Lab2 实验报告

1. 思路概述

我们小组以 Eclipse 官网提供的 bug 缺陷报告数据作为原材料，将这些 bug 条目视为各种需求，进行分析。

首先利用爬虫技术，从网站上获取了 10000 份 bug 数据作为分析的源数据。

其次利用 python 的 pandas 包对获取的数据进行清洗和规范化处理，再利用自然语言处理库 nltk 对 bug 的文本进行分词处理，统计高频名词/动词。

最后利用通过筛选 Comp 属性以及 Abstract 属性中的关键词，对每个关键词进行评分，反过来利用这些关键词对每一条 bug/需求进行综合评分，根据最终评分对所有 bug 进行等级评定，将模型评定结果与官方评定结果进行对比以完成模型检验。

2. 数据爬取

2.1 数据来源：https://bugs.eclipse.org/bugs/

2.2 网站分析：

要针对该网站进行大规模数据获取，首先要进行该网站页面源码的分析。该网站已经提供了十分规范的表格式数据，其中我们较为关注的属性是 bug 报告的 ID，Product（所属产品），Component（所属组件/关键词），Summary（总结/摘要）

ID	Product	Comp	Assignee ▲	Status ▲	Resolution	Summary	Changed
547884	4DIAC	4DIAC-ID	4diac-inbox	UNCO	---	Improvment of the dynamic type loader	2019-07-10
484130	Gendoc	Core	antonio.campesino.robles	UNCO	---	Gendoc batch launcher should allow to specify all the generation parameters instead of the sole template.	2019-07-23
447622	CDT	cdt-core	cdt-core-inbox	UNCO	---	[cdt] Unhandled event loop exception	2017-01-17
545647	Communit	CI-Jenki	ci.admin-inbox	UNCO	---	jenkins node connectivity issues	2019-07-16
482215	Communit	Cross-Pr	cross-project.inbox	UNCO	---	Unsigned bundles used	2019-10-21
401607	z_Archiv	VJET	earlyster	UNCO	---	Unable to create project from existing source	2017-04-11
510989	Ease	Modules	ease-inbox	UNCO	---	[org.eclipse.ease.modules.modeling] Lots of SWTExceptions	2017-02-02
519685	Eclemma	General	eclemma-inbox	UNCO	---	NullPointerException in CellTextConverter.getCounter	2017-07-14
506206	Oomph	Utilitie	Ed.Merks	UNCO	---	Enable redirected catalog's visibility through a command line parameter	2016-10-31
467483	Efxclips	Runtime	efxclipse-inbox	UNCO	---	[controls] FilterableTreeItem - remove reflection hack when JDK-8091687 is fixed	2015-11-25
496650	Egerrit	Core	egerrit-inbox	UNCO	---	SSLHandshakeException: unable to find valid certification path to requested target	2016-06-23
448262	EGit	Core	egit.core-inbox	UNCO	---	[egit] NPE in GitScopeUtil.findRelatedChanges	2014-10-22
446633	EGit	UI	egit.ui-inbox	UNCO	---	[egit] NPE in RepositoryAction.createExecutionEvent	2014-10-22
447967	EGit	UI	egit.ui-inbox	UNCO	---	[egit] Exception in Decorator. The 'Git Upstream Infos' decorator will be disabled.	2014-10-20

图 1.网站整体概览

规范整齐的数据对于爬虫是十分友好的，再随便进入一个 bug 报告的链接，发现链接网址包含了一个 bug 条目的详细信息，以及官方给予这个 bug 的定级（Importance），这个定级将会作为我们实验的重要参考。注意到每个 bug 链接都拥有着相同的 url 结构”https://bugs.eclipse.org/bugs/show_bug.cgi?id=???”，利用这个相同结构的特点，我们可以生成 ID 循环申请访问每一个 bug 报告对应的网站

https://bugs.eclipse.org/bugs/show_bug.cgi?id=547884

https://bugs.eclipse.org/bugs/show_bug.cgi?id=401607

图 2. 一致的网址结构

图 3 网站包含的详细 bug 信息

2.3 爬虫设计步骤（详见 spyder2.0.py):

