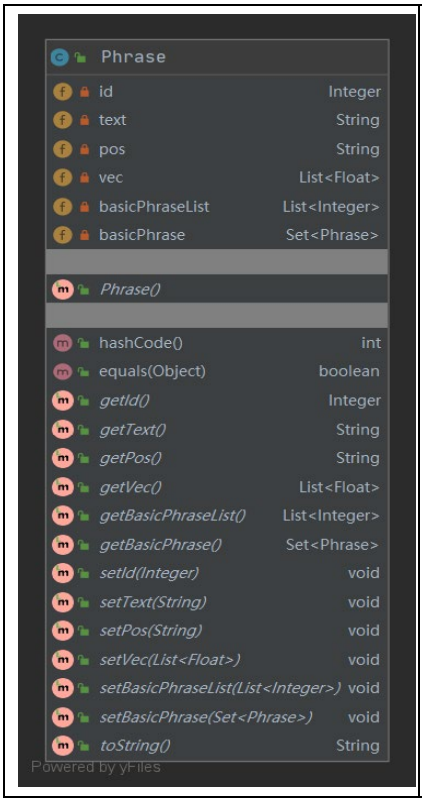
章结构描述类：该数据结构记录由文件服务解析出的文档中与整篇论文有关的信息。这些信息主要包括：章句数、段列表（记录文章中文段的顺序）、段集合（负责记录文段数据，防止重复的文段浪费服务器的储存空间）、预处理状态（preprocess）标记、模型计算状态（deepProcess）标记等。



段结构描述类：记录一篇文章中的文段数据，与章结构关联。该数据结构是预处理器分段处理过程自动生成的。该数据结构主要的信息包括：具体内容、哈希值（用于缓存机制）、句列表（记录文段中句子的顺序）、句集合（记录文段中句子的数据）、预处理状态（preprocess）标记、模型计算状态（deepProcess）标记等。

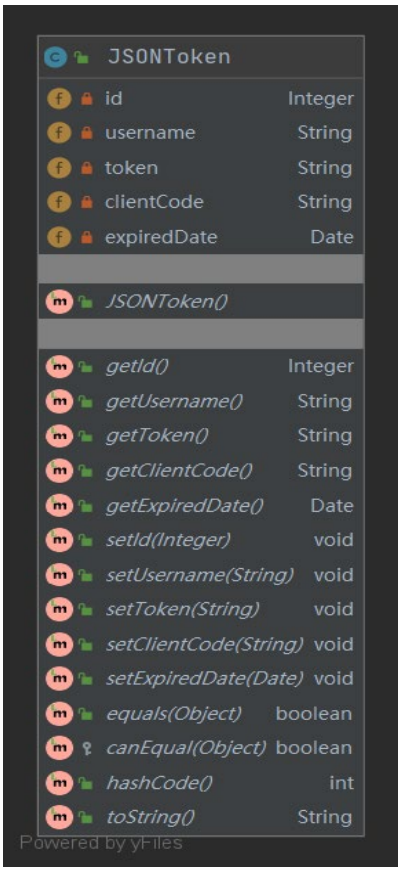


句结构描述类：记录文段中句子的数据。该数据结构是预处理在分句处理过程自动生成的，其词列表与词集合的内容是预分析器自动分词过程填充的。其记录的主要内容有：句子文本内容、哈希值、词列表、词集合、预处理状态（preprocess）标记、模型计算状态（deepProcess）标记。

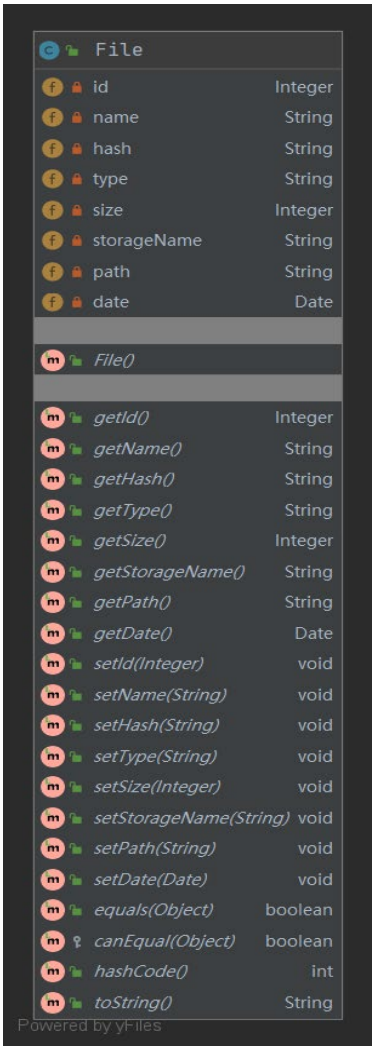


词结构描述类：记录词语的文本、词性、词向量、语素列表等内容。该数据结构是预分析器的分词处理过程自动生成的。

5.5.2 信息及验证类数据结构

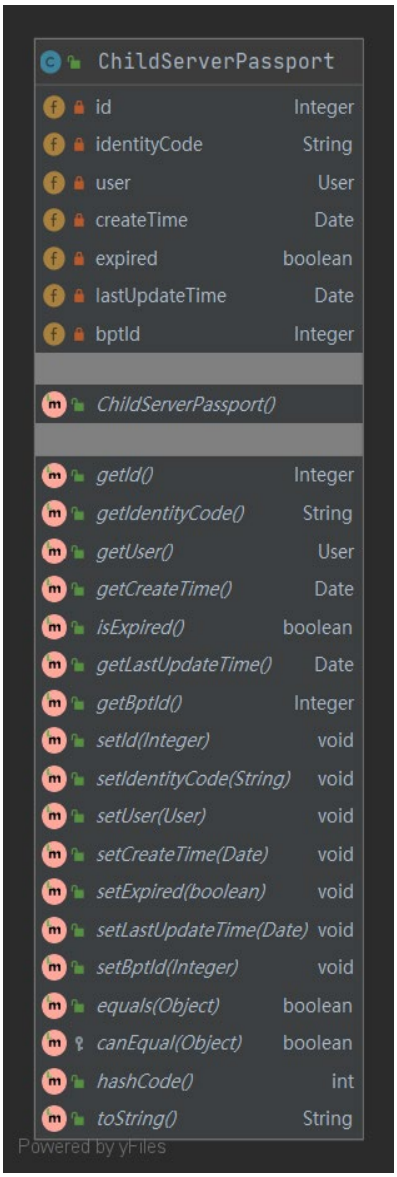


Token 描述类：认证成功后，账号认证服务自动生成该数据结构。该数据结构记录 token 的值、token 对应的账号、客户端代码（用于检验客户端签名）、token 过期的时间（一般设置为 30 分钟）。



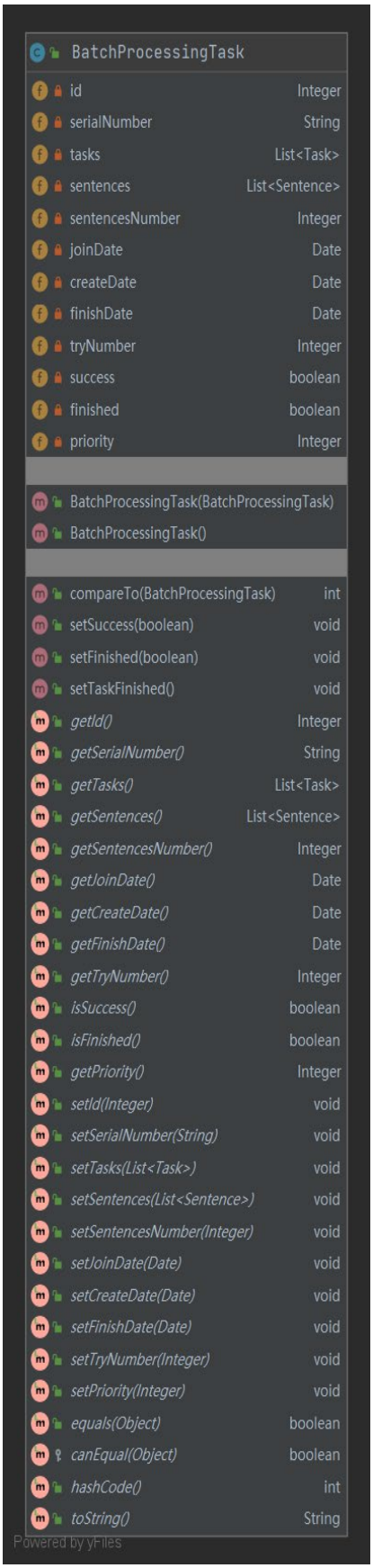
文件描述类：用于记录主服务端当前所掌握的文件的信息，而与之对应的文件的数据会被写出到文件系统中保存。该数据结构记录的主要信息由：文件 ID 号、文件名、文件数据的哈希值、文件的类型（一般为后缀名）、文件大小（字节单位）、文件在主服务端文件系统中的储存的文件名（一般为 UUID 的字符串形式）、文件储存路径等。

5.5.3 子服务器管理类数据结构

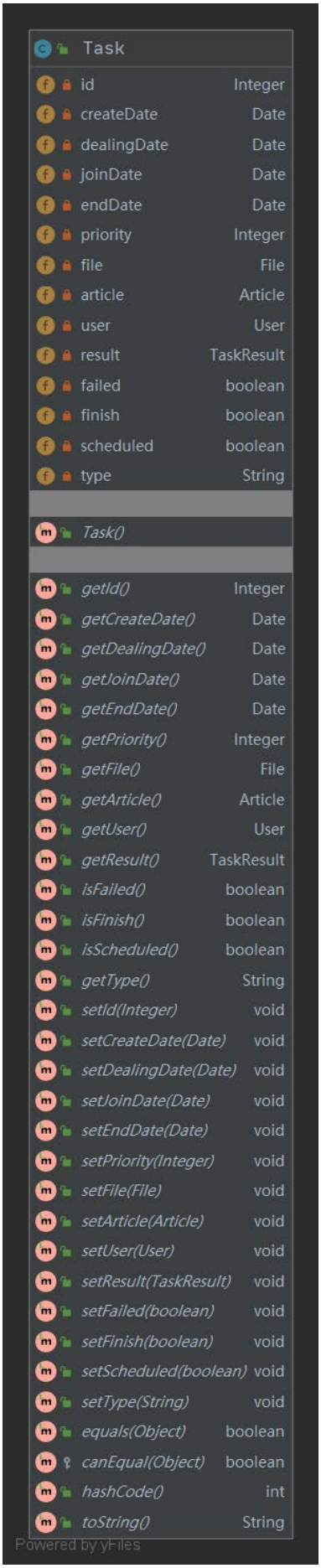


子服务器护照描述类：用于记录子服务器（计算端）的护照信息，其主要信息包括身份认证码（为 UUID 的字符串形式，用于唯一标识护照）、关联的子服务器账号、过期标志位、最后一次更新签证的时间、关联的批处理任务的 ID 号等。

