

软件需求工程Lab2实验报告

2020.12.17

小组成员

学号	姓名	成绩占比
171870559	季镇澜	20%
181860013	储备	20%
181860077	余帅杰	20%
181860094	王佳奕	20%
181860151	周清远	20%

实验目的

为软件需求进行需求优先级排序。

实验尝试一

综述

首先，我们按照实验手册提供的可行思路进行实验，即：爬取Eclipse官方网站的缺陷报告内容，把缺陷报告内容当作一种“软件需求”，根据官方给出的“bug修复的优先级(Priority)”以及“严重性(Severity)”，结合每条缺陷报告的所属产品(Product)、所属组件(Component)、摘要(Summary)，进行自然语言处理后用机器学习的方法训练了一个分类器，遗憾的是，训练的结果并不令人满意。

实验数据

确定的开源项目：

Eclipse

信息源与需求获取：

我们把Eclipse官方网站提供的缺陷报告内容视为需求，利用爬虫，从https://bugs.eclipse.org/bugs/show_bug.cgi?id={}（其中{}填入对应条目的id，即为该条目的内容）上从最新发布的条目开始爬取了约10000条缺陷报告，按官方给出的P1-P5的“bug修复的优先级(Priority)”进行分类。由于本次爬取的数据中P3级别的条目占了所有条目的99%以上，训练出来的分类器的准确率虽然高达99.7%，但是并没有很

强的说服力。

因此我们又把思路转向进行严重性(Severity)分类，关于严重性和优先级，可以在官方文档([https://wiki.eclipse.org/Bug_Reporting_FAQ#What is the difference between Severity and Priority.3F](https://wiki.eclipse.org/Bug_Reporting_FAQ#What_is_the_difference_between_Severity_and_Priority.3F))中看到如下描述：

What is the difference between Severity and Priority?

Severity is assigned by a user and describes the level of impact the bug is having on them.

The levels generally have the following meanings:

- blocker - the bug blocks development or testing of the build (for which there is no work-around)
- critical - implies "loss of data" or frequent crashes or a severe memory leak
- major - implies "major loss of function"
- normal - default value, regular issue, some loss of functionality under specific circumstances, typically the correct setting unless one of the other levels fit
- minor - something is wrong, but doesn't affect function significantly or other problem where easy workaround is present
- trivial - cosmetic problem like misspelled words or misaligned text, but doesn't affect function (such as spelling errors in doc, etc.)
- enhancement - request for enhancement (also for "major" features that would be really nice to have)

But overall, it is up to each project to decide how they triage/handle bugs so some variability from project to project will occur.

Priority is assigned by the developer/committer and describes the importance a developer places on fixing the bug:

- P1 - "stop ship" defect i.e. we won't ship if not fixed
- P2 - intent is to fix before shipping but we will not delay the milestone or release
- P3 - nice to have
- P4 - low priority
- P5 - lowest priority

严重性是由用户定义的，反映bug对用户工作项目的严重程度。我们对爬取的数据做了一些处理和统计，得出以下结果：

('P1 ', 'blocker')	11
('P1 ', 'critical')	3
('P1 ', 'normal')	3
('P2 ', 'blocker')	2
('P2 ', 'enhancement')	4
('P2 ', 'major')	2
('P2 ', 'normal')	7
('P3 ', 'blocker')	216
('P3 ', 'critical')	187
('P3 ', 'enhancement')	846
('P3 ', 'major')	453
('P3 ', 'minor')	180
('P3 ', 'normal')	7451
('P3 ', 'trivial')	52
('P4 ', 'enhancement')	1
('P4 ', 'major')	1
('P4 ', 'normal')	1
('P5 ', 'enhancement')	2
('P5 ', 'normal')	1

可以看到高优先级的bug用户定义的严重性也往往越高，P1级别有blocker、critical，而P5优先级只有enhancement，P2-P4级别的则有更较为中间的严重性。如果我们把bug视为需求，那么bug的严重性和需求的优先级就会正相关的关系，因此我们决定以严重性为分类标准再次尝试训练分类器。

在观察到normal占据数据的大部分后，我们吸取了初次尝试的教训，对数据进行了**强制平均**处理，从严重性的角度，爬取'blocker'及'critical'不超过3000条、'major'不超过3000条、'normal'不超过3000条、'minor'不超过3000条、'trivial'及'enhancement'不超过3000条数据（把两个最高严重性、两个最低严重性的类别合并，仍当作5类进行分类），最终，由于官方缺陷报告内容优先，这样的强制平均使得我们只爬取了约7500条数据，其中'normal'类别占据了40%。

模型多轮训练后，准确率最高也没法突破40%，并不能让人满意。

注：自然语言处理和机器学习分类的实现将在实验方法中进行详述。

实验方法

模型设计

实验的主体模型是BERT，一个基于Transformer的双向编码器。

我们希望通过BERT对需求描述进行提取特征，融合在句子向量中，并依据这个向量对需求进行分类

我们使用bert-base-uncased初始化BERT模型权重，利用BERT对自然语言的需求描述进行编码，得到了需求的特征向量（句子向量），然后利用全连接层映射到对应的类别向量，把隐藏状态的向量值作为概率分布，使用交叉熵损失函数计算Loss，然后利用SGD作为优化器。

