软件需求工程 LAB1

开源社区 SmartIDE 的需求分析分类

学校：南京大学

院系：计算机科学与技术系

姓名：许家铭、黄逸维、陈永健

学号：171860581、171860542、171860568

邮箱：381026323@qq.com

一、概述

本次实验是针对 SmartIDE 的需求分析分类工作。通过对开源社区的 Smart IDE 抓取他们开发和维护过程中的 commit 与 issue，从中分析获取他们所对应的需求，并进行分类。由于我们三年级学生还没有学习和使用过自然语言处理（Normal language process）和机器学习（machine learning），我们选择了使用 LUIS 来进行自然语言处理和分类，尽我们所能的完成了本次实验。

二、实验背景

在 IT 行业日益发达的现今，一个智能的 IDE 会成为很多程序员的得力助手，帮助他们迅速而简洁的完成任务。因此 smartIDE 在现今日益重要。而今天我们就将针对 SmartIDE 中的一个：开源社区中的 Visual Studio Code 进行需求分析与分类。

实验任务：

（1） 数据获取

从 GitHub 的开源代码中，我们爬取了其中的 issue 和 commit 的数据，根据提交的描述作为我们分类所使用的原始数据。

（2） 需求分析

对抓取的数据进行分析，分析他们主要包括的种类，主要有哪些需求，从而可以找出对应的应该有的分类标准，建立起 NLP 所需要的语料库

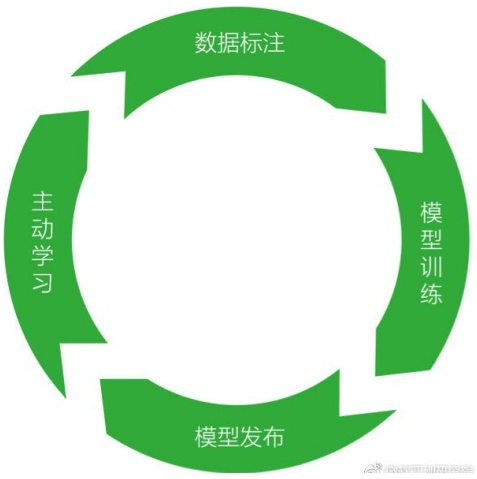
（3） 需求分类

根据之前建立起来的分类标准，对于所有的原始数据进行分类，放入他们所属的类别。

三、名词解释

1. LUIS

LUIS (Language Understanding Intelligent Service，https://www.luis.ai) 是微软发布的面向开发者的自然语言语义理解模块开发服务。



LUIS 开发流程首先是标注一定的起始数据，后训练得到语义理解模型，再对模型进行必要的测试，发布模型并应用到真实用户场景。根据应用日志继续标注更新模型，使其更加精确。

2. 分类标准

由于本次实验没有给出一个统一的分类标准或语料库，我们只能建立自己的分类标

准，并且遵照其进行语料库的建立。

分类标准：通过研究原始数据中的需求，我们小组通过讨论制定了一套需求分类标准。

①.编辑功能：与编辑代码有关的所有功能，包括锁紧功能，代码补全等。

②.文件管理：有关文件与文件夹的功能，包括文件的输入输出，保存，二进制文件支持等

③.编译功能：对于各种语言环境的支持，支持对于程序的编译等功能

④.扩展功能：支持各类插件和扩展按钮功能，包括 git 支持，API 支持等功能

⑤.界面设计：对于 UI 可视化等的支持，包括主题、字体、颜色等

⑥.软件稳定性：对于软件的稳定性的支持，如能够正确启动，不会出现失败或错误

语料库的建立：

本次实验中，为了使用 NLP 对数据进行处理，我们使用了自主建立的语料库。建立的方式主要是通过对于原始数据进行人工的贴标签分类。在小组合作后，将我们选出的语料库放入了在线文档中，这部分语料库我们也复制了一份，放在了文件夹中。