5.5.4 任务类数据结构

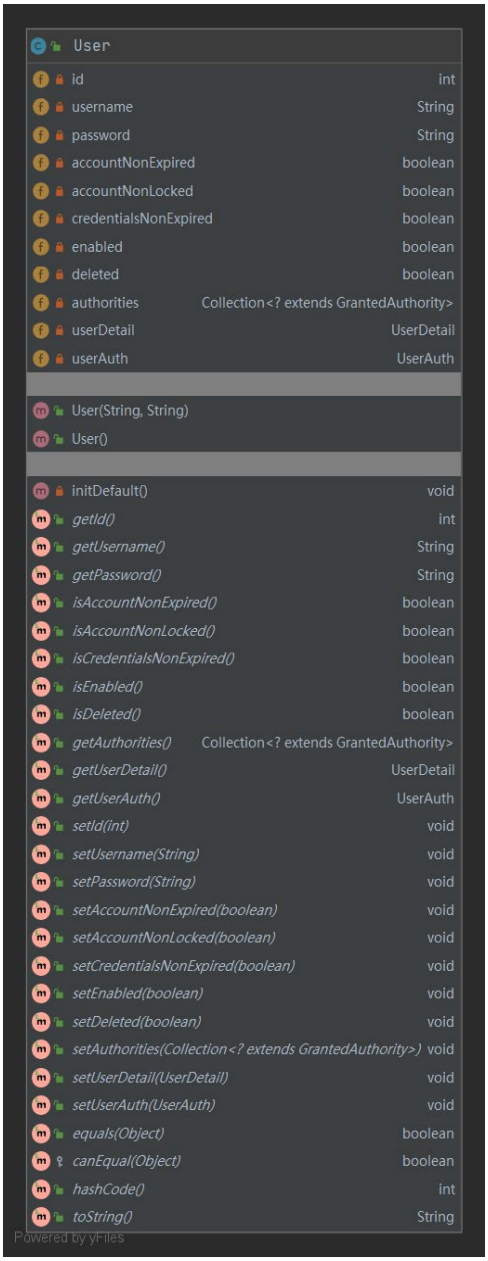


批处理任务描述类：批处理任务是系统作业的基本调度单位，也是模型运算的基本处理单位。用于描述一个批处理任务，在子任务合并为批处理任务时自动创建。其主要记录的信息包括：批处理任务流水号（暂未使用）、关联的子任务列表、句子列表（暂未使用）、加入到优先级队列的时间、创建的时间、完成的时间、尝试处理的次数、是否成功标记、是否完成标记、优先级。

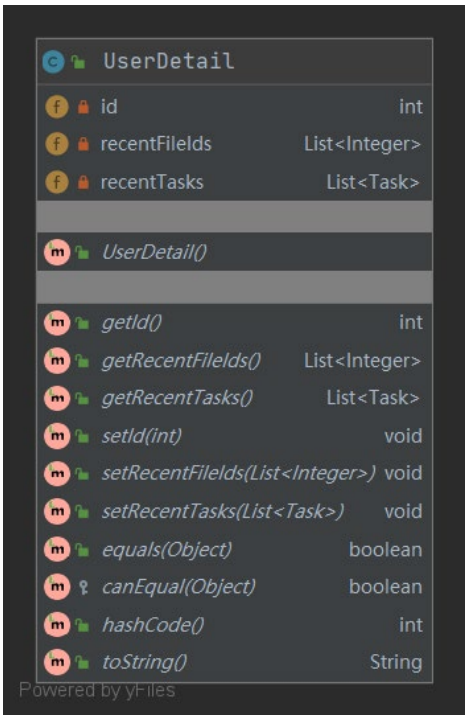


子任务管理信息描述类：子任务是系统作业的最小管理单位。该数据结构用于记录子任务管理信息，这些信息主要包括子任务创建时间、子任务的处理时间、子任务加入优先级队列的时间、子任务完成时间、子任务优先级（实际代码暂未使用）、子任务关联文件 ID 号（用于预处理）、子任务关联文章信息、子任务关联用户

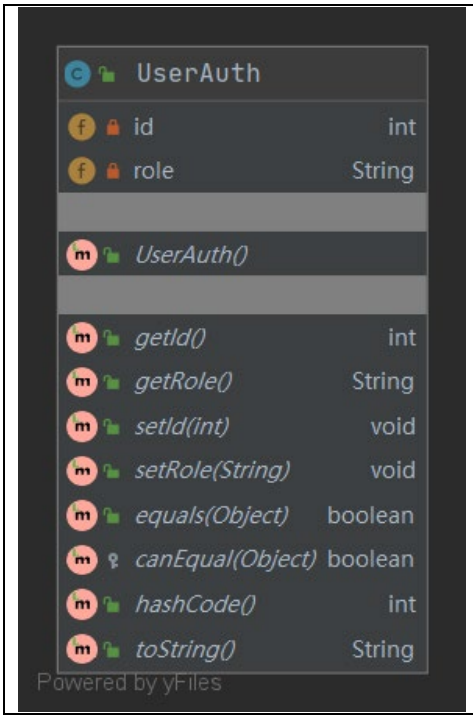
5.5.5 账号类数据结构



账号描述类：用于描述一个账号的基本信息。这些信息包括用户名（openid），密码（经过哈希处理）、账号过期标志、账号封禁标志、账号密码过期标志、账号激活标志（当前默认创建即激活）、账号删除标志、账号身份信息等。

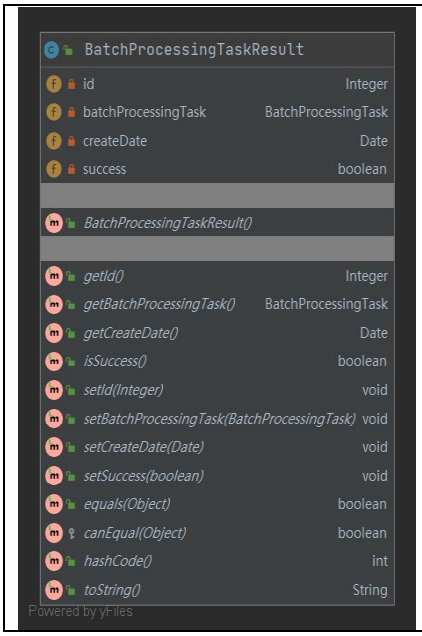


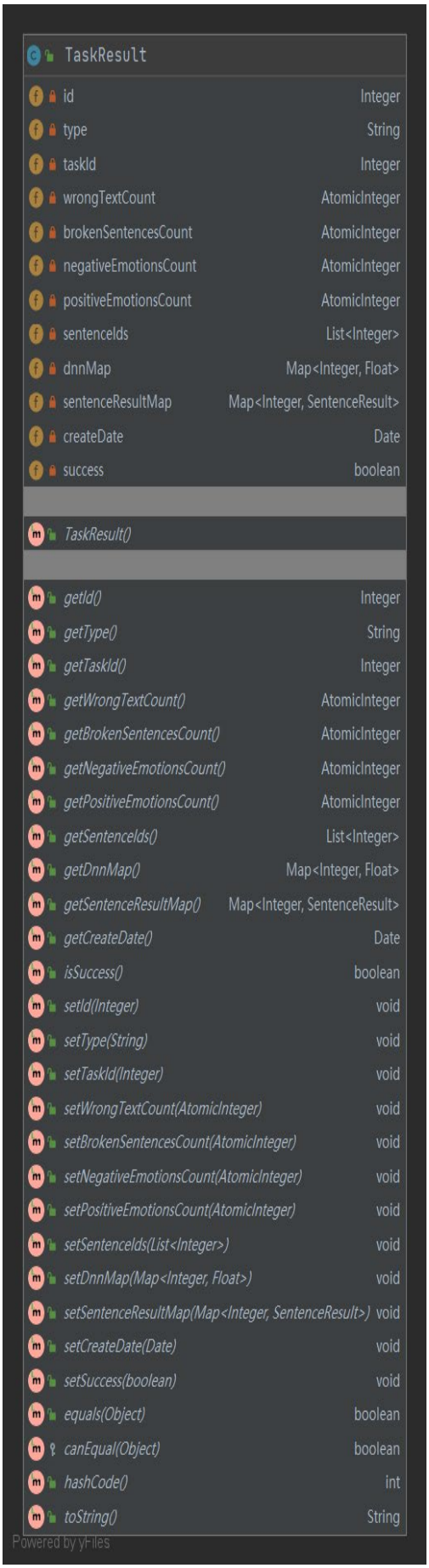
账号详细信息描述类：记录账号的一些拓展信息，当前主要记录用户上传的文件记录、用户创建的子任务记录。



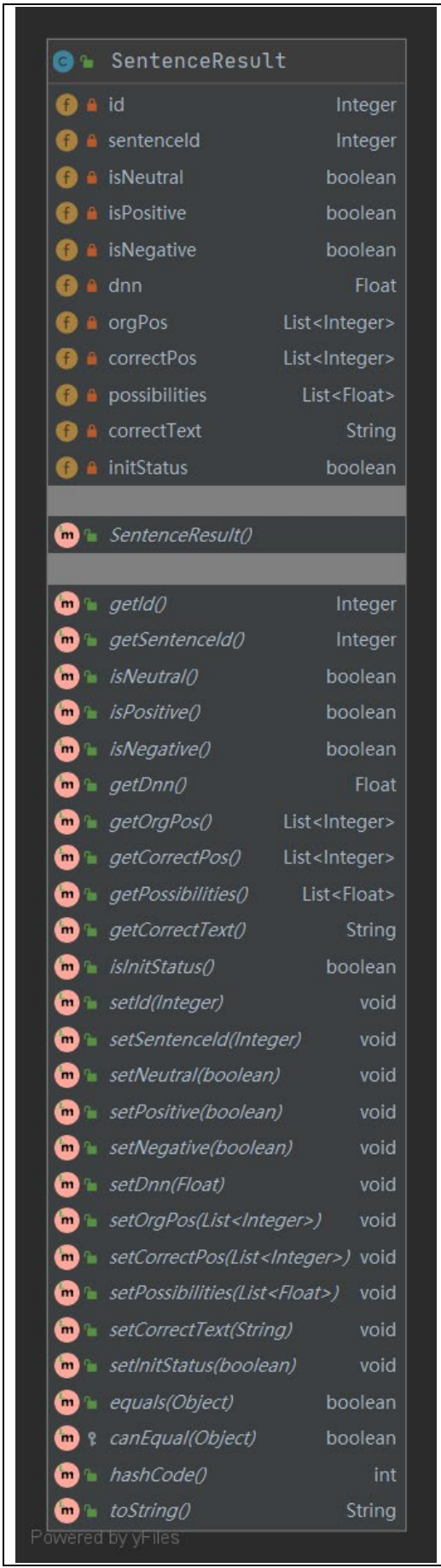
账号认证信息描述类：记录账号的认证信息，当前主要为身份信息。

5.5.6 结果类数据结构

	**批处理任务结果描述类：**用于记录批处理任务处理的结果（实际代码暂未使用）。



子任务处理结果描述类：用于记录子任务的处理结果，主要数据项包括：子任务类型、子任务 ID 号、存在错误的句子数、不完整的句子数、带有积极倾向口语化特征句子数、带有消极倾向口语化特征句子数、句子列表、句子通顺程度 Map 结构、句子的处理结果数据列表等。



句处理结果描述类：记录句子的处理结果，主要数据项包括：对应句子的 ID 号、是否中立、是否积极口语化、是否消极口语化、通顺程度、正确的文本等。

5.6 接口设计

5.6.1 外部接口

快应用端软件接口包括快应用官方提供的获取手机内文件的接口、蓝牙连接的接口、将文件下载至手机的接口、以及团队主服务端提供的注册接口、登录身份验证接口、文件上传接口、获取原文接口、获取文章状态接口、获取错误列表接口。上述有关主服务端的接口，在内部接口设计中有详细说明。

主服务端对于百度 NLP 自然语言处理平台 API 接口调用，接口以及函数如下：

接口名称	简要描述	调用函数
词法分析	分词、词性标注、专名识别	lexer(text,options)
词向量表示	查询词汇的词向量，实现文本的可计算	wordEmbedding(word,options)
DNN 语言模型	判断一句话是否符合语言表达习惯，输出分词结果并给出每个词在句子中的概率值	dnnlmCn(text, options)
文本纠错	识别文本中有错误的片段，进行错误提示并给出正确的建议文本内容	ecnet(text, options)

5.6.2 内部接口

部分数据为了排版美观经过适当裁减，而且大部分数据是真实的计算数据。下列内部接口主要为主服务端对客户端（快应用端）与计算端的接口，且在形式上均采用 Restful API 风格进行设计。当前只给出了核心功能的所涉及的必要接口的信息。

快应用端各个页面之间的跳转以及信息传递，主要通过 router 以及全局变量实现

下列表格中，服务器返回部分给出了接口的正确使用状态下，主服务端的返回的内容及状态号。

5.6.2.1 账号类接口

RESTful API 接口		
接口功能（Function）	创建用户	
HTTP 动词（Method）	POST	
URL	/user	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（精确到毫秒）
	X-Requested-With	
HTTP 报文参数（Params）	无	