上述模型在自然语言处理中作为句子分类领域的一个较强的Baseline，有比较简单的结构和较强的性能。

数据处理

在对数据进行类别平衡之后，把需求文本和对应的需求等级用json格式存储。

用bert-base-uncased tokenizer对需求描述的进行BPE分词和编码，不足部分用0进行padding至128的长度（观察到需求的Summary长度较短），超出的部分截断至128的长度。同时计算Attention mask，对padding的部分进行mask避免padding token的影响。

对label使用map函数，映射到不同的数值，使用0代表UNK（未知类别），便于后续的训练使用。

训练设计

使用的超参数如下

```
Epoch: 10
Learning Rate: 0.001
BatchSize: 32
Random Seed: 44
Train Sample:Test Sample = 9:1
GPU: GTX1060
```

实验结果

如前文所述，模型多轮训练后，准确率最高也没法突破40%，并不能让人满意。可能原因是：所爬取缺陷报告的内容中语义不能很好地体现出分类等级，模型在训练中并没有捕获到比较明显的语义和分类等级的联系，因此效果不佳。

我们结合数据具体的分析了一下原因：

1. 需求的描述本身就不具备有完整的语义信息
 - 1.1. 需求描述有较多的专业名词和缩写，这些词大概率不在常规词表里，BERT tokenizer不能很好的胜任分词和编码的任务
 - 1.2. 需求的描述都很简短，甚至说不上是语法结构完整的自然语言。
2. BERT的Pretrain是基于大规模的英文日常语料，因此在句子结构和内容领域上都有比较大的GAP。在其他领域轻松SOTA的BERT并不能够很好的提取到所需的特征。

综上所述，BERT这种相对处于自然语言处理的技术前沿的模型并不能够很好的适应于这个任务

反观一些以SVM、TF-IDF为代表的相对传统的机器学习技术有可能取得相对好的技术，因为需求相对集中的围绕一些专有名词，反而能够比较好的表征需求，进而更好的分类

最终小组成员们认定，实验尝试一是失败的，于是我们开始了新的尝试。

实验尝试二

综述

在实验一中，我们爬取了GitHub上IDEA、VSCode两款著名IDE的Issue或Pull Request，从中获取了一些需求。其中，VSCode的Issue中，带有'feature-request'标签的Issue能够非常直观地反映用户对这款IDE的需求，而且其数据量也足够大：

❗ 2,154 Open ✓ 11,907 Closed

因此我们决定对这些Issue进行分类。

实验数据

确定的开源项目：

VSCode

信息源与需求获取：

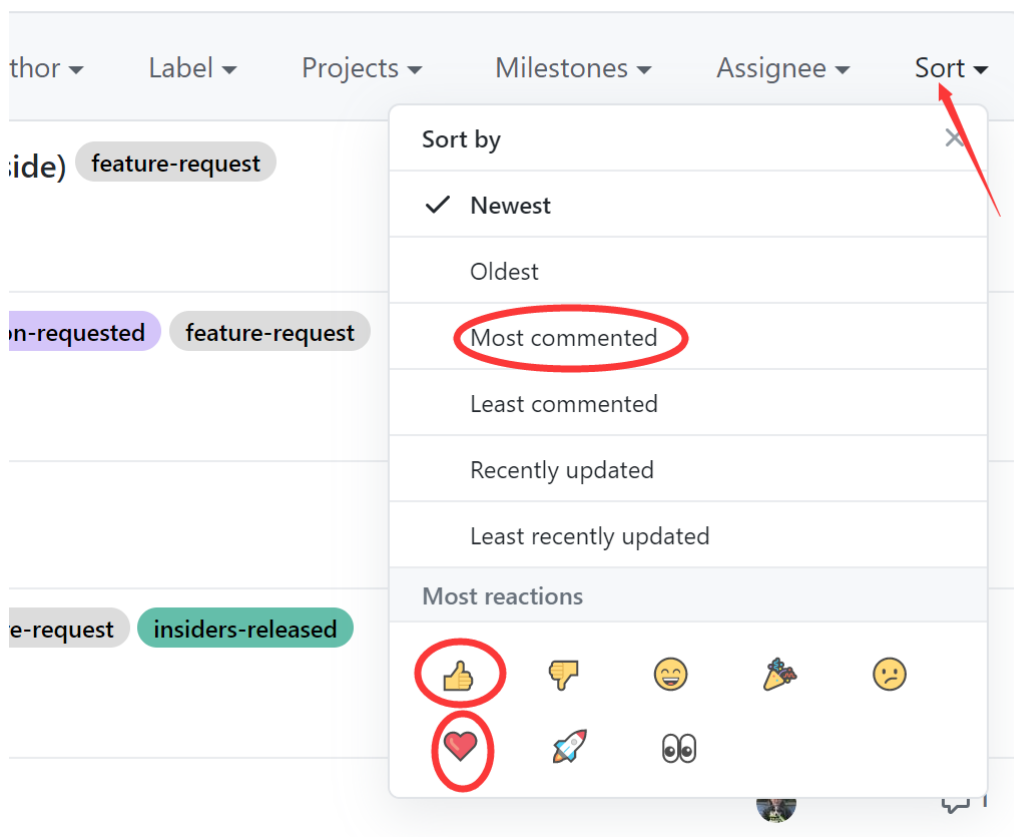
我们爬取了GitHub上VSCode仓库中带有'feature-request'标签的Issue及其相关内容约10000条。信息来源网址：[https://api.github.com/repos/microsoft/vscode/issues?state=closed&sort=created&direction=asc&per_page=100&labels=feature-request&page={} \(其中{}填入页数\)](https://api.github.com/repos/microsoft/vscode/issues?state=closed&sort=created&direction=asc&per_page=100&labels=feature-request&page={}) 和 [https://github.com/microsoft/vscode/issues/{} \(其中{}填入issue的ID\)](https://github.com/microsoft/vscode/issues/{})。