四、实验方法

首先，我们使用 Python 编写了两份爬虫，分别爬取了 vscode 项目中的 issue 和 commit。

其次，我们使用了腾讯在线文档，进行分工贴标签，并进行汇总，得到我们的语料库。

腾讯在线文档地址：

https://docs.qq.com/sheet/DSE1EZEJxcG5NdUJk?tab=BB08J2&c=N6K0A0

最后，我们将语料库输入 LUIS，建立起 NLP 模型，并且训练，测试后发布，形成 API 供我们进行调用。由于微软限制每个用户每个月免费测试条数为 1000 条，我们将测试数据分割成五份进行了分类。

五、实验过程

爬虫程序（~/ lab1_vscode 需求分类/数据抓取）：

72 lines (67 sloc) | 1.98 KB Raw Blame History

```python

import urllib
import urllib2
import re
import json
from bs4 import BeautifulSoup
import io
import sys
import openpyxl

record=[]
x = 4226

def gettitle(page):
        try:
                global record
                #sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoc
                url="https://github.com/microsoft/vscode/issues?page="
                data = urllib2.urlopen(url).read()
                z_data = data.decode('UTF-8')
                soup = BeautifulSoup(z_data, 'lxml')
                a = soup.select('a.link-gray-dark.v-align-middle')
                b = soup.select('span.opened-by')
                c = soup.select('relative-time')
                test = soup.select('div.float-left.col-8.lh-condensed.
                #hostsfile = open('record.txt', 'w', newline='',encodi
                for i in range(1,len(b)):
                        temp=[]
                        temp.append(a[i - 1].get_text())
                        temp.append("opened")
                        temp.append(c[i].attrs['datetime'])
                        z=""
                        for j in test[i - 1].select('a.d-inline-block.
                                z+=j.get_text()+'/'
                        temp.append(z)
                        #sn=b[i].get_text().replace(" ","").split('\n'
                        m = re.search('\d+',b[i].get_text())
                        temp.append(getdata(m.group(0)))
                        record.append(temp)
                #hostsfile.close()
                print(u'hosts刷新成功:',len(a))
                with open("issue.json",'ab+') as fp:
                    json.dump(record,fp=fp,ensure_ascii = True,indent
                record =[]
        except Exception as err:
                print(err)

def getdata(sn):
        value=""
        try:
                url="https://github.com/microsoft/vscode/issues/"+str(
                data = urllib2.urlopen(url).read()
                z_data = data.decode('UTF-8')
                soup = BeautifulSoup(z_data, 'lxml')
                a = soup.select('table > tbody > tr > td')
                #hostsfile = open('record.txt', 'w', newline='')
                for i in a:
                        value=value+i.get_text()+ "\n\r"
                        #hostsfile.write(value)
                #hostsfile.close()
                #print('hosts刷新成功:',len(a))
        except Exception as err:
                print(str(err))
        global x
        print(u"第"+str(x)+u"条抓取完成")
        x += 1
        return value

if __name__=="__main__":
                gettitle(175)
                print(u"第"+str(i)+u"页抓取完成")
```