HTTP 报文内容（Body）	{
	"openid": "test",
	"password": "095d9171a3b2ff39d25ce1db8afca13cc1c795"
	}	

服务器返回代码（Status Code）	201	

服务器返回报文（Data）	{
	"id": 21,
	"openid": "test2",
	"password": "f710a49f46296486759853d9dc63219308"
	}	

RESTful API 接口	
接口功能（Function）	检查用户是否被注册
HTTP 动词（Method）	GET
URL	/user

HTTP 报文头部（Header）	timestamp	Unix 时间戳（精确到毫秒）
	X-Requested-With	
HTTP 报文参数（Params）	openid	用户的 openid
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	201	

服务器返回报文（Data）	{
	"id": 1,
	"openid": "test",
	"password": null
	}	

RESTful API 接口		
接口功能（Function）	用户登录（获得 token）	
HTTP 动词（Method）	POST	
URL	/user/login	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（精确到毫秒）
	X-Requested-With	
HTTP 报文参数（Params）	openid	用户的 openid

HTTP 报文内容（Body）	{
	"openid": "test",
	"password": "5d9171a3b2ff39d25ce1db8afca13cc1c795",
	"clientCode": "1481977208cd482dbe3e7e2fabfc113a89a120"
	}	

服务器返回代码（Status Code）	200	

服务器返回报文（Data）	{
	"data": {
	"loginStatus": true,
	"pvc": "5c6a4ba4-5690-4925-96ca-7df0930e538e",
	"token": "6c5c84bba2ec2dfa9edb791203054f0721",
	"userExist": true
	},
	"info": "Authentication Success",
	"msg": "OK",
	"status": 200
	}

5.6.2.2 文件类接口

RESTful API 接口		
接口功能（Function）	上传文件	
HTTP 动词（Method）	POST	
URL	/file	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）		
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	201	
服务器返回报文（Data）	{ "fileId": 18, "filename": "SWC 技术研究报告_0.2.4.docx", "type": "docx"	

	}

5.6.2.3 任务类接口

RESTful API 接口		
接口功能（Function）	创建子任务	
HTTP 动词（Method）	POST	
URL	/task	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）		
HTTP 报文内容（Body）	{ "fileId": 13 }	
服务器返回代码（Status Code）	201	
服务器返回报文（Data）	{ "taskId": 24, "openid": null, "fileId": 13, "finished": false }	

RESTful API 接口		
接口功能（Function）	查询子任务状态	
HTTP 动词（Method）	GET	
URL	/task	
HTTP 报文头部	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）

（Header）	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）	taskId	任务 ID 号
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	200	

服务器返回报文（Data）	<pre>{
	"taskId": 24,
	"openid": "test",
	"fileId": 13,
	"finished": false
	}</pre>	

RESTful API 接口		
接口功能（Function）	查询子任务的分句列表	
HTTP 动词（Method）	GET	
URL	/task/stnlist	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）	taskId	任务 ID 号
	page	分页号（从 1 到 all）
HTTP 报文内容（Body）		

服务器返回代码（Status Code）	200
服务器返回报文（Data）	{ "page": 1, "all": 11, "stns": [{ "stnId": 170, "text": "NSP，预测下一句模型，增加对句子 A 和 B 关系的预测任务，50%的时间里 B 是 A 的下一句，分类标签为 IsNext，另外 50%的时间里 B 是随机挑选的句子，并不是 A 的下一句，分类标签为 NotNext" }, { "stnId": 171, "text": "该问题的解决使用的 Transformer 编码器与 BERT 模型所属方向是深度学习" }, { "stnId": 172, "text": "对于语段的分句，将采用创新的启发式算法，对于文段进行基于使用习惯的按句子的自动拆分，将夹杂由其他成分的文段转换为纯中文的句集" }, { "stnId": 173, "text": "而对于错字处理、中文语句的分词与词性分析，该项目将采用百度 AI 开放平台的语言处理基础技术" }, { "stnId": 174, "text": "对于词语搭配问题，该项目将采用百度 AI 开放平台的 DNN 模型" }, { "stnId": 175, "text": "对于句子结构混乱的问题，将根据分词与词性分析的结果根据自研启发式算法推断出词语在句子中的成分及角色，根据上述信息结合语言的使用法则得出结论" }, { "stnId": 176, "text": "最后根据该问题的特点，为了摆脱自然语言处理大量语料无监督训练的高成本与获得更好的业务适应能力，该项目的口语化检

	测与严谨性扫描部分将采用 BERT 模型，并对这两个问题进行创新性特化建模
	"
	},
	{
	"stnId": 177,
	"text": "对于分句，启发式算法能够将文段进行句子粒度的拆分，对于句子中的英文字母、公式、引用进行删除、等效替代等处理，尽量不改变原文的意思"
	},
	{
	"stnId": 178,
	"text": "最终，启发式算法将能够把一般文段转换为等效的纯中文的句子集合"
	}
	]
	}

RESTful API 接口		
接口功能（Function）	查询子任务执行结果	
HTTP 动词（Method）	GET	
URL	/task/result	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）	taskId	任务 ID 号
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	200	
服务器返回报文（Data）	{ "taskId": 24,	

```
    "success": true,
    "wrongTextCount": 10,
    "brokenSentencesCount": 0,
    "negativeEmotionsCount": 2,
    "positiveEmotionsCount": 1,
    "score": 93.28554164329445,
    "dnnScore": 89.00305843906396,
    "emotionScore": 88.84195492747085,
    "correctionScore": 95.12089158796465,
    "stnResults": [
        {
            "stnId": 581,
            "appear": 0,
            "score": null,
            "errorList": [
                {
                    "wordIdx": 0,
                    "wordLen": 2147483647,
                    "type": 2,
                    "content": "文本语言识别为具有积极因素的口语化特征，建议修改为书面语。"
                }
            ],
            "neutral": true
        }
]
```


5.6.2.4 子服务器管理类接口

RESTful API 接口		
接口功能（Function）	子服务器账号注册（分布式 GPU 计算端账号）	
HTTP 动词（Method）	POST	
URL	/user/cs	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
HTTP 报文参数（Params）	clientCode	计算端认证口令（约定而成，防御普通攻击）

HTTP 报文内容（Body）	{
	"openid": "gpu-server-0",
	"password": "095d9171a3b2ff39d25ce1db8afca13cc1c795"
	}	

服务器返回代码（Status Code）	201	

服务器返回报文（Data）	{
	"id": 567,
	"openid": "gpu-server-0",
	"password": "5c030341516c204db54ff45fffba470c"
	}	

RESTful API 接口		
接口功能（Function）	子服务器护照获取（计算端护照）	
HTTP 动词（Method）	POST	
URL	/cs	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）		
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	201	

服务器返回报文（Data）	{
	"identityCode": "a23303e2-ffc6-4442-aaf8-04fc54fc45f4",
	"lastUpdateTime": "2020-04-21T09:07:13.427+0000",
	"expired": false
	}	

RESTful API 接口		
接口功能（Function）	子服务器签证获取（更新签证确保护照不过期）	
HTTP 动词（Method）	PUT	
URL	/cs	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）	idcode	即 identityCode，护照身份认证码
HTTP 报文内容（Body）		
服务器返回代码（Status Code）	200	

服务器返回报文（Data）	{
	"identityCode": "a23303e2-ffc6-4442-aaf8-04fc54fc45f4",
	"lastUpdateTime": "2020-04-21T09:07:19.542+0000",
	"expired": false
	}	

5.6.2.5 批处理任务管理类接口

RESTful API 接口		
接口功能（Function）	获得一个合适的批处理任务	
HTTP 动词（Method）	GET	
URL	/cs/bpt	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	

<table>
<tr><td rowspan="2"></td><td>openid</td><td>用户的 openid（用于身份验证）</td></tr>
<tr><td>signed</td><td>客户端签名（实时计算而成，用于身份验证）</td></tr>
<tr><td rowspan="2">HTTP 报文参数（Params）</td><td>idcode</td><td>即 identityCode，护照身份认证码</td></tr>
<tr><td>maxStnNum</td><td>计算端单次最大可处理的句数</td></tr>
<tr><td>HTTP 报文内容（Body）</td><td colspan="2"></td></tr>
<tr><td>服务器返回代码（Status Code）</td><td colspan="2">200</td></tr>
<tr><td>服务器返回报文（Data）</td><td colspan="2"><pre>
{
    "id": 565,
    "stnNumber": 6,
    "stns": [
        {
            "stnId": 70,
            "text": "对于分词部分，自然语言处理基础技术能够正确将句子拆分成词集"
        },
        {
            "stnId": 71,
            "text": "对于语言搭配问题，DNN 模型需要能够结合语言的习惯用法，"
        },
        {
            "stnId": 72,
            "text": "对于句子结构混乱的问题，启发式算法需要能够检测出句子是否含有句子结构混乱的问题"
        },
        {
            "stnId": 73,
            "text": "对于口语化的问题，BERT 模型需要根据语言的习惯用法的得出句子的口语化特征的量化程度"
        },
        {
            "stnId": 74,
            "text": "对于句子严谨性的问题，BERT 模型需要根据微调过程提取的学术论文行文特征得出句子的严谨性量化值，找到不符合学时论文行文特点的句子"
        },
</pre></td></tr>
</table>

	{
	"stnId": 75,
	"text": "此外，BERT 模型需要将不符合中文学术论文的行文特点的句子，通过机器翻译特化应用模型将这样的句子转化成为符合学术论文行文特点的较为严谨的句子"
	}
	]
	}

RESTful API 接口		
接口功能（Function）	子服务器提交模型计算结果（更新 BPT 状态为已完成）	
HTTP 动词（Method）	POST	
URL	/cs/	
HTTP 报文头部（Header）	timestamp	Unix 时间戳（毫秒级）（防御重放攻击）
	X-Requested-With	
	openid	用户的 openid（用于身份验证）
	signed	客户端签名（实时计算而成，用于身份验证）
HTTP 报文参数（Params）	idcode	即 identityCode，护照身份认证码
	status	成功与否
	bptId	批处理任务 ID 号
HTTP 报文内容（Body）	[{ "stnid": 70, "tagPossible": [0.000004637905, 0.9999908, 0.000004513615] }, { "stnid": 71, "tagPossible": [0.0000046930695, 0.9999912, 0.000004136567	

	]
	},
	{
	"stnid": 93,
	"tagPossible": [
	0.060555745,
	0.0001441841,
	0.93930006
	]
	}
	]

服务器返回代码（Status Code）	201
服务器返回报文（Data）	

5.7 错误/异常处理设计

5.7.1 错误/异常输出信息

快应用端：

1. 用户注册页面
   1）账号或密码为空：弹出 message 警告“账号/密码为空”
   2）账号为已注册：弹出 message 警告“该账号已被占用”
2. 用户登录页面
   1）账号或密码为空：弹出 message 警告“账号/密码为空”
   2）账号未被注册：弹出 message 警告“账号未被注册”
   3）账号或密码不正确，弹出 message 警告“登录失败请检查账号密码”
3. 首页上传文件：
   1）调用系统接口失败或系统未相应：抛出异常
   2）文件读取错误/格式不匹配：弹出 message 警告“请上传 doc 或者 docx 格式的文件”
   3）任务上传失败：抛出异常，上传成功标志长暗
   4）用户未输入文本也未上传文件点击提交：提示用户未输入文件/文本

4. 我的页面：

1）文件未处理完点击查看信息：抛出异常，提示文件正在处理中

5. 打印文件：

1）文件未处理完开始打印：抛出异常，提示文件正在处理中

主服务端：

对于主服务端检测到的一般的错误与异常，项目产品主要输出以 JSON 形式来展现错误信息。另外，主服务器返回状态也对应着相关的错误信息。错误信息相对应的返回状态码均遵照 Restful API 风格。

