IRBL迭代二：软件详细设计

特别声明：内部资料，请勿传播

更新历史：

更新日期	更新原因	责任人
2021.4.2	创建文档	韩禧
2021.4.5	更新文档总体设计	韩禧
2021.4.8	更新文档数据结构说明和接口设计	韩禧
2021.4.13	更新各模块的存储要求	程荣鑫

1. 引言

1.1 编写目的和范围

编写目的：完善《IRBL迭代二：软件概要设计》的软件设计细节，将设计落实到编码层面，发挥设计文档的指导作用。

范围：本文档包含IRBL项目迭代二中的全局数据结构、总体设计、模块设计、接口设计和数据传输设计。受众为软工三团队KhyYYDS小组的全体成员。

1.2 术语表

术语	含义
IRBL	基于信息检索的缺陷定位系统

1.3 参考资料

《IRBL迭代二：软件概要设计》、《IRBL迭代一：项目启动》以及moodle上的IRBL基于信息检索的错误定位材料集

1.4 使用的文字处理和绘图工具

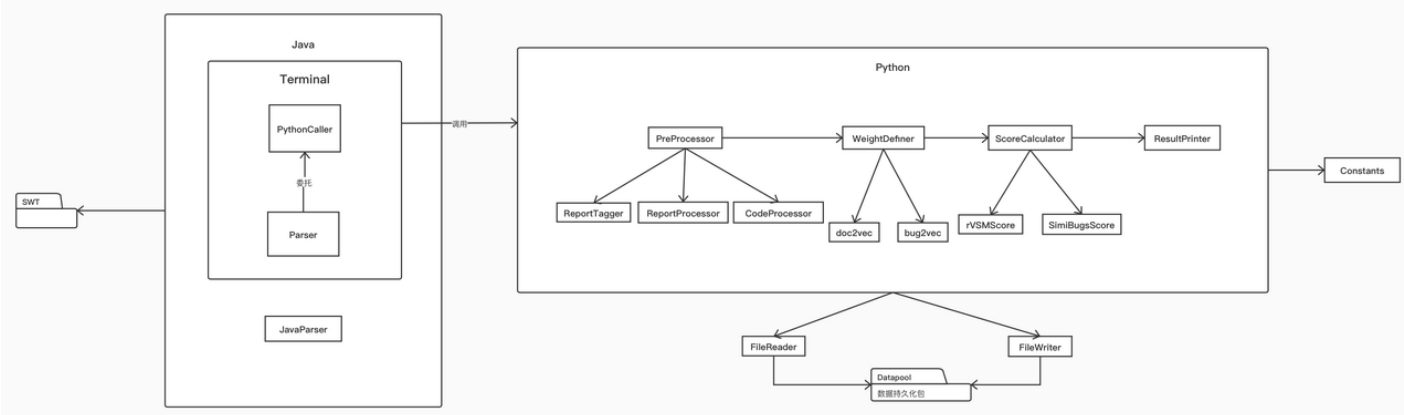
绘图：ProcessOn

文字处理：飞书云文档

2. 全局数据结构说明

数据结构	类型	含义	持久化格式
word2idx	dict<str, int>	对词语集进行编号的结果	json
doc2vec	dict<str, list>	每篇文本对应的词向量	json
simibugs_score	dict<str, dict<str, float>>	bug报告和代码文件的相似度	json
rvsm_score	dict<str, dict<str, float>>	rvsm模型处理后的得分	json
g_dct	dict<str, int>	代码文件的调和值g(#terms)	json
reports_with_tag	dict<str, dict<str, str>>	对bug报告进行解析，分为NL,SC,ST三部分	json
stack_list	dict<str, list[str]>	根据栈信息得出的bug报告与代码文件之间的关联	json
bug2vec	dict<str, list[float]>	将bug报告向量化处理	json
final_score	dict<str, dict<str, float>>	最终的各代码文件得分	json

3. 总体设计



总体设计相比概要设计文档中的设计内容大体没有发生变化，仍然是流水线式设计，只是将原本图中各Python模块的功能模块列出，以及与File System中的交互细化为FileReader与FileWriter模块和Datapool文件夹的交互，Python模块统一通过FileReader与FileWriter存取Datapool中的数据文件。

4. 模块设计

考虑的实现的便利性，模块设计和《IRBL迭代二：软件概要设计》中的模块设计稍有出入，如有冲突，请以**本文档**的的设计为准。

- Parser

  职责：解析命令，委托PythonCaller执行相应程序

  支持命令（同时支持简写命令）：

  preprocess(简写：p, /p): 调用code_preprocessor.py和bug_preprocessor.py，执行文本预处理任务

  defineWeight/define weight(简写：dw, /d): 调用weight_definer.py，执行权重计算和文本向量化的任务

  calculateSimilarity(简写：cs, /c): 调用simi_calculator.py，执行相似度计算任务

  printResult(简写：pr, /pr): 调用result_printer.py，执行运行结果打印的任务

  doAll(简写：all, /a): 依次调用code_preprocessor.py, bug_preprocessor.py, weight_definer.py, simi_calculator.py, result_printer.py，即完整地走完流水线

  exit(简写：q): 退出IRBL主程序

- PythonCaller

  职责：根据Parser的要求调用相应的python程序

- PreProcessor

  职责：执行预处理任务

  包含report_tagger.py, code_preprocessor.py和bug_preprocessor.py

  预处理后的文本数据按文件名/报告id存储为.txt

- WeightDefiner

  职责：执行词语权重计算任务，并将文档向量化

  由weight_definer.py实现，得到两个结果：doc2vec.json, bug2vec.json

  向量化的结果用json存储，即doc2vec.json

- ScoreCalculator

  职责：执行相似度计算任务，并把结果保存下来

  由simi_calculator.py实现，得到两个结果：rVSMScore, SimiBugsScore，并生成最终结果FinalScore，分别存储为rvsm_scor.json, simibugs_score.json, final_score.json

