软件缺陷修复报告

基于Flask的深度学习自动化部署系统

|  |  |  |  |
| --- | --- | --- | --- |
| 编写 | 许京爽，崔昕宇，张利鹏，聂磊 | 日期 | 2020年5月20日 |
| 校对 | 聂磊 | 日期 | 2020年5月20日 |

北京航空航天大学 计算机学院

软件工程综合实验课 C组

二〇二〇年五月二十日

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 |
| 1.0.0 | 2020.5.19 |  | 完成第一版 | 崔昕宇，张利鹏，许京爽，聂磊 |
| 1.1.0 | 2020.5.20 | 1 | 新增错误类型说明和错误处理流程 | 聂磊 |
|  |  |  |  |  |

**目录**

[1 简介 1](#_Toc40870644)

[1.1 目的 1](#_Toc40870645)

[1.2 软件缺陷管理过程 1](#_Toc40870646)

[2 用例104问题报告与修复情况 2](#_Toc40870647)

[2.1 问题说明 2](#_Toc40870648)

[2.2 问题原因 3](#_Toc40870649)

[2.3 问题修复 3](#_Toc40870650)

[3 用例109问题报告与修复情况 3](#_Toc40870651)

[3.1 问题说明 3](#_Toc40870652)

[3.2 问题原因 4](#_Toc40870653)

[3.3 问题修复 5](#_Toc40870654)

[4 用例113问题报告与修复情况 5](#_Toc40870655)

[4.1 问题说明 5](#_Toc40870656)

[4.2 问题原因 5](#_Toc40870657)

[4.3 问题修复 6](#_Toc40870658)

[5总结 6](#_Toc40870659)

[参考文献 6](#_Toc40870660)

基于Flask的深度学习自动化部署系统

软件缺陷修复报告

**关键词：**深度学习，数据即服务（DaaS）

**摘要：**本文描述了2020春季学期《软件工程综合实验》课程中，C组成员开发的深度学习自动化部署系统的软件缺陷修复报告。

**专业术语清单：**

|  |  |
| --- | --- |
| **专业术语** | **中文解释** |
| SRS | SRS指软件需求规格说明书（Software Requirements Specification），是软件需求设计阶段的产物，描述了软件所需满足的需求。 |
| Python | Python是一种跨平台的计算机程序设计语言，具有面向对象和动态类型的特性。Python最初被设计用于编写自动化脚本，随着版本的不断更新和语言新功能的添加，越多被用于独立的、大型项目的开发。 |
| URL | 统一资源定位系统（Uniform Resource Locator）是因特网的万维网服务程序上用于指定信息位置的表示方法。 |
| 深度学习 | 深度学习是机器学习领域中一个新的研究方向，通过学习样本数据的内在规律和表示层次，让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。 |
| 模型部署 | 将训练完毕的深度学习模型放置在特定硬件平台上，辅以合适的软件环境，使该模型能够被用户使用。 |
| HTTP | 超文本传输协议（英文全称：HyperText Transfer Protocol）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。 |
| REST API | 表现层状态转换（英文全称：Representational State Transfer）是一种万维网软件架构风格，目的是便于不同软件/程序在网络中互相传递信息。表现层状态转换是根基于超文本传输协议之上而确定的一组约束和属性，是一种设计提供万维网络服务的软件构建风格。 |

# 1 简介

## 1.1 目的

本文档是C组开发项目 ***基于Flask的深度学习自动化部署系统*** 的软件缺陷修复报告，通过对项目代码的评估，在明确测试需求的基础上，进一步完成对软件缺陷的修复。

本轮软件缺陷修复过程以软件测试报告及软件源代码为输入，分析软件存在的问题，结合软件工程综合实验具体要求，输出错误修复报告。本文档是为软件开发人员、测试人员、项目验收方编写的，可作为测试过程的评价指标和项目验收达标的依据。