以下是错误输出信息的 HTTP 报文内容的一般形式（JSON）：

```
{
    "data": {
        "date": 1587476031927,
        "exception": "org.codedream.epaper.exception.notfound.NotFoundException",
        "exceptionMessage": "99"
    },
    "msg": "Not Found",
    "status": 404
}
```

以下是对身份验证失败的错误的返回形式：

```
{
    "msg": "Unauthorized",
    "status": 401
}
```

注明：HTTP 返回状态码与 status 的值一一对应。

5.7.2 错误/异常处理对策

主服务端采用 Spring 框架提供的 ControllerAdvice 来统一对一般的运行时产生异常管理，具体地主服务端将一般的异常归类，然后与常见的 HTTP 状态码所对应的错误含义进行一一对应。在 ControllerAdvice 捕捉到相应的异常后，依照异常与 HTTP 状态码的对应关系，结合异常对象自身包含的信息构造 JSON 并连同状态码返回给客户端。对于认证与验证服务产生的异常，由于主服务端对 Spring Security 框架的深度定制，使得其发生错误而返回的形式与一般的异常基本相同，返回的状态码也是有意义的。这样，绝大多数的异常无法使主服务端停

止运行，异常信息的返回又能辅助客户端与计算端针对异常进行相应的自动调整。

计算端采用异常捕捉的方式捕捉所有的 Python 本身或者相应的库产生的异常，然后对一些基本的异常采取预先设置的合适的方案进行处理。对于在与主服务端交互过程中主服务端发挥的异常信息，计算端将按照状态码分类进行处理。对于无法处理的异常，由于计算端采取多线程定时器的形式运作，计算端将采取中止线程并重设定时器来完成部分功能的自我重启。这样所有的错误与异常也将无法使计算端的所有服务停止，大大地提高了计算端服务运行的稳定性。

5.8 系统配置策略

主服务端的配置策略十分灵活，它的要点在于服务的伸缩性、系统的整体稳定性、吞吐量。而主服务端的配置的可操作点在于线程池的配置、子任务合并为批处理任务的最长等待时间、批处理任务计算量上限、批处理任务的优先级设置、批处理任务的解锁最大时间、护照管理服务的运行间隔、护照的签证不更新导致护照过期的最长允许时间。而具体的配置项的设置要以主服务端软件系统实际的运行环境为准。

- 若主服务端的运行环境 CPU 功能较为强大，所支持的并发数多，则线程池可以随之增大，可并行运行的线程也可以相应增多，线程的活跃时间也可以相应延长。

- 如果单位时间内客户端的并发数比较大或者分布式 GPU 计算端不够，那么可以增加子任务合并为批处理任务的最长等待时间，使得子任务可以延迟合并成批处理任务。另外，批处理任务的计算量上限也可以随之增大。这样分布式 GPU 计算端的工作压力可以适当减轻。

- 如果分布式 GPU 计算端的数量足够多，那么子任务可以设置更快地转换成批处理任务，这样子任务的周转时间就可以缩短，提升用户的使用体验。

- 以子任务的周转时间为标准，如果 GPU 计算端的性能足够强大，吞吐量足够大，那么其处理人物解锁最大时间应该相应下调或者护照管理服务

的运行周期可以相应缩短，防止批处理任务被锁定太久（批处理任务锁定过久则表明计算端中途退出或者与主服务器的网络连接中断）。

- 根据网络环境的实际情况也可以调整护照的签证不更新导致护照过期的最长允许时间，这样主服务器就可以更快地检测到失去连接的计算端，并在更短时间内释放被其锁定的批处理任务。

分布式 GPU 计算端的配置策略简单可靠，它的要点在于吞吐量、与主服务器连接的稳定性。分布式 GPU 计算端的可配置点主要有：获得批处理任务获取服务的定时器时间、签证更新服务定时器时间、单次最大任务处理量。

- 如果 GPU 的性能足够强，则可以相应上调该计算端单次最大任务处理量，也可以缩短批处理任务获取服务的定时器时间。

- 如果网络环境足够好，也可以相应延长签证更新服务定时器时间，降低由于高频率更新签证带来的开销（虽然很小）。

5.9 系统部署方案

主服务端采用基于 Jenkins 的自动构建、自动部署的方案，对服务器的硬件与网络要求较低。首先，主服务器端仅要求部署服务器带有固定的公网 IP、具备有不少于 5M 的上行带宽。在处理能力上，部署服务器的 CPU 推荐具备双核四线程、运行内存推荐具备有 4GB、剩余储存空间具备有 16GB。主服务器部署的时候推荐安装有 Jenkins 来支持主服务端的自动构建与自动部署（直接编译效果相同）。Jenkins 可以连接到项目的私有仓库地址，设置好 Web Hook 与构建状态反馈。这样，维护人员能够快速对主服务端存在的漏洞进行修补，使得主服务端能够实时保持最新版本。部署服务器推荐安装 JDK 1.8_241，JRE 1.8_241（实际上仅安装 JDK1.8 即可）。另外部署服务器推荐安装 Maven、Git，以进一步实现自动构建功能。

分布式 GPU 计算端的部署采取多计算机部署的方案，支持根据系统的事实际负载增减计算端数量。首先，在网络环境上，并不要求计算端部署在带有公网 IP 的服务器上，只要求部署的计算机能够连接到互联网。另外，计算端允许 NAT 转

发的存在。在运算能力上，计算端不要求计算端具备有性能较强的 GPU。计算端推荐的 GPU 硬件的计算能力仅与 NVIDIA GTX 1050ti 相当，当然可以更高。在 CPU 能力足够的情况下，计算端甚至可以不安装 GPU 就能正常作业。在软件环境上，只要求 Python 的版本为 3.6，并且安装 TensorFlow-GPU 1.14，CUDA 10.1 以及配套的 CDNN （或者只安装 TensorFlow 1.14）。Python 只要求安装最新版本的 shutil、requests、hashlib 与 pandas。对于计算端相关运行环境，推荐使用 Anaconda3 快速安装。

5.10 其他相关技术与方案

- 主服务端软件系统采用 Spring Boot 框架，建立 Maven 工程，并使用 Spring Security、Spring Web 进行工程配置。
- 对于数据库的管理，采取 JPA 技术，在数据库与 Java Object 之间建立关联，简化数据库的管理与应用程序的编写。
- 在 JSON 的解析上，使用 Alibaba FastJson 包，提升 JSON 解析速度。
- 对于接口文档，使用 swagger 的基于注解的自动生成技术，使得软件系统具备自动生成 API 文档的能力。
- 在单元测试时使用 junit，加快单元测试。
- 使用 Lombok 提供的注解，加速 Java 应用程序的开发。
- 数据库采用 MariaDB 10.2，并且根据情况进行有针对性的优化配置。
- 使用 slf4j-simple 提供的注解简化对日志代码的编写。
- 使用 org.apache.poi 包对 Word 文档进行解析。

6 数据库设计

6.1 数据表设计

- 文章表(article)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		文章 id
create_time	datetime				创建时间
deep_process	bit(1)				是否深度处理
file_id	int(11)				文件 id
preprocess	bit(1)				是否预处理
sentences_number	int(11)				待处理句子总数
total_length	int(11)				文本总长度
user_id	int(11)		X		用户 id

- 文章-可重复段落中间表(article_paragraph_list)

Name	Data Type	Primary	Is Key	Foreign Key	Description
article_id	int(11)		X	X	文章 id
paragraph_list	int(11)				可重复段落 id

- 文章-不可重复段落中间表(article_paragraphs)

Name	Data Type	Primary	Is Key	Foreign Key	Description
article_id	int(11)	X	X	X	文章 id
paragraphs_id	int(11)	X	X		不可重复段落 id

- 批处理任务表(batch_processing_task)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		批处理任务 id
create_date	datetime				任务创建时间
finish_date	datetime				任务完成时间
finished	bit(1)				任务是否处理完成
join_date	datetime				加入处理队列时间
priority	int(11)				任务优先级
sentences_number	int(11)				待处理句子总数

serial_number	varchar(255)				任务流水号
success	bit(1)				任务是否成功返回
try_number	int(11)				任务尝试次数

- 批处理任务-句子中间表(batch_processing_task_sentences)

Name	Data Type	Primary	Is Key	Foreign Key	Description
batch_processing_task_id	int(11)		X	X	批处理任务 id
sentences_id	int(11)		X		待处理句子 id

- 批处理任务-任务中间表(batch_processing_task_tasks)

Name	Data Type	Primary	Is Key	Foreign Key	Description
batch_processing_task_id	int(11)		X	X	批处理任务 id
tasks_id	int(11)		X		任务 id

- 批处理任务记录(bpt_record)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		批处理-记录过渡 id
bpt_id	int(11)				批处理任务 id
message	varchar(255)				记录信息
operation_type	varchar(255)				操作类型
time	datetime				记录时间

- 批处理结果(bpt_result)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		批处理任务结果 id
create_date	datetime				创建时间
success	bit(1)				成功标记
batch_processing_task_id	int(11)		X	X	批处理任务 id

- 分布式计算业务子服务器护照(csp)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		子服务护照 id
bpt_id	int(11)				批处理任务 id
create_time	datetime				服务创建时间

expired	bit(1)				服务超时时间
identity_code	varchar(255)				身份码
last_update_time	datetime				上次更新时间
user_id	int(11)		X		用户 id

- 文件(file)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		文件 id
date	datetime				文件上传时间
name	varchar(255)				文件名
path	varchar(255)				文件路径
sha256	varchar(255)				文件 SHA256 加密码
size	int(11)				文件大小
storage_name	varchar(255)				储存名
type	varchar(255)				文件类型

- 文件记录(file_record)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		文件记录 id
file_id	int(11)				文件 id
message	varchar(255)				记录信息
operation_type	varchar(255)				操作类型
time	datetime				记录时间

- 用户 token(json_tokens)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		token 记录 id
client_code	varchar(255)				客户端标识口令
expired_date	datetime				token 过期时间
token	varchar(255)		X		token 值
username	varchar(255)		X		token 对应的用户

- 长文本(long_text)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		长文本 id
create_time	datetime				生成时间

sha512hash	varchar(255)				SHA512 编码
text	longtext				文本内容

- 段落(paragraph)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		段落 id
deep_process	bit(1)				深度处理标记
preprocess	bit(1)				预处理标记
sha512hash	varchar(255)				SHA512 码
text	longtext				段落内容

- 段落-可重复句子中间表(paragraph_sentence_list)

Name	Data Type	Primary	Is Key	Foreign Key	Description
paragraph_id	int(11)		X	X	段落 id
sentence_list	int(11)				待处理句子 id

- 段落-不可重复句子中间表(paragraph_sentences)

Name	Data Type	Primary	Is Key	Foreign Key	Description
paragraph_id	int(11)	X	X	X	段落 id
sentences_id	int(11)	X	X		不可重复句子 id

- 短语(phrase)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		短语 id
pos	varchar(255)				短语词性
text	varchar(255)				文本内容

- 基本词(phrase_basic_phrase)

Name	Data Type	Primary	Is Key	Foreign Key	Description
phrase_id	int(11)	X	X	X	短语 id
basic_phrase_id	int(11)	X	X	X	基本词 id

- 基本词-词向量中间表(phrase_vec)

Name	Data Type	Primary	Is Key	Foreign Key	Description
phrase_id	int(11)		X	X	短语 id
vec	float				词向量

- 预校验(pre_validation)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		预校验 id
date	datetime				预校验时间
value	varchar(255)				校验值

- 句子(sentence)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		句子 id
create_time	datetime				句子生成时间
deep_process	bit(1)				深度处理标志
preprocess	bit(1)				预处理标志
sha512hash	varchar(255)				SHA512 编码
text	varchar(255)				句子文本

- 句子-可重复基本词中间表(sentence_phrase_list)

Name	Data Type	Primary	Is Key	Foreign Key	Description
sentence_id	int(11)		X	X	句子 id
phrase_list	int(11)				可重复基本词组 id

- 句子-不可重复基本词中间表(sentence_phrases)

Name	Data Type	Primary	Is Key	Foreign Key	Description
sentence_id	int(11)	X	X	X	句子 id
phrases_id	int(11)	X	X	X	不可重复句子 id