（1）由于 Eclipse 的页面源码是静态源码, 因此利用构造 url 并且 requests.get(url), 很容易得到了一个页面的所有内容

（2）利用 BeautifulSoup 库定位到需要得到的 bug 信息（ID,Component,Importance...),简单处理一下得到文本信息

（3）因为初定的源数据数量为 10000 个 bug，因此需要对 10000 多个网址进行访问获取（因为很多 ID 对应的 bug 信息已被删除或修复导致网址内容已经为空），所以采用并发编程的思想，创建了大约 200 个处理线程，多线程并发访问多个网址，最后大约 5 分钟把 10000 多个 ID 对应的网址数据获取完毕

```
if bf.find('div',id='error_msg',class_='throw_error'):
    print("Error ID :"+target_url)
    error_urllist.append(ID)

else:
    print(req.status_code, req.url)
    Table=bf.find('td',id='bz_show_bug_column_1',class_='bz_show_bug_column')
    Component=Table.find_all('td')[6].get_text().replace(' ','').replace('\n',' ').split('  (
    Importance=Table.find_all('td')[10].get_text().replace(' ','').replace('\n',' ')
    df['Abstract'][ID]=bf.find('span',id='short_desc_nonedit_display').string
    df['Comp'][ID]=Component
    df['Importance'][ID]=Importance

print("多线程爬取")
ts=[Thread(target=get,args=(root_url,i)) for i in range(max_urlcount)]
for t in ts:
    t.start()
for t in ts:
    t.join()
```

图 4 BeautifulSoup 处理和多线程访问网址

2.4 爬取结果（详见 bugs.xlsx):

获取的近万条数据中，Comp 关键词和 Abstract 文本将会是我们后续数据处理和评分分析的重要属性，部分内容如下图所示

ID	Comp	Abstract	Importance
1	Team	Usability issue with external editors (1GE6IRL)	P3 normal (vote)
2	Team	Opening repository resources doesn't honor type (1	P5 normal (vote)
3	Team	Sync does not indicate deletion (1GIEN83)	P5 normal (vote)
4	Team	need better error message if catching up over reac	P5 normal (vote)
5	Team	ISharingManager sharing API inconsistent (1GAUL8H)	P3 normal (vote)
6	Team	API - IResource.setLocal has problems (1G5TC8L)	P5 normal (vote)
7	Team	[Team API] move/copy semantics not preserved for V	P5 normal with 1 vote (vote)
8	Team	how can we support	P3 normal (vote)
9	Team	VCM Implementation - disallow root resource to be	P3 normal (vote)
10	Team	API - VCM event notification (1G8G6RR)	P3 normal (vote)
11	Team	API: ISharingManager::load mapping vcm projects to	P3 normal (vote)
12	Team	Manage/unmanage support and policies (1GALAEG)	P3 normal (vote)

图 5.数据结果概览

3. 数据处理

3.1 目的：在后续的分析中仅对关键词进行分类/排序是很粗糙的，因此对于 bug 的 Abstract 文本内容分析比较重要，我们的想法是将文本中的名词和动词提取出来并加以统计分析，因为这两种单词（比如 File,Compile,Update..)能反映一些 bug 信息的普遍内容

3.2 过程：利用 python 的 ntlk 库能快捷地完成词性标识这一步，但如果要进行词频统计，还需要进行单词的词形还原和大小写统一处理，比如单复数的 file/files 蕴含的意思相同，而动词的时态不同 build/built 也是相近的意思，必须加以处理和统一。除此之外，还要进行字母的小写转换和统一，比如 File/file 表示的是同一个单词。其余细节不再冗余详述，详见 word_segementation.py 和 word_statistics.py

```
if word[1]=='NN' or word[1]=='NNS' or word[1]=='NNP' or word[1]=='NNPS':#名词
    new_word=lmtzr.lemmatize(word[0].lower(),'n')
    if new_word not in noun_list:
        noun_list[new_word]=1
    else:
        noun_list[new_word]+=1

elif word[1]=='VB' or word[1]=='VBD' or word[1]=='VBG' or word[1]=='VBN' or word[1]=='VBN' or w
    new_word=lmtzr.lemmatize(word[0].lower(),'v')
    if new_word not in verb_list:
        verb_list[new_word]=1
    else:
        verb_list[new_word]+=1
```

