# 特征提取及相似度计算模型介绍：

## 模型介绍

### 1.1 模型概述

此模型借鉴了衡量字符串相似度的dice距离与非参数统计中的Wilcoxon秩和检验法，目的是为了提取case/patch的核心特征，并量化各个特征的相似度，以达到推荐case与求解case关联分析的目的。

### 1.2 模型步骤

模型主要分为两个部分：

1. 特征提取
2. 相似度计算

第一步主要通过经验制作特征辞典，根据特征辞典将case/patch处理为有序特征序列。

第二步则是根据我们case的显示情况改进两种方法，使之达到能处理相似度的目的。

### 1.3 思路说明

（1）特征提取

可以看到，我们的case是存在固定的语法的，而且patch后期只需要规范化comments亦可以达成一个较为规范的语法形式，于是则有可能根据先验的、预先设置好的一些规范提取其特征。

以case为例，首先我们需要根据经验将一些出现频率较高的、有代表性的语句提取出来。

举例来说，先看如下case：

**"添加项目，缺省配额":["点击身份管理->项目2->创建项目",**

**"输入名称为pro1%时间戳",**

**"点击配额->创建项目",**

**"循环10秒,页面上存在pro1%时间戳",**

**"清理项目为pro1%时间戳"]**

**"项目名称为空":["点击身份管理->项目2->创建项目",**

**"点击创建项目",**

**"检查页面上存在取消"]**

**"创建1个云硬盘": ["点击项目->云硬盘->云硬盘2->创建云硬盘",**

**"输入云硬盘名称为volume%时间戳",**

**"点击创建云硬盘",**

**"检查页面上存在volume%时间戳",**

**"循环10秒,类型为云硬盘,名称为volume%时间戳的对象状态为可用",**

**"清理云硬盘为volume%时间戳"],**

**"使用cirros镜像创建云硬盘": ["点击项目->云硬盘->云硬盘2->创建云硬盘",**

**"输入云硬盘名称为volume\_from\_image%时间戳",**

**"点击没有源，空白云硬盘。->镜像",**

**"点击选择一个镜像->cirros-0.3.5-1 (12.5 MB)",**

**"点击创建云硬盘",**

**"检查页面上存在volume\_from\_image%时间戳",**

**"清理云硬盘为volume\_from\_image%时间戳"]**

在case中，可以看出前两个case主要与项目相关，来自keystone；而后两个与云硬盘相关，来自cinder。

根据经验，我标出了三种不同颜色的高亮部分，其中黄色代表与项目相关的特征语句，换言之，只要出现黄色部分，则可以断定这个case调用到了与项目相关的part；而绿色则是与云硬盘相关的语句，只要出现“云硬盘2”、“创建云硬盘”、“volume”等字样，那么我们可以认为这个case的确用到了云硬盘相关的功能；特别地，我标注了一些红色的部分，这些部分在我看来是不能踩中的“雷”，比如说“镜像”、“项目”这些字样，看起来他们说到了这些功能，但事实上只是在用其他feature时“路过”了它们，而并未真正用到，这些则是我们后期丰富特征辞典时要特别注意的。

直观上我们可以想到，特征语句的出现的频率与用到此feature的紧密程度成正相关的关系，而不同case之间的同色高亮频率可以作为一种衡量它们相似程度的度量。

于是我们不妨假设特征辞典为：

{

“项目”:[“**身份管理->项目2**”, “**创建项目**”, “**pro1**”, “**清理项目**”]

“云硬盘”:[“**云硬盘2**”, “**创建云硬盘**”, “**volume**”, “**清理云硬盘**”]

}

那么则将上述四个case变为

**{'添加项目，缺省配额': ['项目', '项目', '项目', '项目', '项目', '项目', '项目'],**

**'项目名称为空': ['项目', '项目', '项目'],**

**'创建1个云硬盘': ['云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘'],**

**'使用cirros镜像创建云硬盘': ['云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘', '云硬盘']}**

由此，将每一个列表看做一个有序的特征序列，则特征提取步骤完成。通过上述例子，我们直观的可以看到：

**'添加项目，缺省配额'是与项目功能结合最紧密的。**

**'创建1个云硬盘'与'使用cirros镜像创建云硬盘'他们用到的feature几乎完全相同。**

可以看到，这与我们直观上的经验是符合的，那么由此特征提取步骤则完成。

1. 相似度计算

从上述例子易得一二两个case之间相似度较高，而三四两个case之间相似度极高。那么下一步则主要在量化他们。

经过上一步的处理，计算case之间的相似度转换为了计算特征序列的相似度，不放将特征“项目”看做“a”，特征“云硬盘”看做“b”，那么直观上看，四个case则变为了：aaaaaaa, aaa, bbbbbbb, bbbbbbb。

接下来我试图用处理字符串的相似度的经验去处理它们。但是传统的字符串对于顺序十分看重（例如：three与there就是完全不同的意思），而我们的case则对于重现频率更加侧重。于是，我选择了dice距离，这是衡量两个集合之间相似度的一种距离，表达式为



但因为我们case的特点，我去除了集合的互异性，即只考虑重现的频数。这个指标可以衡量case/patch之间的内容相关性。

可是顺序对于我们来说也是有所影响的，举例说明：

假设有两特征序列：

rank(ai/bi) = 1 2 3 4 5

**a = ['项目' , '实例' , '项目' , '用户' , '项目']  
 b = ['实例' , '项目' , '实例' , '项目' , '项目']**

他们的元素相同，但是可以他们涉及到的功能的先后顺序不同。说明他们也并不完全相同。为了衡量顺序的差异，我借鉴非参数统计中秩和（排名的和）的概念，

做表达式为：



Pi为第i个特征关键词出现的频率，diffranki为两个序列中秩的差的归一化，其中秩和是用作衡量模糊顺序中常见的处理方式，衡量顺序的有效性由wilcoxon严格证明过。我们计算秩和的差，则是衡量出他们顺序的差异性，归一化是为了将order控制在[0, 1]，那么每一种feature都有一个秩和差，但是由于出现的频率不同我们不能将它们同等看待，所以则用加频率权的方式计算。

最终得到相似度表达式为：



dice衡量内容相关性，而order衡量顺序差异性，由于更加注重内容相关则将order系数降为0.2。特别地，引入示性变量I是因为如果内容相关度为0，那么讨论顺序则毫无意义，直接默认相似度为0。

下面以上述特征序列为例；试计算a b之间相似度，首先



其秩和分别为：

{'实例': 2, '项目': 9, '用户': 4}

{'实例': 4, '项目': 11}

秩和对应的差为：

{'实例': 2, '用户': 4, '项目': 2}

归一化后为：

{'实例': 0.25, '用户': 0.5, '项目': 0.25}

分别占比为：

{'实例': 0.3, '项目': 0.6, '用户': 0.1}

最终：



由此得到最终



## 应用范围

（1）

上述为计算两个case之间的相似度，以此类推，若我们有n个case，则可以构造一个n相关系数矩阵。那么每fail一个case就可以找出与其高度相关的若干case去跑。

（2）

将patch中的语法规范化后，将关键词添加进特征辞典中，则可以根据patch的特征序列匹配到与其高度相关的若干的case，则达到了推荐case的目的。

## 进展程度

现在已经可以由存n个case的json文件自动获得一个相关系数矩阵，但是特征辞典尚未写全，patch的规范尚未制定，根据相关性矩阵找相关的case名的工作尚未完成。