- 句子结果(sentence_result)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		句子处理结果 id
correct_text	varchar(255)				处理结果文本
dnn	float				DNN 处理结果
init_status	bit(1)				是否处理过

is_negative	bit(1)				是否消极
is_neutral	bit(1)				是否中立
is_positive	bit(1)				是否积极
sentence_id	int(11)				句子 id

- 句子结果-修改后文本不同处位置中间表(sentence_result_correct_pos)

Name	Data Type	Primary	Is Key	Foreign Key	Description
sentence_result_id	int(11)		X	X	句子结果 id
correct_pos	int(11)				修改后的文本与原文本不同的位置

- 句子结果-修改前文本不同处位置中间表(sentence_result_org_pos)

Name	Data Type	Primary	Is Key	Foreign Key	Description
sentence_result_id	int(11)		X	X	句子结果 id
org_pos	int(11)				原文本与修改后的文本不同的位置

- 句子结果-情感倾向中间表(sentence_result_possibilities)

Name	Data Type	Primary	Is Key	Foreign Key	Description
sentence_result_id	int(11)		X	X	句子结果 id
possibilities	float				情感倾向

- 任务(task)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		任务 id
create_date	datetime				任务创建时间
dealing_date	datetime				任务处理时间
end_date	datetime				任务完成时间
failed	bit(1)				失败标记
finish	bit(1)				完成标记
join_date	datetime				加入等待队列时间
priority	int(11)				优先级
scheduled	bit(1)				是否加入批处理任务
type	varchar(255)				任务类型
article_id	int(11)		X	X	文章 id
file_id	int(11)		X	X	文件 id
result_id	int(11)		X		结果 id

user_id	int(11)		X		用户 id

- 任务记录(task_record)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		任务记录 id
msg	varchar(255)				记录信息
operation_type	varchar(255)				操作类型
task_id	int(11)				任务 id
time	datetime				记录时间

- 任务结果(task_result)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		任务结果 id
create_date	datetime				结果创建时间
success	bit(1)				成功标记
task_id	int(11)				任务 id
type	varchar(255)				结果类型
broken_sentences_count	tinyblob				不通顺句子计数
negative_emotions_count	tinyblob				消极句子计数
positive_emotions_count	tinyblob				积极句子计数
wrong_text_count	tinyblob				错误文本计数

- 任务结果-DNN 处理结果中间表(task_result_dnn_map)

Name	Data Type	Primary	Is Key	Foreign Key	Description
task_result_id	int(11)	X	X	X	任务结果 id
dnn_map	float				DNN 处理结果
dnn_map_key	int(11)	X	X		DNN 处理结果对应键值

- 任务结果-句子中间表(task_result_sentence_ids)

Name	Data Type	Primary	Is Key	Foreign Key	Description
task_result_id	int(11)		X	X	任务结果 id
sentence_ids	int(11)				句子 id

- 任务结果-句子结果中间表(task_result_sentence_result_map)

Name	Data Type	Primary	Is Key	Foreign Key	Description
task_result_id	int(11)	X	X	X	任务结果 id
sentence_result_map_id	int(11)		X	X	句子结果 id
sentence_result_map_key	int(11)	X	X		句子结果对应键值

用户(user)V

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		用户 id
account_non_expired	bit(1)				账号过期标识
account_non_locked	bit(1)				账号封禁标识
credentials_non_expired	bit(1)				证书过期标识
deleted	bit(1)				账号删除标识
enabled	bit(1)				账号激活标识
password	varchar(255)				密码
username	varchar(255)		X		用户名
user_auth_id	int(11)		X		用户认证 id
user_detail_id	int(11)		X		用户详细信息 id

- 用户权限(user_auth)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		权限 id
role	varchar(255)				角色名

- 用户详细信息(user_detail)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		详细信息 id

- 详细信息-最近文件中间表(user_detail_recent_file_ids)

Name	Data Type	Primary	Is Key	Foreign Key	Description
user_detail_id	int(11)		X	X	详细信息 id
recent_file_ids	int(11)				最近文件 id

- 详细信息-最近任务中间表(user_detail_recent_tasks)

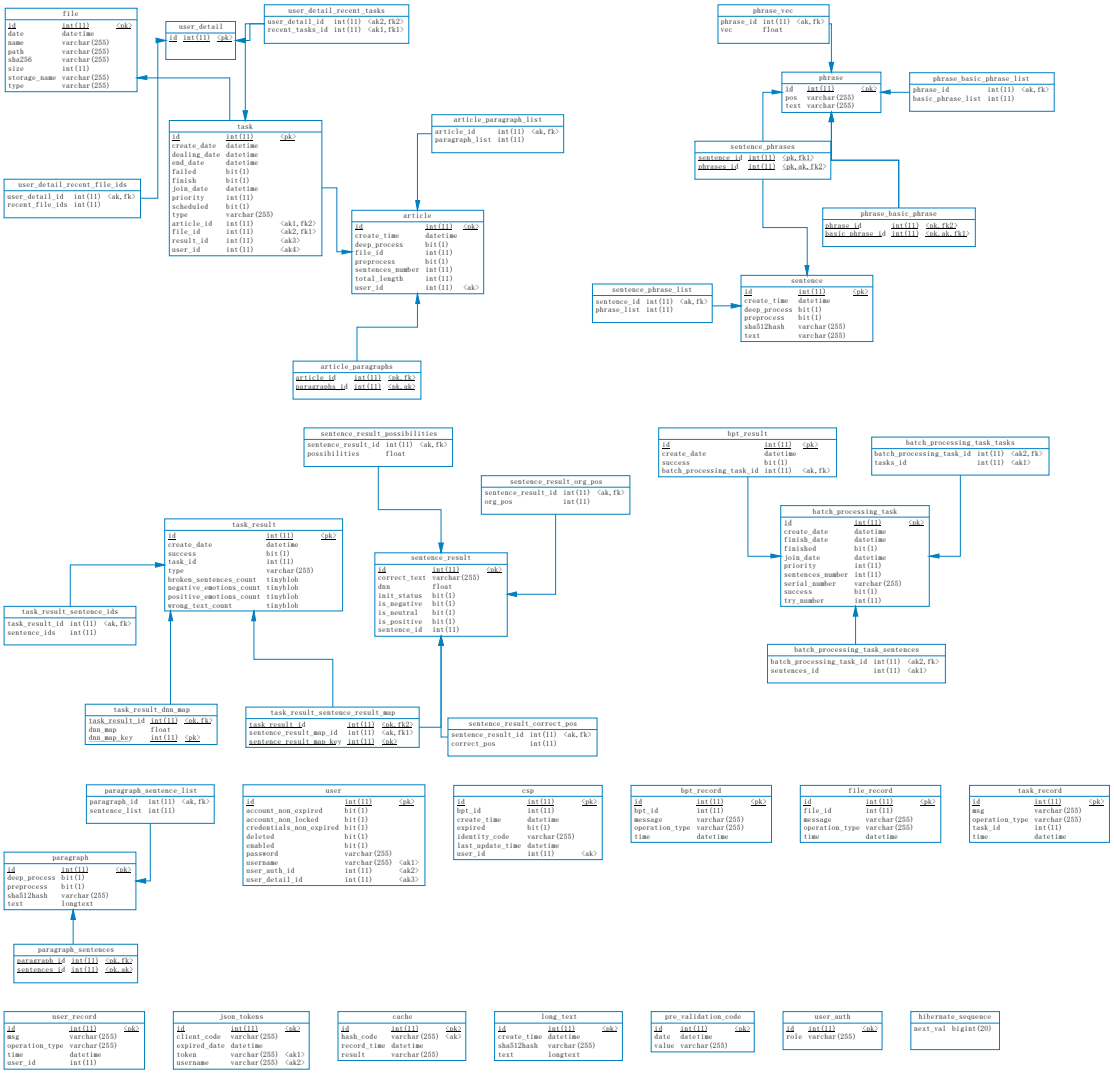
Name	Data Type	Primary	Is Key	Foreign Key	Description

user_detail_id	int(11)		X	X	详细信息 id
recent_tasks_id	int(11)		X	X	最近任务 id

- 用户记录(user_record)

Name	Data Type	Primary	Is Key	Foreign Key	Description
id	int(11)	X	X		用户记录 id
msg	varchar(255)				记录信息
operation_type	varchar(255)				操作类型
time	datetime				记录时间
user_id	int(11)				用户 id

6.2 数据库 ER 图


file
user_detail
user_detail_recent_tasks
task
user_detail_recent_file_ids
article_paragraph_list
article
article_paragraphs
phrase_vec
phrase
phrase_basic_phrase_list
sentence_phrases
phrase_basic_phrase
sentence_phrase_list
sentence
sentence_result_possibilities
sentence_result_org_pos
task_result
task_result_sentence_ids
sentence_result
bpt_result
batch_processing_task_tasks
batch_processing_task
batch_processing_task_sentences
task_result_dnn_map
task_result_sentence_result_map
sentence_result_correct_pos
paragraph_sentence_list
user
csp
bpt_record
file_record
task_record
paragraph
paragraph_sentences
user_record
json_tokens
cache
long_text
pre_validation_code
user_auth
hibernate_sequence

7 详细设计

7.1 账号认证服务

7.1.1 功能描述

用户认证服务主服务端用于接收客户端发来的认证请求（用户名，密码），并判断是否合法。这些功能包括：依据 openid 与预设密码创建新的账号、检验 openid 与密码是否正确、记录登录时的客户端代码（clientCode）给身份验证服务提供凭据、生成 token 用于客户端计算客户端签名。