语料库建立：

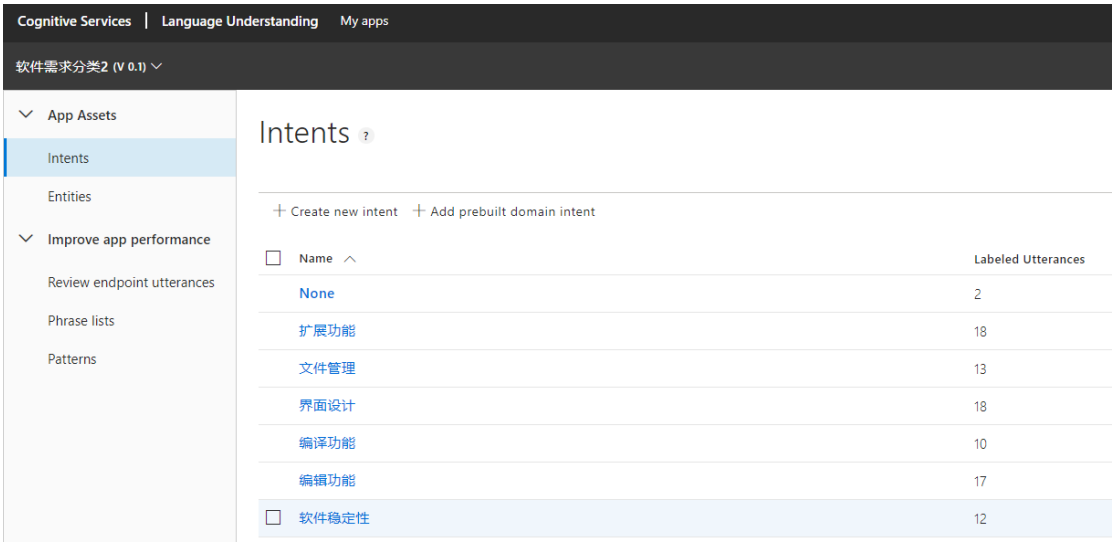
软件需求分类标准 只能查看

常规 .0 .00 微软雅黑 10 B I U S

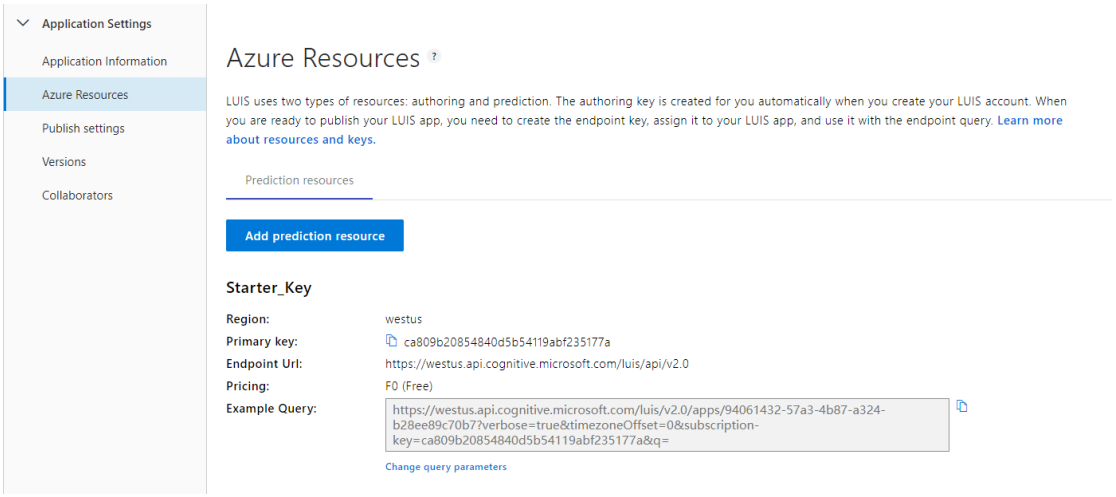
A8

	A	B	C	D
1	功能需求	编辑功能	editor	italic link , updated ReadMe file
2		文件管理		Remove executable bits from non-executable files
3		编译功能	compile	Process debug adapter messages in separate tasks
4		扩展功能	Support GitHub Des	Add tasks to user configuration
5	非功能需求	界面设计	ui	add or remove buttons
6		软件稳定性	launch VS code	unresponsive
7				