图 6.利用词性标签进行统计计数

3.3 处理结果：根据 P1~P5 得到了每一级别的高频词统计结果（详见 Px_noun.xlsx 和 Px_verb.xlsx)，统计之后发现各级别出现的高频词其实是有一定重合的，因此再进行了二次处理，统计了每个高频词在各级别出现的频率和权重，用作后续评分依据

0	1	P1	P2	P3	P4	P5
view	571	0.099825	0.122592	0.607706	0.10683	0.063047
editor	510	0.086275	0.105882	0.647059	0.113725	0.047059
file	489	0.100204	0.120654	0.640082	0.071575	0.067485
project	450	0.097778	0.115556	0.633333	0.071111	0.082222
error	446	0.183857	0.105381	0.569507	0.08296	0.058296
class	378	0.108466	0.092593	0.687831	0.092593	0.018519
dialog	358	0.083799	0.106145	0.662011	0.092179	0.055866
java	332	0.138554	0.14759	0.575301	0.126506	0.012048
package	294	0.088435	0.139456	0.591837	0.159864	0.020408
code	248	0.120968	0.133065	0.596774	0.092742	0.056452
type	248	0.084677	0.137097	0.568548	0.185484	0.024194
npe	247	0.408907	0.093117	0.481781	0	0.016194
page	230	0.121739	0.108696	0.66087	0.065217	0.043478
method	226	0.141593	0.097345	0.60177	0.128319	0.030973
source	208	0.129808	0.192308	0.504808	0.120192	0.052885
search	206	0.058252	0.179612	0.650485	0.106796	0.004854
problem	201	0.129353	0.099502	0.626866	0.094527	0.049751
menu	201	0.094527	0.099502	0.661692	0.064677	0.079602
name	195	0.097436	0.107692	0.630769	0.087179	0.076923

图 7.部分高频词出现总数&&各级别出现频率

4. 数据分析与需求评分

4.1 抽取可用与评分的关键字

- 抽取 Comp 属性中可用于评分的关键词

经过筛选，得出如下 13 个关键词，并对这 13 个关键词进行评分。

Compare	3
UI	2
Build	4
Resources	3
UserAssistance	2
Debug	4
Ant	3
Core	3
Scripting	2
SWT	1
Team	1
Text	3
Update	3

Comp 属性中筛选出来的关键词及其对应评分

- 抽取 Abstract 属性中可用于评分的关键词

经过筛选，得出 47 个关键词，并对这 47 个关键词进行评分。

abs	grade
block	4
break	3
breakpoint	3
bug	3
build	4
cannot	3
collapse	5
compilation	4
compile	4
...	...