7.1.2 性能描述

对于快应用端单次请求的发起到处理完毕，不超过 500ms。

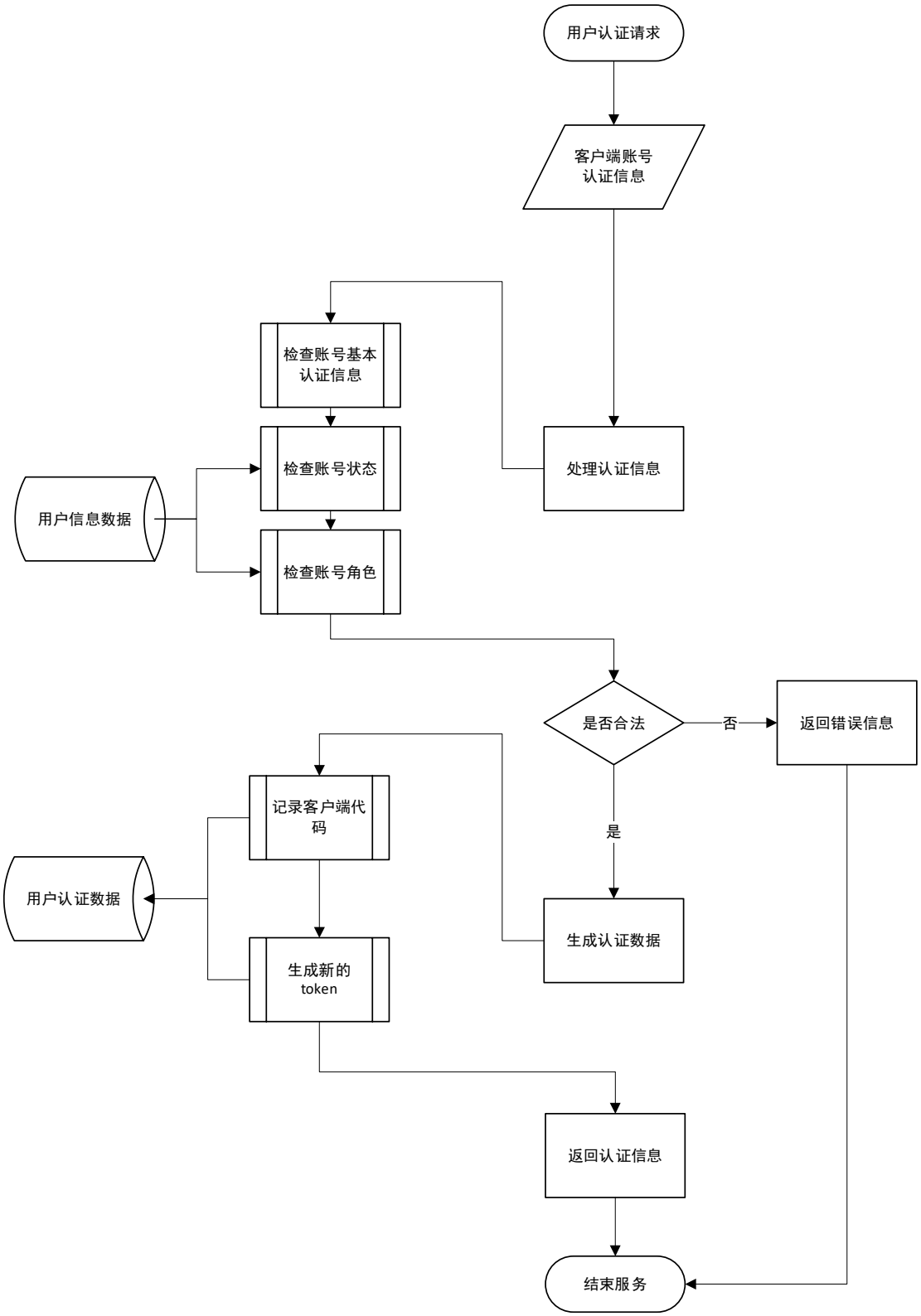
7.1.3 输入

通过 RESTful API 风格的 API 接口，传入 openid、密码、客户端代码（clientCode）。

7.1.4 输出

通过服务端返回 JSON 的形式输出新账号的信息或者 token。

7.1.5 程序逻辑


用户认证请求
客户端账号
认证信息
检查账号基本
认证信息
检查账号状态
处理认证信息
用户信息数据
检查账号角色
是否合法
否
返回错误信息
记录客户端代
码
是
用户认证数据
生成认证数据
生成新的
token
返回认证信息
结束服务

7.1.6 限制条件

输入输出均通过 Restful API 接口调用，且输入字段不能为空。

7.2 身份验证服务

7.2.1 功能描述

对于受保护的 API 接口的请求的身份检验功能，这些功能细分可包括对于记录受保护的 API 接口的 URL 地址（路由表）、拦截对于受保护的 API 接口的访问请求并进行过滤、对于客户端依据 token 计算出的客户端签名（signed）进行核验、为接口控制器提供用户身份信息。

身份验证服务能够阻挡对于系统 API 接口的恶意攻击，具备有基于角色的用户权限控制功能，能够依照用户所具备的权限对 API 接口进行区分。

7.2.2 性能描述

对于快应用端单次请求的发起到处理完毕，不超过 100ms。

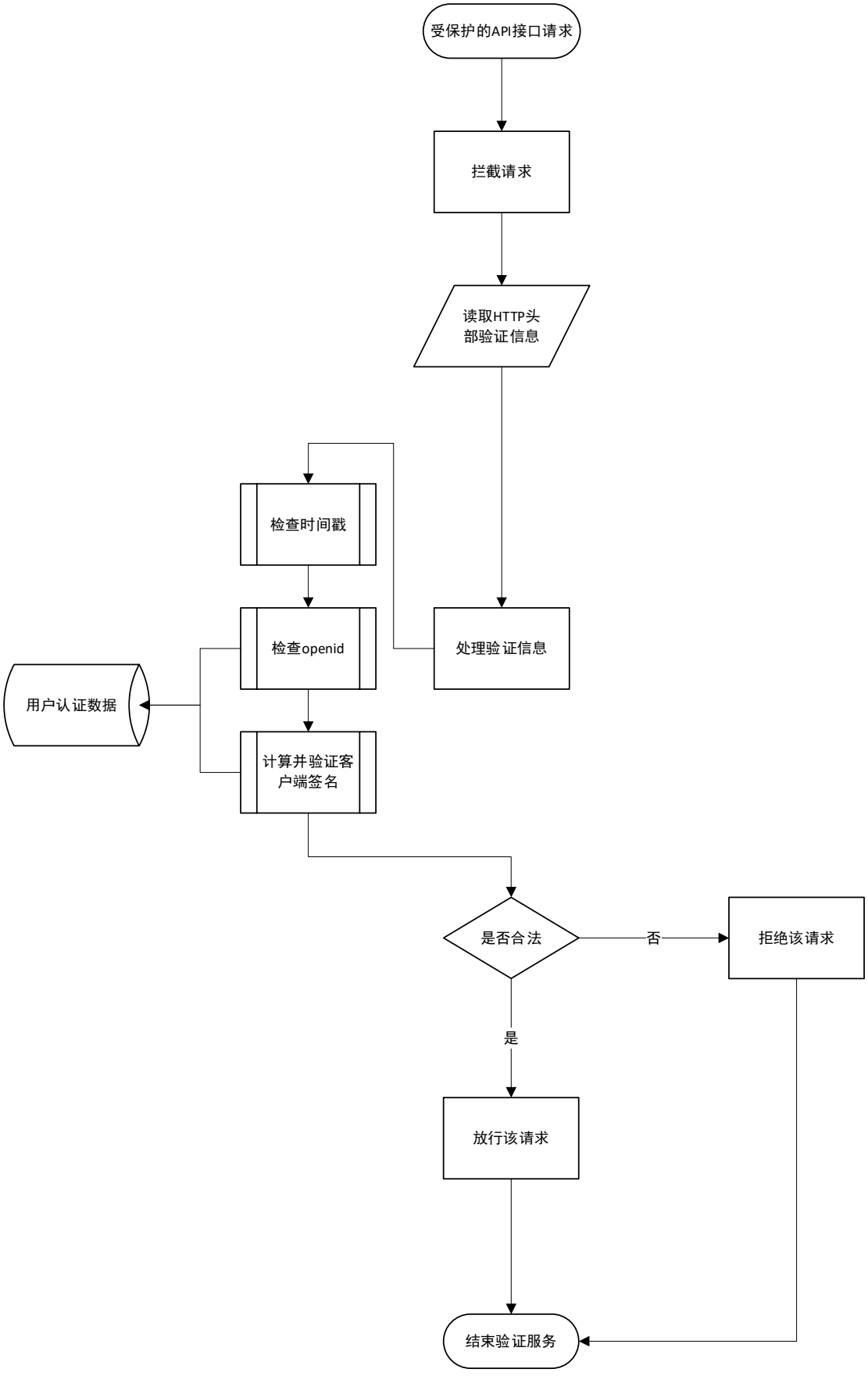
7.2.3 输入

通过 RESTful API 风格的 API 接口，传入 openid、客户端签名（signed）、时间戳。

7.2.4 输出

认证通过则放行请求，使得请求能够到达对应的控制器。认证失败则向快应用端返回相应的错误信息及错误代码。

7.2.5 程序逻辑


受保护的API接口请求
拦截请求
读取HTTP头
部验证信息
检查时间戳
检查openid
处理验证信息
用户认证数据
计算并验证客
户端签名
是否合法
否
拒绝该请求
是
放行该请求
结束验证服务

7.2.6 限制条件

由于功能特殊，需要面对多种情况，该功能模块在使用上无限制条件。

7.3 文件上传与散列值识别模块

7.3.1 功能描述

使得主服务端能够保存客户端能够上传文件数据及相应信息，并能够对文档文件（Word）进行解析。另外，该模块能够计算文件数据的散列值，使得文件服务能够找到并引用文件缓存。

7.3.2 性能描述

对于快应用端单次请求的发起到处理完毕，不超过 10s（存在文件大小的以及带宽的制约）。

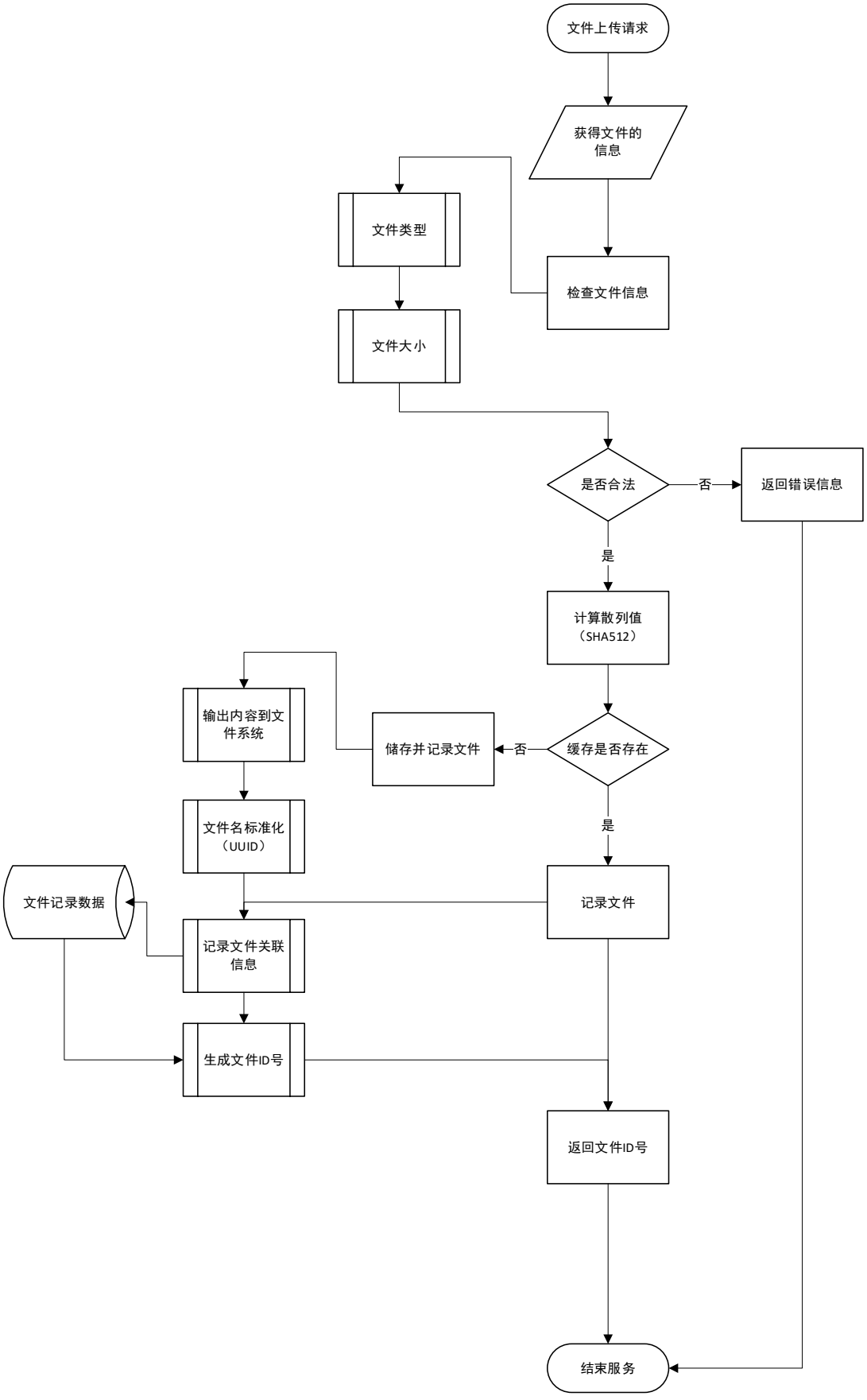
7.3.3 输入

按照接口要求上传文件。

7.3.4 输出

返回文件 ID 号。

7.3.5 程序逻辑


文件上传请求
获得文件的信息
文件类型
检查文件信息
文件大小
是否合法
否
返回错误信息
是
计算散列值（SHA512）
输出内容到文件系统
储存并记录文件
否
缓存是否存在
文件名标准化（UUID）
是
文件记录数据
记录文件
记录文件关联信息
生成文件ID号
返回文件ID号
结束服务