- ResultPrinter

  职责：输出相似度计算的结果及相关指标

  由result_printer.py实现

- FileReader

  职责：读取Datapool中文件的内容

  由file_reader.py实现，其中包含基类FileReader，派生类JSONReader和NpyReader

- FileWriter

  职责：向Datapool中文件写入内容

  由file_writer.py实现，其中包含基类FileWriter，派生类JSONWriter和NpyWriter

5. 接口设计

5.1 Parser

供接口：无

需接口：

1. PythonCaller(String file, String[] args)
2. PythonCaller.exec()

5.2 PythonCaller

供接口：

1. PythonCaller(String pyPath, String[] args)

   pyPath为调用的脚本所在路径，args为传递的参数，对调用信息进行初始化
2. PythonCaller.exec()

   在初始化完成的前提下，对相应Python脚本进行调用

需接口：

1. code_processor.main
2. bug_processor.main
3. weight_definer.main
4. simi_calculator.main
5. result_printer.main

5.3 PreProcessor

供接口：

1. code_processor.main

   对代码文件进行预处理，并存储为JSON文件
2. bug_processor.main

   对bug报告进行预处理，并存储为JSON文件
3. report_tagger.main

   对bug报告进行解析，将不同类型的内容加上相应tag（有NL,SC,ST三种），并存储为JSON文件

需接口：

1. FileWriter.writeFile

   PreProcessor模块需要使用FileWriter写入txt文件

5.4 WeightDefiner

供接口：

1. weight_definer.main

   包括以下子流程：

   get_words

   获取某个文件中的所有单词

   get_word_count

   获得某个文件中每个单词以及出现的次数

   get_g

   根据文档大小加权，帮助大文档有更高排名，得到dict格式的结果并存为JSON文件

get_tfidf

计算各代码文件的tf-idf结果，得到dict格式的结果并存为JSON文件

需接口：

1. FileReader.readFile()
   WeightDefiner需要用到FileReader读取.txt文件
2. FileReader.writeFile()
   WeightDefiner需要用到JSONWriter

5.5 ScoreCalculator

供接口：

1. score_calculator.main
   包括以下子流程：

   get_cos_sim

   计算相似度余弦值

   load

   进行预处理结果的加载，包括预处理后的报告文件及预处理后的代码文件

   get_rvsm_score

   结合 tf-idf 以及 g 的计算结果，得到rVSM模型的最终结果，并存为JSON文件

   get_simibugs_score

   根据预处理的结果，计算历史相似Bug报告并得到对应代码文件的得分，并存为JSON文件

   get_final_score

   计算加权后的最终得分情况，并将所有代码文件的得分存为JSON文件

需接口：

1. FileReader.readFile()
   SimiCalculator需要用到JSONReader
2. FileWriter.writeFile()
   SimiCalculator需要用到JSONWriter

5.6 ResultPrinter

供接口：

1. result_printer.main

   包含以下子流程：

   get_top_K

   获取与每个BUG报告前K个最相关的代码文件名（不含.java后缀）

   计算公式为

   $$Top@K = (|R_k|)/n \quad \cdots\cdots (1)$$

   print_top_K

   输出与每个BUG报告前K个最相关的代码文件名（不含.java后缀）

   getMRR

   计算MRR（首位倒排均值）指标，计算公式为

   $$MRR = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{rank_j} \quad \cdots\cdots (2)$$

   式（2）中 $rank_j$ 表示第j个缺陷报告所对应的列表中第一个与缺陷报告相关的源码的排名，并返回计算结果

   getMAP

   计算MAP（平均准确率均值）指标，计算公式为

   $$MAP = \frac{1}{n}\sum_{j=1}^{n}AvgP_j \quad \cdots\cdots (3)$$

   $$AvgP_j = \frac{1}{|k_j|}\sum_{k\in K_j}\{Prec@k\} \quad \cdots\cdots (4)$$

   $$Prec(k) = \frac{\sum_{i=1}^{k} IsRelevant(k)}{k} \quad \cdots\cdots (5)$$

   其中，式（4）的 $Prec@k$ 表示 l 的前 k 个文件中与缺陷报告相关的源码文件的比例， K_j 表示列表l中所有与缺陷报告相关的源码文件位置集合；式（5）的 $IsRelevant(k)$ 为1表示列表l中第k个源码文件与缺陷报告相关，否则无关

print_metrics

输出top1、top5、top10、MRR、MAP指标

需接口：

1. FileReader.readFile()

   ResultPrinter需要用到XlsReader和JSONReader

5.7 FileReader

供接口：

1. FileReader.readFile()：读取文件，返回文件内容

   由FileReader的子类提供不同的实现，JSONReader返回dict，NpyReader返回ndarry，FileReader本身有默认实现（返回str）

需接口：无

5.8 FileWriter

供接口：

1. FileWriter.writeFile()：向文件写内容

   由FileWriter的子类提供不同的实现，JSONWriter写入dict，存为json文件；NpyWriter写入ndarry，存为.npy文件

需接口：无

6. 数据传输设计

数据传输主要发生在python包中，考虑到模块间数据传输的量非常大，我们采用文件读写的方式存取数据。

1. 所有的数据文件都在src/main/python/irbl/Datapool文件夹中
2. python程序统一通过file_reader.py和file_writer中的接口读取数据