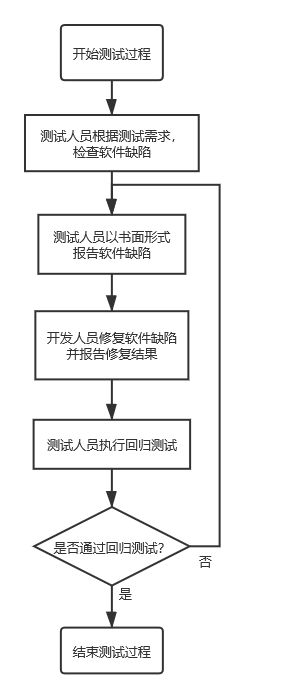
## 1.2 软件缺陷管理过程

测试人员根据预先设计的测试用例执行测试过程，充分验证软件的功能是否满足预期，寻找可能存在的软件缺陷。具体而言，软件缺陷的严重程度分为三个级别：严重、中等、轻微，其具体描述如表1所示。

**表1 软件缺陷严重程度分类**

|  |  |
| --- | --- |
| **严重程度** | **缺陷造成的影响** |
| 严重 | 软件主要功能失效  软件运行结果与预期有大量偏差  软件性能大幅下降  用户数据丢失  信息泄露 |
| 中等 | 软件次要功能失效  软件运行结果与预期有轻微偏差  软件性能中度下降 |
| 轻微 | 软件性能轻微下降  影响产品易用性 |

当发现软件缺陷后，测试人员以书面形式告知开发人员软件缺陷的相关信息：测试环境、测试过程、测试用例编号、软件缺陷表现。开发人员修复软件缺陷后，以软件缺陷修复报告的形式将修复结果反馈给测试人员，以供确认。测试人员对产品进行回归测试，确保软件缺陷已被修复。图1展示了项目的软件缺陷管理流程。



**图1 软件缺陷管理流程示意图**

# 2 用例104问题报告与修复情况

## 2.1 问题说明

**表2 用例104问题说明**

|  |  |
| --- | --- |
| 标题 | 用户新建项目下创建同名项目的异常处理 |
| 测试环境 | Windows10，Python3.7 |
| 准备工作 | 安装好Flask及系统运行环境 |
| 步骤 | 1. 在个人页面中，点击新建项目； 2. 跳转到创建项目页面中，输入项目名称、项目路由和项目描述，其中输入的项目名称为已存在项目的名称； 3. 点击创建按钮。 |
| 预期结果 | 提示已存在同名项目，无法创建 |
| 实际结果 | 显示请求超时，请重试 |
| 严重程度 | 轻微 |

## 2.2 问题原因

后端在接收到前端传递的所需创建的项目信息后，返回值出现了返回异常，多返回了一个html页面，导致和前端的不匹配，前端没有得到需要返回已存在同名项目的约定代码，无法给出正确的错误提示。

## 2.3 问题修复

后端按照约定返回给前端所需的code和msg，前端得到约定代码后，提示已存在同名项目，无法创建，修复完毕。



**图 2 用例104错误代码修改**

# 3 用例109问题报告与修复情况

## 3.1 问题说明

**表3 用例109问题说明**

|  |  |
| --- | --- |
| 标题 | 模型配置中变量的数据类型bug |
| 测试环境 | Ubuntu 18.04，Python 3.7(& Anaconda Python 3.7)  TensorFlow v2.1.0 (& v1.14.0 in Anaconda Python 3.7)  PyTorch v1.5.0 |
| 准备工作 | 安装好Flask及系统运行环境 |
| 步骤 | 1. 上传模型文件压缩包，选择模型类型为cpkt； 2. 导入该模型后，在模型参数页面设置内存限制为300MB，输入节点为mnist\_image，输出节点为result； 3. 保存以上设置后，启动该模型的实例。 |
| 预期结果 | 页面返回设置成功，更新指定模型的参数配置，可成功地重新启动模型实例 |
| 实际结果 | 尝试启动实例，浏览器显示“服务器错误，请检查参数”错误提示 |
| 严重程度 | 严重 |