7.3.6 限制条件

仅支持单文件上传，且文件大小最大为（16MB），HTTP 请求的内容格式为 form-data，文件对应的 key 值 file。

7.4 子任务创建、自动预处理与自动预分析模块

7.4.1 功能描述

接受客户端的发来的子任务创建请求，并创建子任务管理结构。

异步调用预处理器与预分析器，并给它们分配线程。

自动对已经创建成功的子任务进行章结构、段结构、句结构的预处理，涉及在异步条件下在利用线程池获得空闲线程并启动预处理器。

自动对已经经过预处理的子任务进行基于词性、词向量、语句通顺程度的分析，涉及在异步条件下在利用线程池获得空闲线程并启动预分析器。

7.4.2 性能描述

从接收到客户端请求到服务端返回不超过 200ms（预处理过程异步执行，服务端可以立即返回）。

预处理过程所需的时间平均不超过 3 分钟，预分析过程平均不超过 15 分钟。

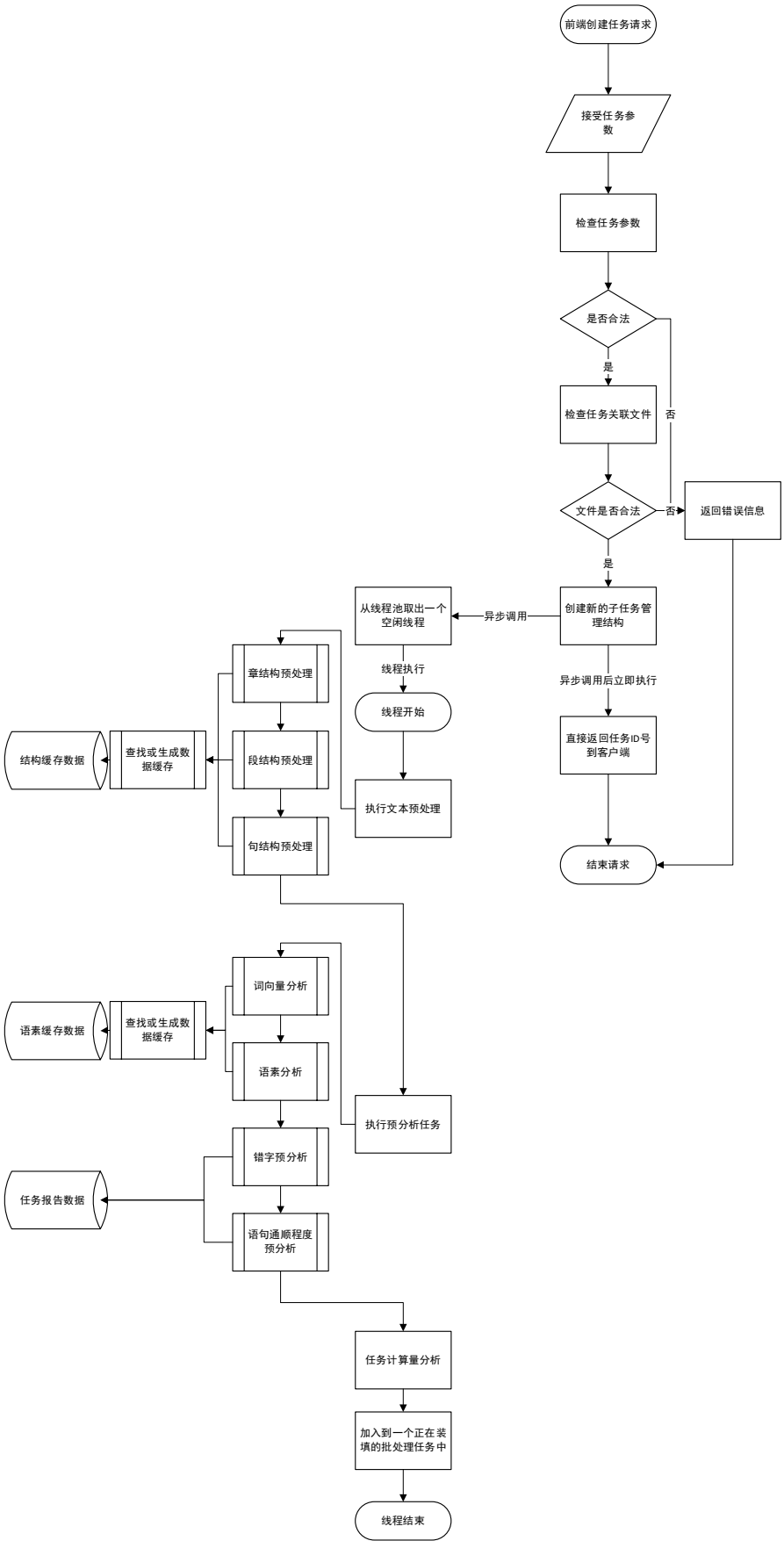
7.4.3 输入

通过 API 接口传入的文件 ID 号。

7.4.4 输出

子任务 ID 号。

7.4.5 程序逻辑


前端创建任务请求
接受任务参数
检查任务参数
是否合法
是
否
检查任务关联文件
文件是否合法
否
返回错误信息
是
创建新的子任务管理结构
异步调用
从线程池取出一个空闲线程
异步调用后立即执行
直接返回任务ID号到客户端
线程执行
线程开始
执行文本预处理
章结构预处理
段结构预处理
句结构预处理
查找或生成数据缓存
结构缓存数据
结束请求
词向量分析
语素分析
错字预分析
语句通顺程度预分析
查找或生成数据缓存
语素缓存数据
执行预分析任务
任务报告数据
任务计算量分析
加入到一个正在装填的批处理任务中
线程结束

7.4.6 限制条件

文件 ID 号有效，且对应文件为 Word 文档。

7.5 子任务自动批处理转换

7.5.1 功能描述

由于同一时间段可以有多个子任务预处理与预分析完成，该功能自动地将这短时间内的多个子任务合并成为批处理任务提供给计算端处理。

7.5.2 性能描述

单次定时任务运行时间不超过 500ms。

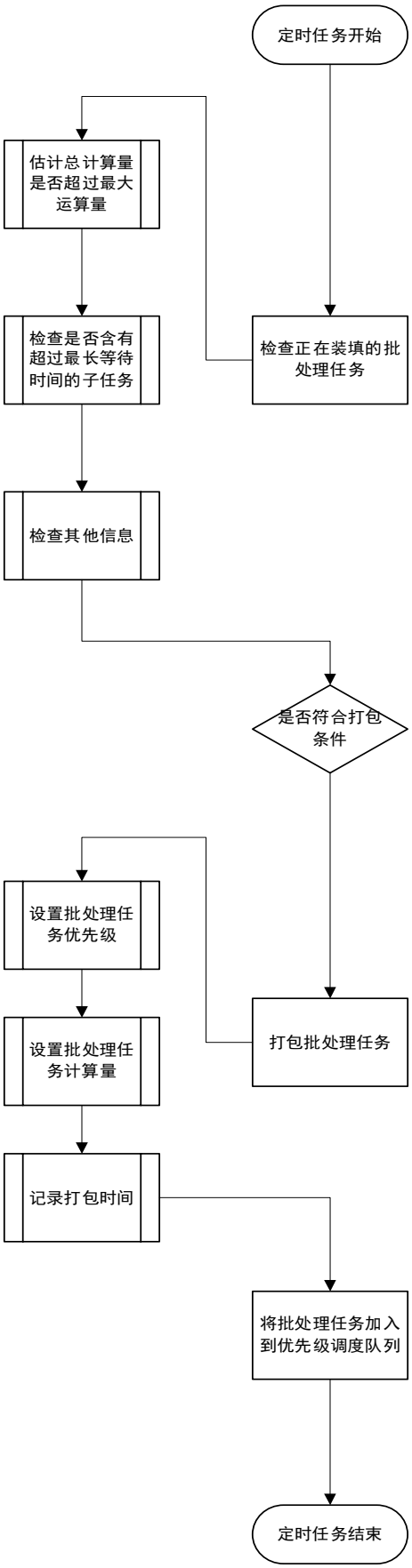
7.5.3 输入

该时间段内预处理与预分析完毕的子任务队列。

7.5.4 输出

符合执行条件的批处理任务。

7.5.5 程序逻辑


定时任务开始
估计总计算量是否超过最大运算量
检查是否含有超过最长等待时间的子任务
检查正在装填的批处理任务
检查其他信息
是否符合打包条件
设置批处理任务优先级
设置批处理任务计算量
打包批处理任务
记录打包时间
将批处理任务加入到优先级调度队列
定时任务结束

7.5.6 限制条件

合并称为批处理任务的子任务必须经过预处理与预分析，或者超过最长等待时间。

7.6 批处理任务自动调度模块

7.6.1 功能描述

提供批处理任务的自动调度，解决分布式并行环境下的调度冲突问题。为计算端分配合适的批处理任务并且使得高优先级的批处理任务尽可能快的处理完成。

以句的粒度估算批处理任务的工作量，为批处理任务的拆分、调度与分配提供重要依据。

按照计算端反馈自动拆分工作量过大的批处理任务，直到不可拆分为止。

7.6.2 性能描述

理论上单次定时任务处理的最长时间不超过 1s。对于接口调用，该模块进行调度或者锁定处理的最长时间不超过 50ms。

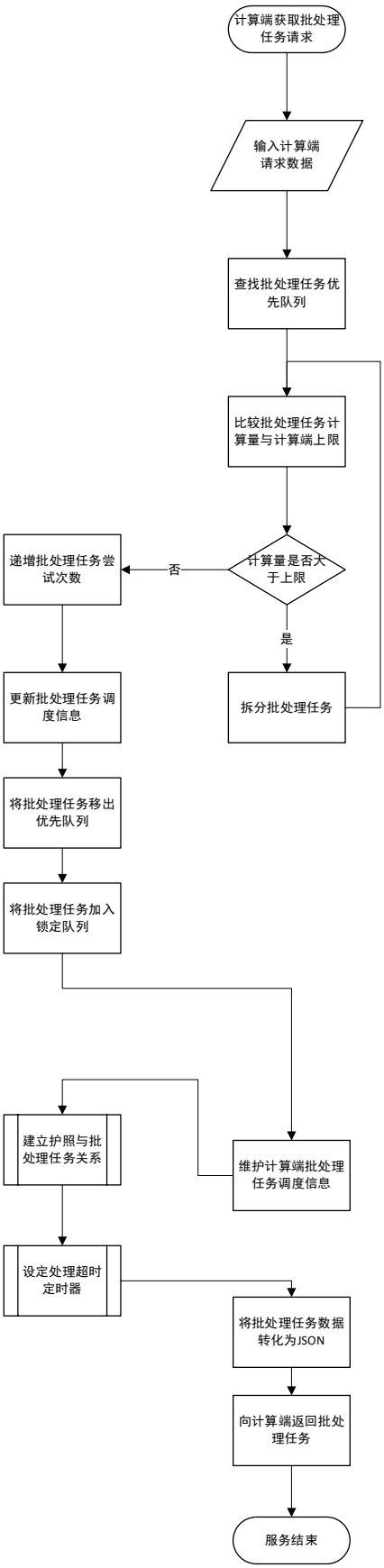
7.6.3 输入

批处理任务

7.6.4 输出

批处理任务（到计算端）

7.6.5 程序逻辑


计算端获取批处理任务请求
输入计算端请求数据
查找批处理任务优先队列
比较批处理任务计算量与计算端上限
计算量是否大于上限
否
递增批处理任务尝试次数
是
拆分批处理任务
更新批处理任务调度信息
将批处理任务移出优先队列
将批处理任务加入锁定队列
建立护照与批处理任务关系
维护计算端批处理任务调度信息
设定处理超时定时器
将批处理任务数据转化为JSON
向计算端返回批处理任务
服务结束

7.6.6 限制条件

批处理任务在创建时必须设置优先级，计算端在获得批处理任务后必须锁定该批处理任务以免丢失。

7.7 批处理任务结果处理模块

7.7.1 功能描述

结合计算端返回的模型计算结果与预分析处理结果，生成与批处理任务结果相适应的数据结构，使得数据能够储存到数据库中。并且，该功能模块负责根据批处理任务结果生成对应的子任务的结果，使得客户端能够获得子任务的处理结果。