实验方法

这些Issue的提出者、回答者，有很多VSCode的用户和开发者，但是这些Issue却没有官方提供的一个优先级分类或者排序，因此我们需要人工制定一套分类/排序准则。

而对上万条的需求一个个地通过组员协商划分优先级别显然是不可行的，于是我们尝试找到一套能够对所有Issue都通用的进行优先级分类的方法。

在观察了大量Issue后，一个不起眼的小东西引起了我们的注意：



于是，在本次实验中，我们提出了一个新颖的方法，利用每条issue下的评论数和emoji评论内容衡量每条带有feature-request标签的issue的重要性。在本小节中，我们将给出具体细节。

为什么使用评论数

当程序用户在GitHub上提交带有feature-request标签的issue时，这一issue的内容将会完全公开，不仅仅是开发者，其他的程序用户也能够对该issue发表评论。一个常见的现象是：当某个issue提出的需求符合大量用户的需求时，这些用户也会在该issue下留言，或提出改进的意见，即受用户群体重视的需求下往往有较多的留言；相反的，如果一个issue所提及的内容不重要或者并不受欢迎，该issue下的评论也会非常稀少。事实上，从事issue排序的研究者们已经注意到了这一点，issue下的评论数已成为衡量一个issue是否该被重视的重要指标在多篇论文[1-2]中被使用.除此之外，利用评论内容衡量被评论软件、产品的质量的研究方法也在最近的软件工程研究领域多有应用[4-5]。

为什么使用emoji

emoji作为一种广受欢迎的符号，近年来在软件工程领域同样获得了一定的关注[3]。这一语言形式已广泛应用于自然语言处理（例如情感分析）中，但在软件工程这一学科，尤其是软件需求分析中，因为具体场景的缺失，相关的研究工作依然很少。

在GitHub中，emoji可以被用户用于直接对issue内容发表观点和感受，emoji以其远超一般语言的简单易用性而广受欢迎。相较于自然语言，emoji有以下几点优势：

1. emoji因为其便于理解的特点，成为了一种全球通用的语言，不同国籍、文化背景的开发者 and 用户都可以便捷地使用它。因此，有emoji评论的issue数量巨大，满足实验的数据要求
2. emoji本身含义明确，例如，当用户对一条issue的内容感到满意时，往往会使用笑脸或者竖起的大拇指；相反的，如果用户对issue不满意时，则会使用倒着的大拇指。由于emoji相较于自然语言而言几乎不存在多意和歧义，在利用emoji对issue内容进行评估时，能够享受极大的便利。

3. Lu, Xuan et al. [3]研究证明, GitHub的用户往往在有较强且较直接的情绪时更倾向于使用emoji进行评论, 由此我们可以假设, issue上的emoji能够真实地反映用户对该issue的看法。

组合利用评论数和emoji

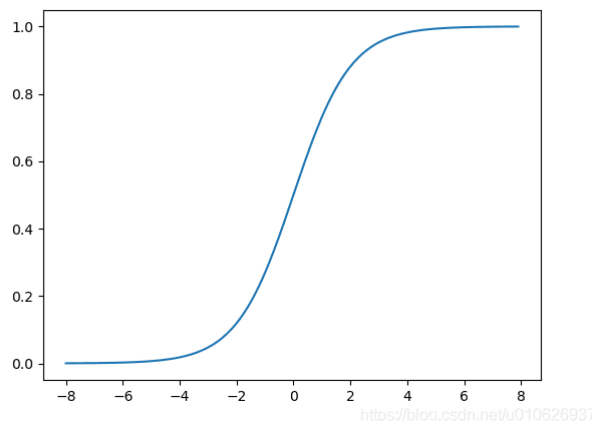
综上所述, 我们决定利用评论数和emoji这两个指标对所有带有feature-request标签的issue进行全排序。相较于一般的将所有需求简单的分为5级 (Highest, High, Medium, Low, Lowest) 或7级 (P1,P2,P3,P4,P5,P6,P7), 全排序显然拥有着更高的准确度和实用性。然而, 在本次实验中, 如何将评论数和emoji这两个指标组合在一起, 形成一个混合排序指标成为了一个棘手的问题。简而言之, 在缺乏大量的研究分析情况下, 我们并不能得出上述两个指标在混合指标中比重的一个准确数值。受Yang et al.[6]工作的启发, 我们假设这两者在排序过程中同样重要, 并简单地将这两个指标以1: 1的比重组合在一起, 实验结果表明, 这一朴素的方法在实际场景中有着优秀的效果。