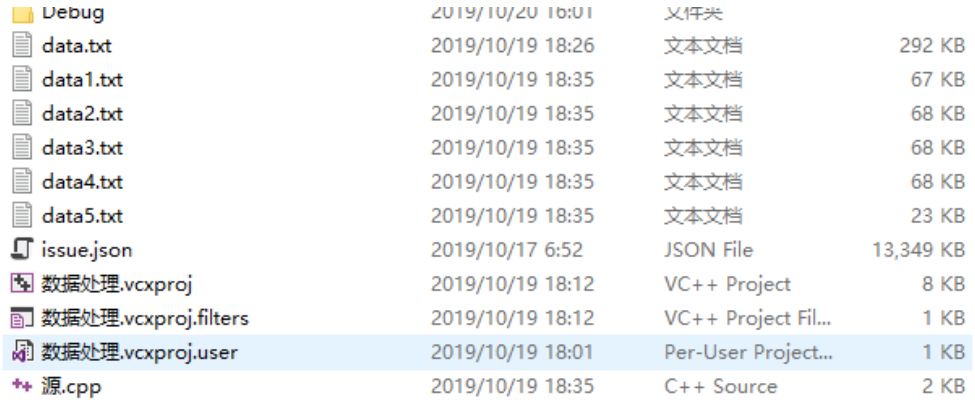
LUIS 模型建立：



API 的建立：



数据分割（~/ lab1_vscode 需求分类/数据处理）：



调用 API 的程序：

```python
import requests
import time

global null
null = ''

if __name__ == "__main__":
    f = open("data2.txt", "r")
    res = {"编译功能": [], "文件管理": [], "编辑功能": [], "扩展功能": [], "界面设计": [], "软件稳定性": [], "None": []}
    url = "https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/94061432-57a3-4b87-a324-b28ee89c70b7?verbose=true&timezoneOffset=0&subscription-key=ca809b20854840d5b54119abf235177a&q="
    for line in f:
        stri = url + line
        print(line)
        req = requests.get(url = stri)
        print(req)
        if req.text == '': break
        dic = eval(req.text)
        if "topScoringIntent" not in dic.keys(): break;
        type = dic["topScoringIntent"]['intent']
        print(type)
        res[type].append(line)
        time.sleep(1)
    print("result:")
    f1 = open("result2.txt", "w")
    for key in res:
        print(key, ":" , len(res[key]))
        f1.write(key + ":")
        f1.write(str(len(res[key])))
        f1.write('\n')
        for i in res[key]:f1.write(i)
        f1.write("\n")
```

以上使用的所有程序均附在 GitHub 中

六、实验结果

根据初步测试的结果来看，分类的精确度还是有一定的。结果示意图如下：

扁译功能:9

"Improve Syntax Coloring Scope",
"env variables failed to load in debug for nodejs app",
"[Debug] Expose the contribution.menus for Variable view",
"More flexible input variables: Multiple values & labels",
"Improve code block commenting",
"Improve CLI startup performance",
"javascript: highlight matching parens and braces in regular expression literals",
"Support lazy resolving of SignatureInformation#documentation",
"consider to allow for variable substitution in the inputs section",

文件管理:172

"Cannot expand code blocks after collapsing and closing the file",
"Emmet built-in plugin cheats start with @ not added in",
"Multi Cursor in different Files",
"Recent Workspaces list broken with UNC paths and root of a drive",
"tasks failing to run when workspace is on UNC path",
"Review places where we use `IDisposable[]` in the code",
"Files navigation feature in tabs section",
"Render multi-root workspace folders in multiple views",
"`process.stdout.write` in Extension Host does nothing",
"Progress bar for long file operations",
"Python increases tabsize in file with for loop in function",
"SharePoint SiteAssets files empty post saving and closing VS Code",
"Problem Updating VS Code - Blocked By AppLocker Due To Insufficient Information In Update Executable",
"Add a Windows Explorer Preview Handler for associated file types",
"Have a button to create a new file on the tabs stripe",
"Open unknown files with associated programs",
"Allow addition of individual files to a workspace",
"[json] End of file expected when formatting JSON.",
"Terminal:`Run active file` doesn't work in remote",
"Update workspace configuration before firing onDidChangeWorkspaceFolders event",
"Support file-level snippet variables. ",
"Enable FileSystemProvider to stat a file as readonly",