7.7.2 性能描述

从计算端提交任务结果到相关联的子任务结果数据处理完毕不超过 500ms。

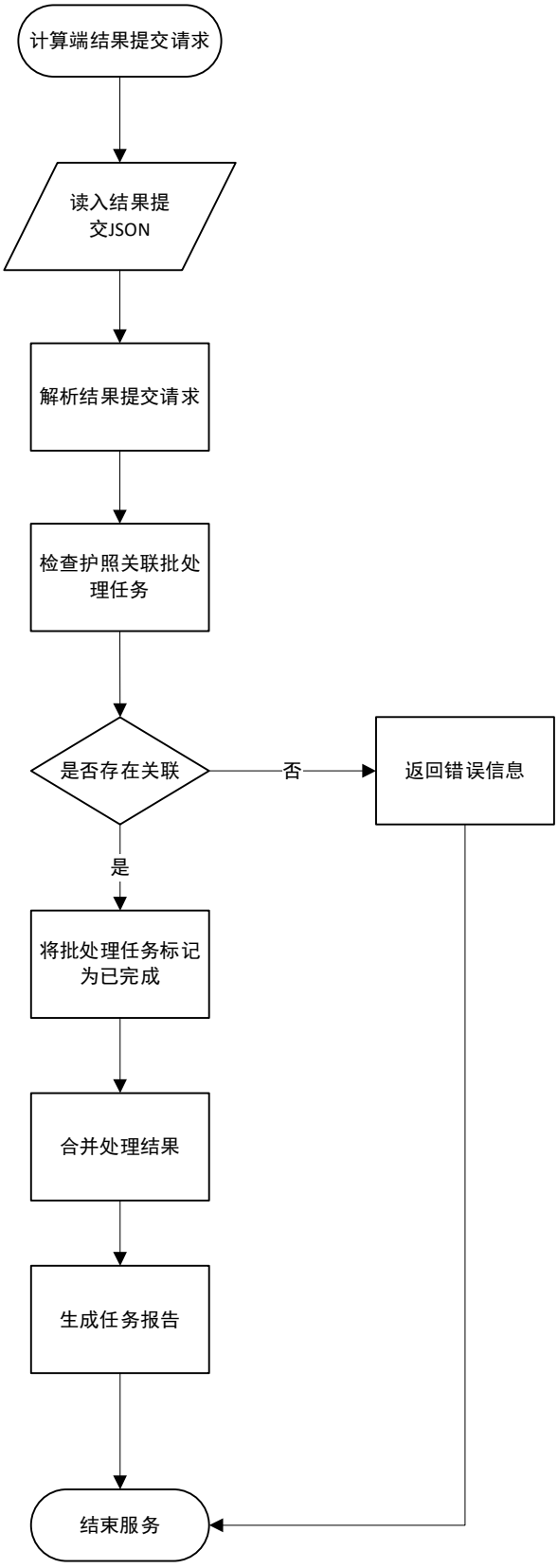
7.7.3 输入

批处理任务 ID 号、批处理任务执行结果（计算端模型预测结果）

7.7.4 输出

一个或多个子任务执行结果数据对象。

7.7.5 程序逻辑


计算端结果提交请求
读入结果提交JSON
解析结果提交请求
检查护照关联批处理任务
是否存在关联
否
返回错误信息
是
将批处理任务标记为已完成
合并处理结果
生成任务报告
结束服务

7.7.6 限制条件

执行结果及所对应的批处理任务 ID 号必须有效。

7.8 子服务器自动注册与签证更新模块

7.8.1 功能描述

提供分布式 GPU 计算端（以下称为计算端）的注册功能，并给相应的计算端创建并分配护照。

提供计算端的签证的更新功能，以此确认计算端的在线状态，防止计算端护照因为过久未更新签证而被注销。

7.8.2 性能描述

所有服务从接收计算端请求到返回周转时间不超过 200ms。

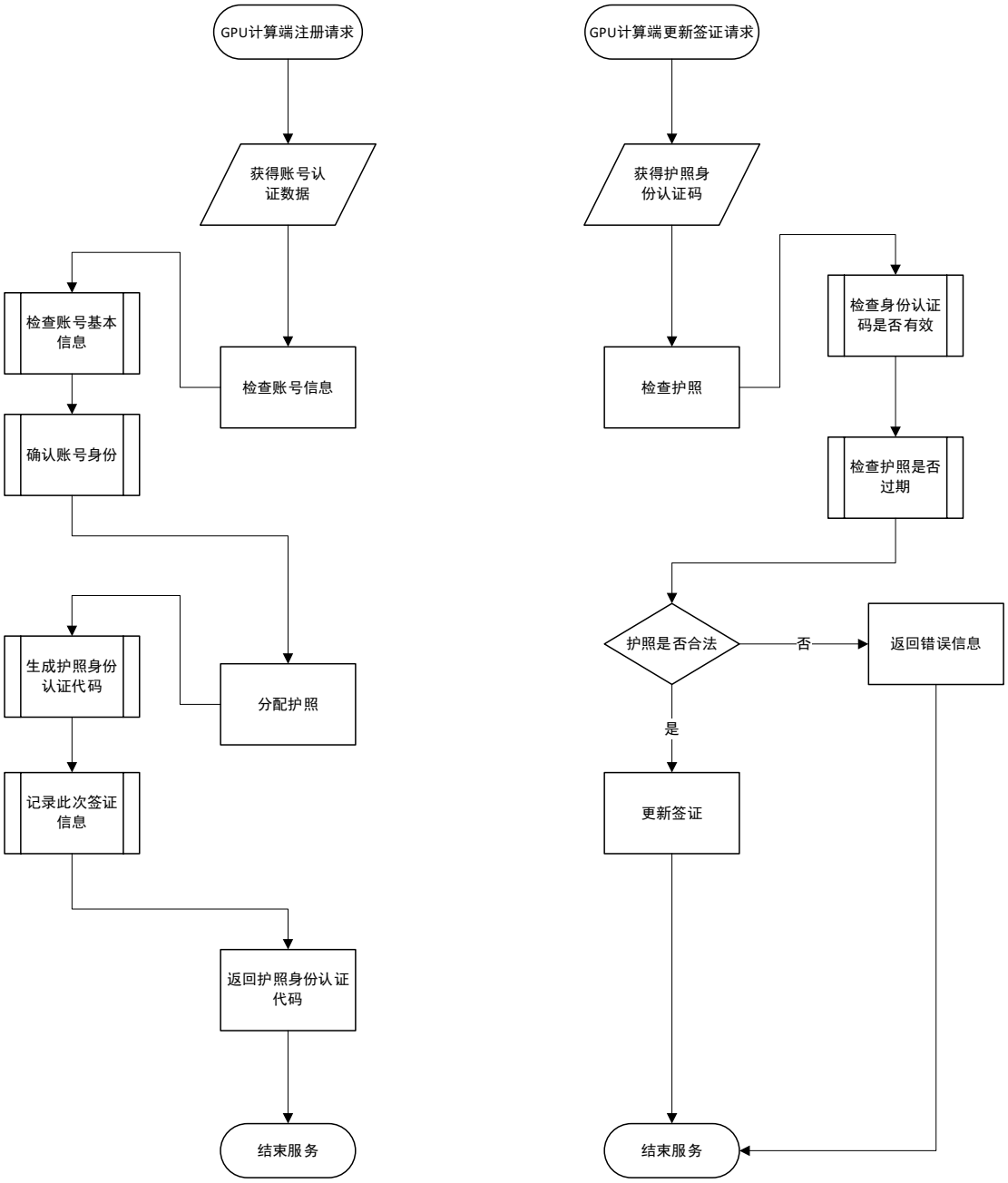
7.8.3 输入

子服务器账号 openid、客户端代码 clientCode、子服务器护照身份认证码 idcode。

7.8.4 输出

子服务器护照信息。

7.8.5 程序逻辑



7.8.6 限制条件

账号经身份认证的结果必须为子服务器类型，客户端代码必须遵循约定。

7.9 子服务器护照自动管理模块

7.9.1 功能描述

提供子服务器的护照的管理功能，查找签证超过一段时间为更新的护照并注销。对于绑定了批处理任务的护照，该功能解锁相应的批处理任务并重新将该批处理任务加入优先级队列。

7.9.2 性能描述

单次定时任务的运行时间最长不超过 500ms。

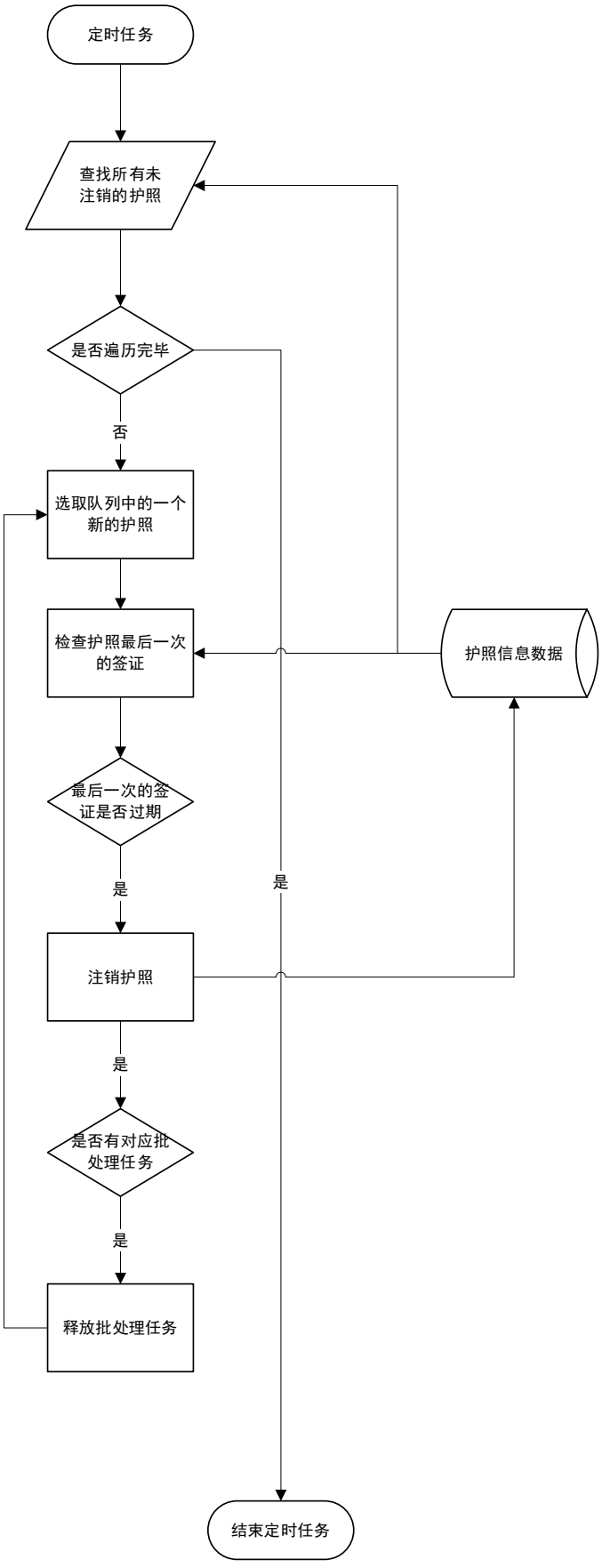
7.9.3 输入

子服务器护照身份认证码 idcode。

7.9.4 输出

无对外输出。

7.9.5 程序逻辑


定时任务
查找所有未注销的护照
是否遍历完毕
否
选取队列中的一个新的护照
检查护照最后一次的签证
护照信息数据
最后一次的签证是否过期
是
是
注销护照
是
是否有对应批处理任务
是
释放批处理任务
结束定时任务

7.9.6 限制条件

子服务器护照身份认证码必须有效。