另外, 我们还应该关注emoji中'thumbs down'👎和'confused'😕对issue的负效益。如前文所述, 用户对其他用户提出的自己不赞成、不满意的请求往往会使用这两个emoji。最初, 我们在进行统计时, 将这两个emoji按照-1的权值参与计算, 即:

$$\begin{aligned} Val &= Val(comment) + Val(emoji), \\ Val(comment) &= Num(comment), \\ Val(emoji) &= Num(emoji) - 2 * (Num(thumbs_down) + Num(confused)) \end{aligned}$$

但我们做出一些统计后发现绝大部分的issue中emoji数比评论数要多得多, 也就是说, emoji会对排序值有极大的影响。另外, 个别issue的评论数和emoji数较与其他issue多出极多, 对统计图的整体展示不便。我们会在实验结果中给出统计图和说明。

我们观察数据特征后发现绝大多数的样本数据都集中在一个区间, 但是偶尔也会有偏离区间很远的值, 例如emoji的值大部分都在0-700之间, 但是最大值达到了3000, 所以我们需要一种映射, 加大数据集中区域的离散程度, 一开始我们想直接使用sigmoid函数:



此函数在0附近的导数是最大的, 而在正负无穷区域导数趋于0, 这样就可以加大集中数据的离散程度, 弱化极端数据的作用, 但是, 该函数在大于5时的函数值就近乎于1了, 所以我们需要先把数据映射到一个比较小的区间在使用sigmoid函数, 所以我们将数据除以最大值再乘以5, 最后再使用sigmoid函数进行映射。

最后, 考虑到两个指标的真正1: 1比重组合, 我们对实验获取的原始统计数据进行了标准化处理, 标准化处理的方法如下:

$$\begin{aligned}
S_Val &= 0.5 * S_Val(comment) + 0.5 * S_Val(emoji), \\
S_Val(comment) &= \text{sigmod}\left(\frac{5 * Val(comment)}{Max(comment)}\right), \\
S_Val(emoji) &= \text{sigmod}\left(\frac{5 * Val(emoji)}{Max(emoji)}\right), \\
\text{sigmod}(x) &= \frac{1}{1 + e^{-x}}, \\
Val(comment) &= Num(comment), \\
Max(comment) &= \text{Maximum number of } Val(comment), \\
Val(emoji) &= Num(emoji) - 2 * (Num(thumbs_down) + Num(confused)), \\
Max(emoji) &= \text{Maximum number of } Val(emoji)
\end{aligned}$$

标准化处理把emoji数和评论数用于1: 1结合的值映射到了[0, 1]的区间，其中0.5代表原始数据的0值。

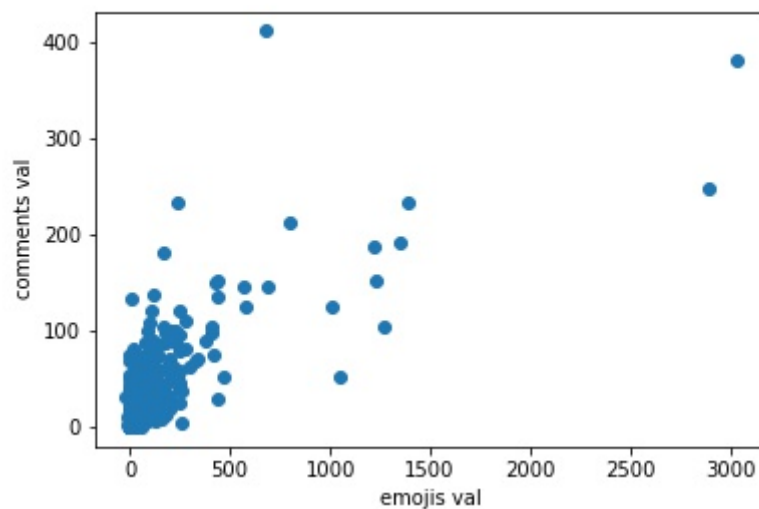
实验结果

我们将实验得到的排序结果储存到了一份csv文件中，该文件可以在我们本次实验的仓库中查看 (https://github.com/NJU-Software-Requirement/Software_Requirement_EXP2/blob/main/data/val_sort_ed.csv)。