具体的结果我已经放入了~/ lab1_vscode 需求分类/数据存储/分类后数据中

七、结果分析

从我们测试所得的数据来看，我们分类的数据虽然有一定的精确度，但还是和我们人工分类结果有一定区别。

这些误差主要来源分别如下：

首先最主要的是，我们的语料库不够完整和全面。由于我在网上没有能够找到完整的软件需求方面的语料库，我们只能自主建立一个语料库。

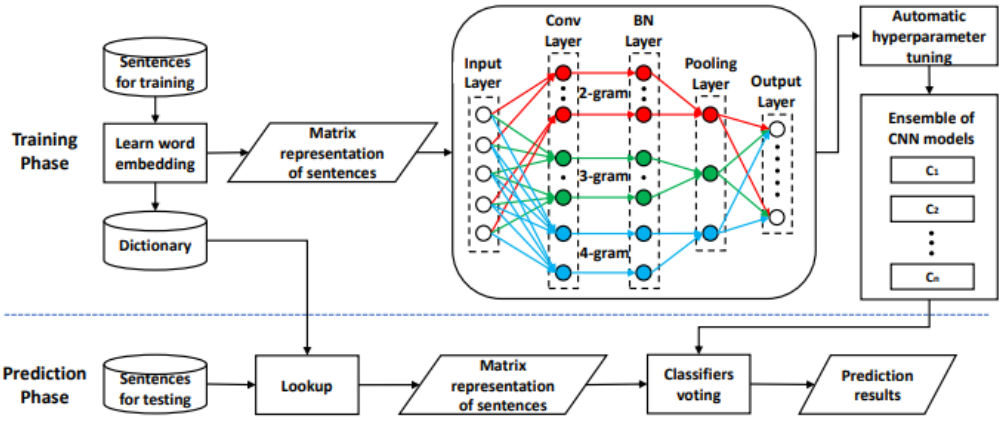
很显然，我们这种方法是不够的完整而精确的。我们的模型例句不够充足，由于我们没有学习过 NLP 和 ML，确定我们实验的方法并且学习就用掉了很大一部分时间，我们没有能够建立起完整精确的模型。若要进行后续实验，我们可以通过增加例句的形式来使我们的模型更加精确。

并且我们使用的是每个人各自去找例句分类的方法，相当于只有一个人在进行 label 的工作，这显然会导致我们贴上的标签不够客观，会有很强的个人理解在其中。如果有更多时间，我们可以通过从中抽取一份样本，每个人给样本进行分类，然后对我们的分类结果进行汇总，如果出现分类结果不同的情况，我们可以通过讨论或者进行投票，请更多人进行评判，从而获得一份更加符合大众价值观的分类标准和语料库。

接着是我们的分类标准不够清晰，由于没有统一的规范，我们查询论文时，也没有找到特别适合使用的分类标准，因此我们只能通过对于我们抽选的样本进行了分类，并且我们的分类之间的界定也没能做到很清晰，这一点我们可以通过对于分类进行精确定义来实现。后续也有待改进。

八、相关的实验

在查阅论文的时候，我注意到浙江大学的一位教授也在做这一方面的研究。他们使用的模型是这样的。



先对句子进行分割，再学习其中嵌入的词，通过矩阵计算其代表语句的权值从而算出该语句的最终属于的 intension。由于其中使用的分类标准和模型方法过于复杂，我们也没能复现出来。

论文的地址如下：

https://xin-xia.github.io/publication/tse185.pdf

九、结论

本次实验，我们使用了微软的 NLP 模块 LUIS 进行了自然语言处理与分类，取得的结果还是比较精确的，但是还是存在一定误差。在后续实验中，我们可以通过增加例句，界定分类标准，多人贴标签汇总等方式进行实验的细化。

在本次实验中，我们组的分数分配为每个人得分 1/3。

虽然我们没有学过 NLP 和 ML，但我们还是用我们自己的方式尽力地完成了实验。

十、参考文献

Huang Q, Xia X, Lo D, et al. Automating intention mining[J]. IEEE Transactions on Software Engineering, 2018.

https://docs.microsoft.com/zh-cn/azure/cognitive-services/luis/what-is-luis