**图3 浏览器显示“服务器错误，请检查参数”错误提示**

## 3.2 问题原因

按照设计预期，此时实例应该处于运行状态，因为300MB对于该模型的运行来说已经足够了，且输入输出节点的名称也与训练时设定的一致，所以可能问题出在代码实现上。

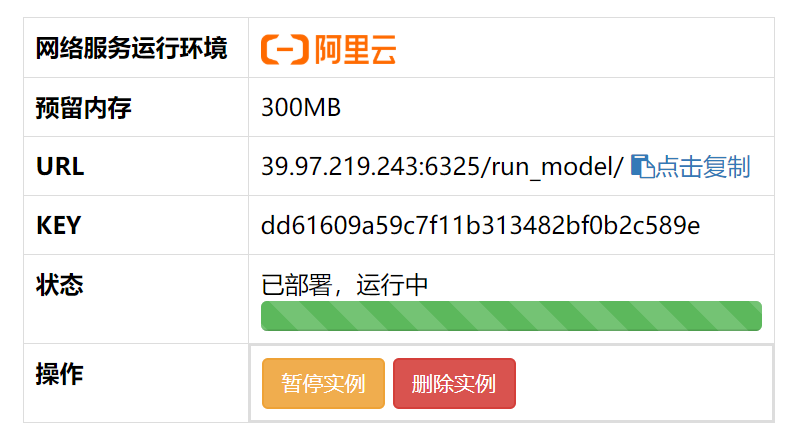
在控制台打印相关变量、设置断点后将错误范围锁定在了内存检查函数内，经过进一步判断，发现在模型flask程序运行成功后的内存检查函数处出现了异常。遂将此处的函数参数输出至控制台以甄别错误：



**图4 函数参数输出结果**

可以看到，此处内存限制变量的类型是str，所以才会在后续的内存大小比较中报出异常，引发这一异常的原因应该是从yml文件读取配置项时未对其进行类型的转换，这一错误直接导致带有参数的模型无法部署，是较为严重的一个软件错误。

## 3.3 问题修复

经过对配置项类型的转换和异常情况的判断与处理，最终这一软件错误被成功修复，部署成功的截图如图所示：

**图5 部署成功的页面截图**

# 4 用例113问题报告与修复情况

## 4.1 问题说明

**表4 用例113问题说明**

|  |  |
| --- | --- |
| 标题 | 删除实例后配置文件与数据库信息不对等问题 |
| 测试环境 | Windows10，Python3.7 |
| 准备工作 | 安装flask框架、flask\_login、flask-sqlalchemy插件  安装yaml和ruamel.yaml插件，能够实现对yaml文件的操作  配置TensorFlow v2.1.0&v1.14.0  配置pyTorch v1.5.0 |
| 步骤 | 1. 在查看模型详情界面点击“启动实例”； 2. 点击暂停实例使实例可以被删除； 3. 点击删除实例，跳转到设置部署参数界面； 4. 确认设置参数界面中，各参数都被清空。 |
| 预期结果 | 在用户没有对参数修改的情况下，再次点击“启动实例”，应当按照系统的默认参数运行实例 |
| 实际结果 | 在用户没有对参数修改的情况下，再次点击“启动实例”，按照用户上一次设置的参数运行了实例，此时用户并不知情 |
| 严重程度 | 中等 |

## 4.2 问题原因

运行实例需使用保存在系统中的配置文件config.yml，该文件保存了运行实例需要的一些参数。

在删除实例后，数据库实例表对应参数信息不可被查询，此时在设置运行参数界面，用户针对该实例设置的参数被清空，但此时保存在系统中的配置文件config.yml没有更新。

若此时用户再次部署模型生成实例，则使用的是用户上一次设置过的参数，而用户可能以为是用系统默认参数运行的。

## 4.3 问题修复

在删除实例接口中，调用editConfig方法，增加对config.yml文件的操作，将文件中的参数设置与数据库保持一致。通过以上修改即可完成对问题的修复。

# 5总结

本文档描述了C组开发项目 ***基于Flask的深度学习自动化部署系统*** 的软件缺陷修复情况，本次软件测试环节共修复了3个已发现的软件缺陷，提升了软件质量。接下来软件产品可交付评审小组，进行下一轮的测试评审工作。

# 参考文献

1. 邹欣.构建之法[M].北京.人民邮电出版社,2015:264-296
2. 软件工程测试方法[EB/OL]. <https://blog.csdn.net/fightfaith/article/details/47858101>