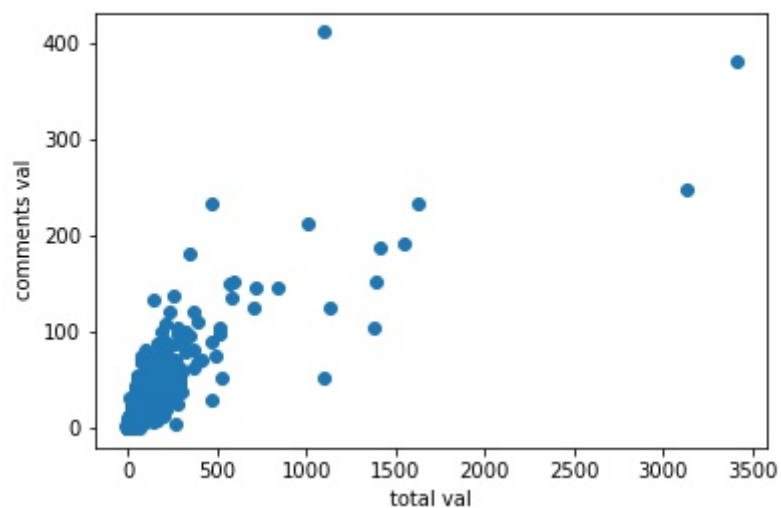
对于我们的排序指标（评论数、emoji数）和排序值（Val），我们绘制了一些统计图来表示它们之间的相关程度，这些图片文件可以在我们本次实验的仓库中查看 (https://github.com/NJU-Software-Requirement/Software_Requirement_EXP2/tree/main/img)。