Abstract 属性中筛选出来的关键词及其对应评分

- 根据抽取出来的关键词对原 bug 条目(bugs.csv)进行评分

编写代码，对 Bug 条目进行评分。

若 Comp/Abstract 属性中出现所抽取的关键词，则对该条目加上所出现关键词的分数

- 将评分后的条目进行分级

若分数 grade≥8 则该 Bug 为 P1 等级

若分数 8 > grade≥6 则该 Bug 为 P2 等级

若分数 6 > grade≥2 则该 Bug 为 P3 等级

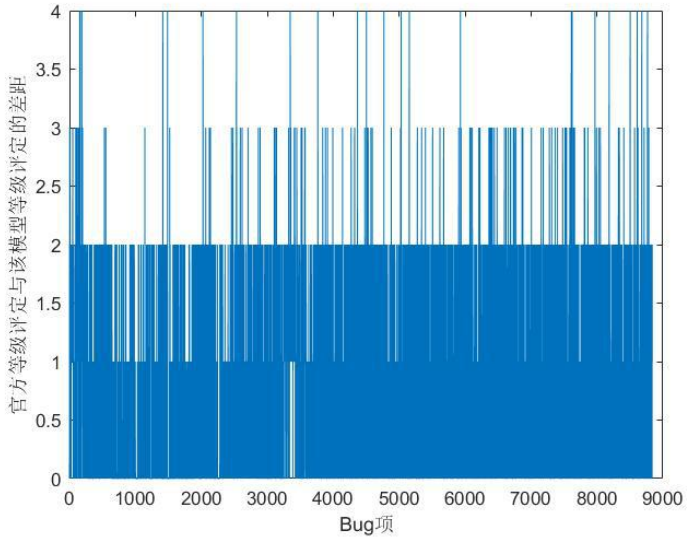
若分数 2 > grade≥1 则该 Bug 为 P4 等级

若分数 1 > grade≥0 则该 Bug 为 P5 等级

5. 排序结果检验

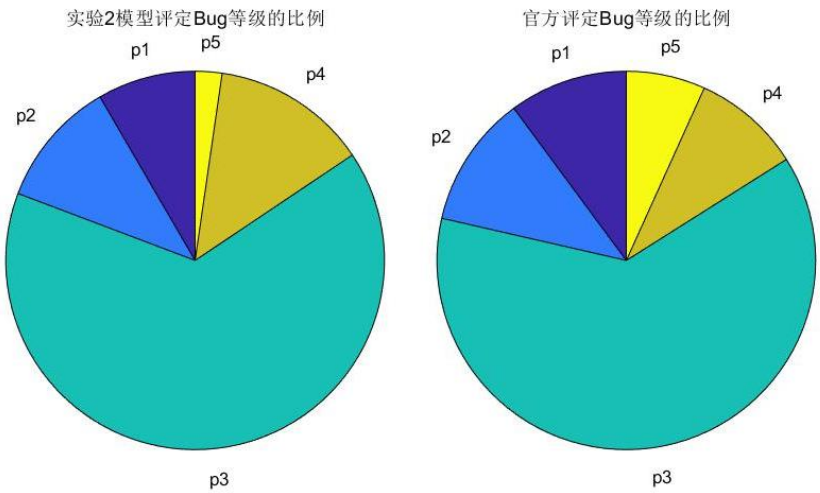
通过将该模型得出的等级结果与 Bug 原等级结果进行对比，得出如下结论：

- 8139 条数据中有 4158 条数据的等级评定结果与原等级评定完全一致，一致率达到 47.03%
- 下图为官方等级评定与本次实验模型评定结果的差距，可以看出，等级评分差距大概在 1~2 之间，模型相对可靠。



- 下图为官方等级评定以及本次实验模型等级评定后的等级分布比例

可以看出本次模型评定结果与官方评定结果在 P1~P3 部分比例大致相同，在 P4、P5 部分比例出现一定程度上的偏差。总体模型评定效果良好。



6. 误差分析

我们选择的 bug 分级方法是通过统计关键词在 bug 描述和类别中出现的次数和比例，通过人工方式筛选出部分有代表性的关键词作为评分标准，这种评分方法不一定能够客观反映 bug 的重要程度。由于关键词的筛选是在高频词中人工筛选，这样选出的关键词虽然有一定的代表性，但也有可能有遗漏，导致有些关键词未被正确选入或是选入了没有代表性的关键词。

在一开始的数据统计过程中，我们发现 P3 等级的 bug 占了 bug 总数的一半以上，而其他等级的 bug 数量少，因此我们认定大多数重要性不太高的关键词的等级应该被评为 3，重要的关键词等级为 5，而不重要的关键词等级为 1，这样的评级方式受到主观判断的影响很大，导致关键词的重要性有可能受到误判。

在统计每一条 bug 的具体评分时，我们采取了每当出现一个关键词时，直接将其评分累加的方式。这样的方式有可能导致当一些关键词之间的出现有一定程度的逻辑关联时，使得该条 bug 的统计得分虚高。

本实验所采用的的关键词评分方式的默认条件是：每一个关键词有着不变的评分，每一条 bug 中出现的关键词越多，该条 bug 评分越高。但是实际情况并非如此，同样一个关键词在不同的 bug 中可能体现出不同的评级，统计数据只能一定程度反映这个关键词的评级倾向，并不能决定具体某一条 bug 的评级。而多个关键词同时出现的 bug 也不一定评分高，有些关键词处于相近的等级，那么他们组合后的等级可能也是相近的，简单相加的方式无法反映这样的可能。

总而言之，我们采取的实验方法只是一种粗糙的、大致模拟的分级方式，默认的条件与现实情况有着较大差异，因此得出的实验结果与原本结果有较大差异是正常情况。

注：小组成员及成绩比例

学号	姓名	成绩比例
171860657	徐浩	0.25
171860677	吴鸿祜	0.25
171868570	周吴成	0.25
171870642	王子豪	0.25