初始排序：原始统计量直接相加

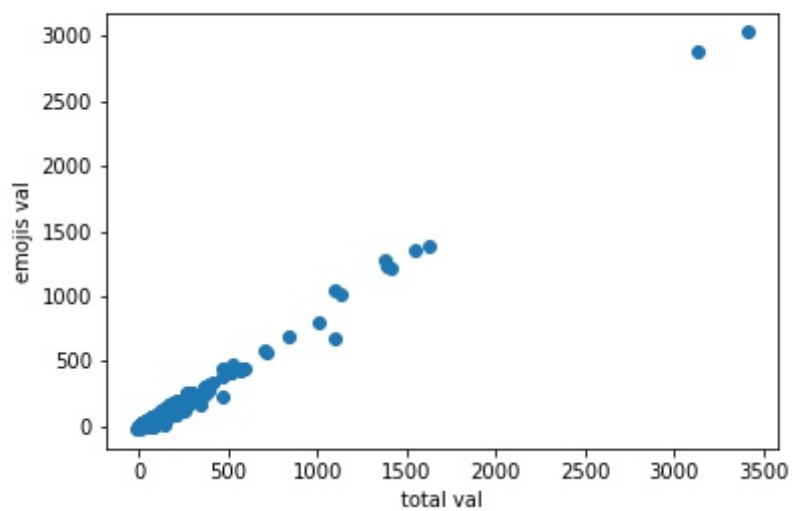
1. 评论数和emoji数：



2. 评论数和排序值



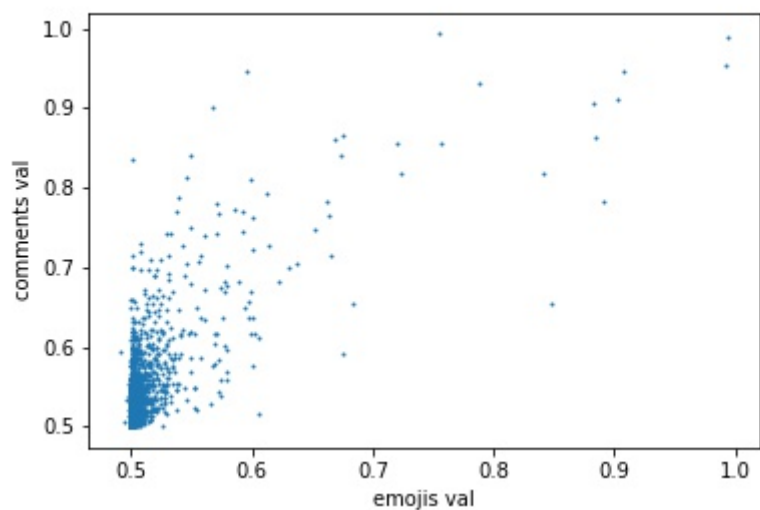
3. emoji数和排序值



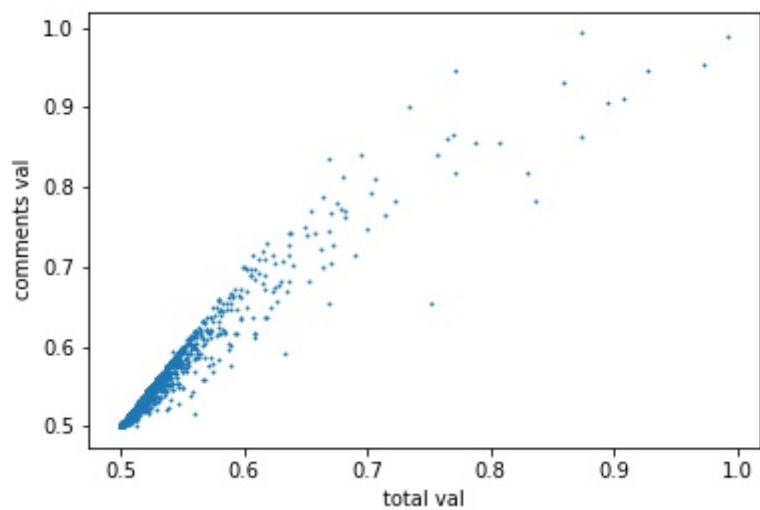
从以上三张图可以看出，emoji数几乎主导了排序值，而评论数的影响非常之小，这并不算是二者的**真正1:1结合**。另外，个别偏离的点影响了统计图整体的坐标缩放，使得其他集中的点并没有得到很好的展示

最终排序：标准化处理

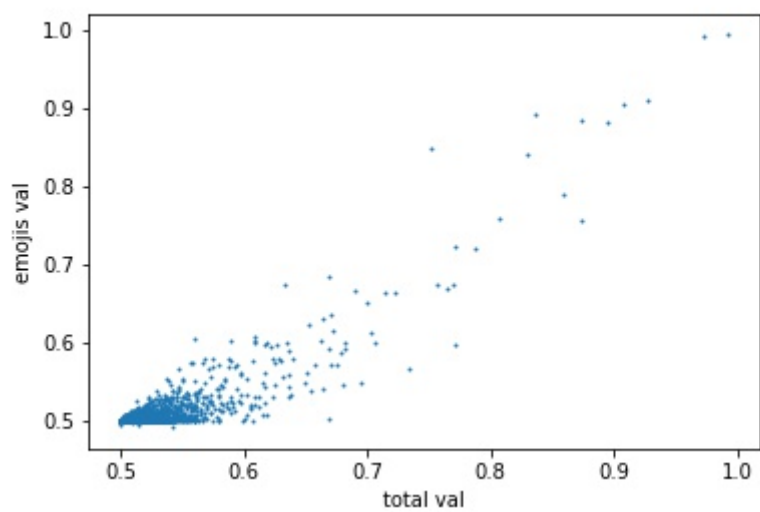
1. 评论值和emoji值



2. 评论值和排序值



3. emoji值和排序值



以上三张图反映了评论值和emoji值标准化1：1结合后对排序值的影响。

样例研究

在本小节中，我们选取了几个issue作为实例进行分析：

#9543号issue (<https://github.com/microsoft/vscode/issues/9543>) :

Hint for task name on hover #9543

 Closed

normalser opened this issue on 21 Jul 2016 · 0 comments

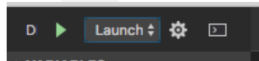


normalser commented on 21 Jul 2016

- VSCode Version: 1.4.0-insiders
- OS Version: OSX

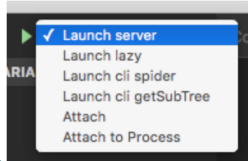
Steps to Reproduce:

1. Hover `Launch`



2. No hint/tooltip showing

3. Would be nice if it showed full task name - right now one need to click dropdown to see the task name and there are multiple tasks starting with `Launch` ...



- 4.

 1

这条issue对需求有比较清晰的描述，但是只收获1个点赞，0条评论。分析其需求可以发现，该用户认为鼠标点击下拉菜单显示任务名过于繁琐，希望能够在鼠标悬浮时就能将任务名完全显示。这显然并不应该是一个高优先级的需求，这条issue仅仅收获1个点赞，说明并没有太多用户认可这样的需求，大多数用户可能已经习惯于点击下拉菜单的操作，这样的操作也符合大多数编程人员的习惯。从这条issue收获的互动数（emoji数、评论数）可以看出，个别用户认为“更好”的需求可能并不能得到多数用户的集体认同，不能盲目地实现每个用户提出的需求，个人的“便利”可能会造成更多人的“不便”。相反，如果这条issue收获了成百上千的点赞，在所有的issue中排序靠前，那极有可能说明这种“悬浮显示”的需求更符合多数用户的操作习惯，将其优先级提高也将会是合理的。

#61752号issue (<https://github.com/microsoft/vscode/issues/61752>) :

Add a feature like quokkajs in vs code #61752

 Closed

Aamir2000 opened this issue on 25 Oct 2018 · 0 comments



Aamir2000 commented on 25 Oct 2018

Please add all the features of quokkajs in vs code for free please don't integrate it or collab just add your own and do it for free not like quokkajs properitery Shit.....and a lot of people going to get attracted by this feature and this going to knock all the editors out of the competition and vscode become no.1 do it fast....fast fast fast fast

 1



weinand added the `feature-request` label on 25 Oct 2018



weinand closed this on 25 Oct 2018



vscodebot bot locked and limited conversation to collaborators on 10 Dec 2018

这是一条让开发者非常无语的无赖需求，他要求VSCode团队添加Quakka js工具的所有功能，并让用户免费使用，而不是像Quakka js一样收费，这样一来VSCode就能成为宇宙第一IDE/editor了。0评论、1踩、当天close掉，足以说明这条feature-request是一则笑话。这样的需求会让开发者有非常之大的工作量，而且VSCode真的需要靠免费提供quakka js的功能来击败其他IDE收获更多用户吗？这条需求的优先级无疑是极低极低的。

#396号issue (<https://github.com/microsoft/vscode/issues/396>) :

Add support for opening multiple project folders in same window #396

Closed


stoffeastrom opened this issue on 21 Nov 2015 · 380 comments

Labels


feature-requestworkbench-multiroot


Milestone


📌 October 2017


stoffeastrom commented on 21 Nov 2015


Right now it doesn't seem possible to opening multiple project folders in the same window which imho is a bit constraining. If you are working on modular modern projects it's a must have to be productive.


 2628

 36

 69


 149

 19

 237


这条需求有着380条之多评论数和极高的emoji评论数，其中，表示真面意义的emoji如竖起的大拇指、笑脸和爱心占了极大的比重，而负面emoji如倒着的大拇指只有36个。这条需求描述了VSCode并不能在同一个窗口打开多个项目的文件夹，而程序员在现代编程项目中工作时，在同一个窗口中打开多个项目的文件夹已经成为一个必要的生产力。


评论区有大多数的支持者，但也有一些反对的声音：


dmccaffery commented on 22 Nov 2015


I'm not sure I understand the ask; its a lightweight code editor, not an IDE ... why would you need to open multiple "project" folders that aren't hierarchical (where you could set the working path to a mutual parent)?


If you're working on modules that are disparately stored on disk that are somehow interacting with each other to that degree, then they are too tightly coupled to begin with... are your projects siblings? If so, just open the parent folder, or parent / parent folder... wherever the root of your "solution" is.

 27

 539

 9

 3

 18



不过这个“反对的声音”似乎并没有得到太多人的支持（539踩）。

有其他的用户声援issue的提出者，并且得到了极高的赞同率：



yoorek commented on 22 Nov 2015
...

Sublime, Atom, Webstorm - they are also "lightweight" code editors (except maybe Webstorm) and they allow opening multiple root folders from different locations. These editors are probably 99% of what web developers use.

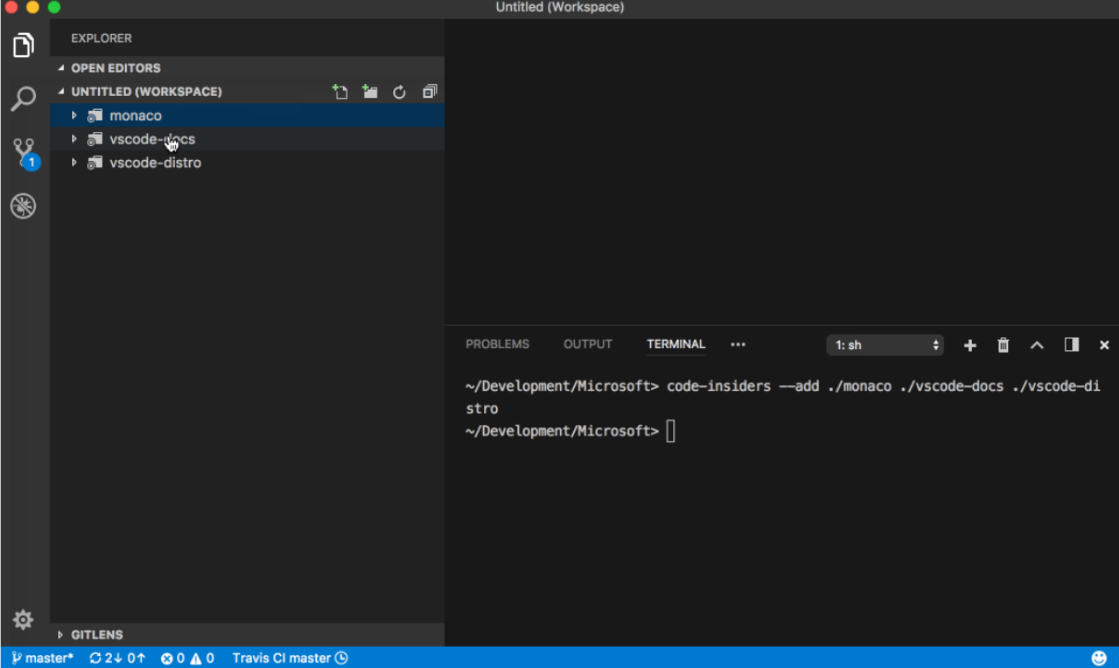
Code can compete with them with much better Typescript support (and it will be very popular consider Angular 2 is coming) but only if it will give developers what they use now as a basic functionality.

 374
 1
 9
 12
 32

最终，该issue也受到vscode开发者的重视，并且在17年2月份实现了这条需求：







bpasero commented on 2 Nov 2017
Member
...

Support for multi-root workspaces is now enabled by default in the stable release.



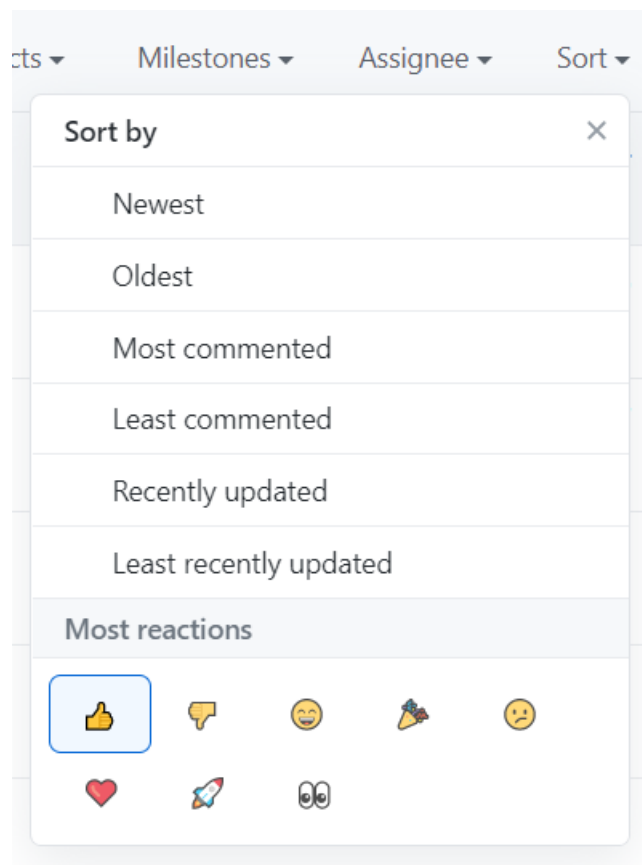
Please refer to our [documentation](#) for a full explanation of all the features that come with it. Extension authors should refer to our [wiki](#) that explains the new extension APIs to make your extension ready for multi-root workspaces. We are happy that quite some extensions have already started to adopt the new multi-root API, thanks for that!

Please do not hesitate to file issues for multi-root as you encounter them, we are still planning on making tweaks and providing additional APIs for extensions to work with multi-root workspaces in the future.

 29
 5
 38
 1
 13

在我们的排序标准中，这条issue因为有着极高的互动数所以拥有极高的优先级，开发者耐心的回复和最终的实现成果很好地证明了这条需求的高优先级。

事实上，在Github网页中，利用评论数和emoji使用情况进行排序的思想已经有所体现，详情见下图：



也正是这个排序选项给了我们很大的启发。

结论

在软件开发维护生命周期中，常常有多个需求同时提交给软件开发维护团队。由于资源受限，软件团队往往无法同时满足所有需求的实现。因此，通过对软件需求进行优先级排序，帮助软件开发维护团队合理分配有限的资源。而在我们小组本次实验的第二次成功的尝试中，GitHub上VSCode的issue给VSCode开发维护团队提供了海量需求，对于如此之多的需求，开发维护团队不仅仅需要判断哪些需求是合理的、需要实现，哪些需求是无理的、没有必要实现的，还需要对待实现的需求进行优先级排序，即哪些需求亟需实现以方便用户使用、哪些需求目前阶段并不需要实现，可以当作增强、待扩展的功能等到开发资源充足时再实现。我们进行的互动量排序就是一种非常有效需求优先级全排序方法。而在GitHub上已经有了相关指标的issue排序方式，说明这种排序方式已经很有可能被开发人员使用，作为评定需求优先级的一大因素。

参考文献

1. Di Sorbo, Andrea, et al. "" Won't We Fix this Issue?" Qualitative Characterization and Automated Identification of Wontfix Issues on GitHub." arXiv preprint arXiv:1904.02414 (2019).
2. Dhasade, Akash Balasaheb, Akhila Sri Manasa Venigalla, and Sridhar Chimalakonda. "Towards Prioritizing GitHub Issues." Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference. 2020.
3. Lu, Xuan, et al. "A first look at emoji usage on GitHub: an empirical study." arXiv preprint arXiv:1812.04863 (2018).
4. McIlroy, Stuart, et al. "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews." Empirical Software Engineering 21.3 (2016): 1067-1106.

5. Lin, Dayi, et al. "An empirical study of game reviews on the Steam platform." *Empirical Software Engineering* 24.1 (2019): 170-207.
6. Yang, Xinli, et al. "Combining word embedding with information retrieval to recommend similar bug reports." 2016 IEEE 27Th international symposium on software reliability engineering (ISSRE). IEEE, 2016.