Piazza问答平台历史数据分析及其在edx中的应用

张燕妮

　　　　　　　　　 (北京理工大学　北京　100081)

**摘 要** 随着MOOC在中国高校课程教学中的推扩，越来越多的学习者选择利用在线课程学习平台进行学习，如何提高在线课程学习的学习质量以及教学质量成为学习者与教师的关注重点。中国MOOC在线课程学习平台依托edX平台开发供学生进行在线课程学习，edX原有讨论区模块的主题繁杂，学生无法快速查找有用的信息的缺点，讨论区模块没有得到广泛使用。本文首先从Piazza问答平台获取数据并对数据进行分析，然后基于Open edX平台开发课件组件，实现edx平台对piazza平台数据的展现，让piazza平台的讨论记录信息能够在open edX平台中使用。该课件组件让学习者能够在edx平台直接查看piazza平台中的课程讨论信息，并实现讨论信息的按标签筛选功能，使学生在edX平台进行学习的过程中及时查找所遇到的疑问的相关讨论信息，提高学习效率，还避免了相同的问题被重复发贴提问的问题。不仅解决了edX原有讨论区主题繁杂，学生无法快速查找有用的信息的缺点，还把发展成熟的第三方问答平台与edX平台相结合，达到资源的合理高效利用。

**关键词** MOOC; Open edX; piazza问答平台; 数据分析; 讨论区

**The historical data analysis of Piazza and its application in edX**

Yanni Zhang

(Department of Computer Science, Beijing Institute of Technology, Beijing 100081)

**Abstract** With the widely using of MOOC in the Chinese university curriculum teaching, more and more learners choose to use it for learning. How to improve the quality of online courses learning and teaching quality has become the focus of attention of learners and teachers. China MOOC online course learning platform based on edX platform and used by students. The original discussion area module of edX has a shortcoming is that the subject is complicated, and the students can't find the useful information quickly. Thus, The module is not widely used . In this paper, we first get the data from the Piazza Q & A platform and analyze the data. And then developed a courseware components based on the Open edX platform. This components implement data showing which come from Piazza platform in the Open edX. The courseware components allow learners to directly view the discuss information of the Piazza platform in the Open edX platform. It also provides the function of information filtering by tags. Enable students to learn in the process of learning edX platform to find the relevant information in a timely manner to find the relevant discussion, improving the learning efficiency, avoiding a same problem repeatly posted. Not only solve the shortcomings of theme of the original edX discussion area are too complex, also combining third party inquiry platform with edX platform to achieve the goal of reasonablely and efficiently use of resources.

**Keywords** MOOC; Open edX; Piazza Q&A platform; data analysis

**目 录**

[1. 引言 4](#_Toc446404539)

[2. 研究背景及相关工作 6](#_Toc446404540)

[3. edX平台与piazza平台系统介绍 9](#_Toc446404541)

[3.1 edX平台系统结构 9](#_Toc446404542)

[3.2 Open edX平台组件 10](#_Toc446404543)

[3.2.1 LMS 10](#_Toc446404544)

[3.2.2 Studio（课程制作工具） 11](#_Toc446404545)

[3.2.3 XQueue 11](#_Toc446404546)

[3.2.4 XBlcok 课件组件 13](#_Toc446404547)

[3.2.5 其它组件 14](#_Toc446404548)

[4. Piazza数据获取与分析 15](#_Toc446404549)

[4.1 整体思路 15](#_Toc446404550)

[4.2 实现方法 16](#_Toc446404551)

[4.2.1 python脚本实现piazza平台的登录操作 16](#_Toc446404552)

[4.2.2通过API获得所需的json数据 17](#_Toc446404553)

[3.3 piazza平台 22](#_Toc446404554)

[4.3 数据存储 25](#_Toc446404555)

[4.4 数据更新 25](#_Toc446404556)

[4.5 数据解析与展现 25](#_Toc446404557)

[4.5.1 json数据结构分析 25](#_Toc446404558)

[4.5.2 handlebars模板引擎 29](#_Toc446404559)

[4.5.3 多级标签筛选 32](#_Toc446404560)

[5. 与Open edX平台的集成与测试 34](#_Toc446404561)

[5.1 编写XBlock 34](#_Toc446404562)

[5.2 XBlock安装到OpenedX平台 35](#_Toc446404563)

[5.3 结果展示 36](#_Toc446404564)

[6. 总结与未来工作 38](#_Toc446404565)

## 引言

MOOC是大规模网络开放课程（Massive Online Open Course）的简称，是面向社会公众的免费开放式网络课程[[[1]](#endnote-2)]。随着MOOC在中国高校课程教学中的推扩，越来越多的学习者选择利用在线课程学习平台进行学习，如何提高在线课程学习的学习质量以及教学质量成为学习者与教师的关注重点。由于在线课程学习与传统课程教学相比存在学习时间灵活，学习者背景多样化，学习地点不固定等特点，学习者之间，学习者与教师之间无法像传统教学那样在课堂上实现面对面交流。然而，学习者之间，学习者与教师之间针对所学内容的交流是学习者提高学习效率，教师了解学生学习情况的关键环节，因此在线课程学习平台必须具备实时，高效，方便的在线交流功能，才能使在线学习得以推扩。

edX平台作为目前MOOC领域三大在线课程平台之一，该平台提供了讨论区组件作为学习者与老师交流的工具。由于目前Edx平台提供的讨论区功能不够完善，仅利用edX平台提供的讨论区组件无法实现学习者之间，学习者与教师之间的及时、高效的互动。为了使在线课程学习达到更好的效果，学习者之间，学习者与教师之间的交流更加便捷高效，edX平台需要有一个更好的在线讨论工具，在线问答平台作为学习者之间，学习者与教师之间的交流媒介，实现课程参与者之间的相互交流，及时为学习者答疑解惑，便于教师了解学习者的学习情况。

目前教育类在线问答平台主要有Piazza，BlickBook，和Answer Underground。其中Piazza是一个专门为教师和学生提供的问答平台，集wiki和论坛模式于一体[[[2]](#endnote-3)]，该平台由Poooja Sankar 于2011 年7月创立，Piazza用户群体遍布美国各大知名高校，包括哈佛、斯坦福、MIT、哥伦比亚和普林斯顿等。Piazza平台相比其他问答平台有很多优点，更适合在线课堂教学使用。该平台根据学生和教员的身份不同，把学生发的post与教师发的post分开，学生和教员的回答也会分开，使讨论记录的浏览更加方便。而且每一条讨论记录都会被打上标签，用户可以根据标签筛选post，快速从大量的讨论记录中找到自己想要的讨论信息。另外，Piazza平台上的每一个讨论记录都有唯一的id，根据id能唯一检索到讨论记录。

本文通过对piazza课程讨论平台的数据进行分析，基于edx平台开发课件组件，实现edx平台对piazza平台数据的合理利用。把Piazza平台与在线课程学习平台Open edX相结合之后，能利用Pizza 平台的优势，把Piazza平台的讨论历史记录在Open edX平台上展示出来，方便学习者直接在Open edX平台上查看历史讨论记录，并且本文开发的课件组件还新增了piazza平台不具备的多级标签筛选功能，能够对历史讨论记录进行按标签多次筛选，快速找到学习者遇到的问题的答案，并且能够必免相同的问题被重复提问，从而提高学习者在edx平台中的学习效率，提高教学质量。

## 研究背景及相关工作

MOOC的出现预示着高校课程教学方式将发生历史性的转变，并且在2012年引发了一场教育风暴[[[3]](#endnote-4)]。应运而生的是大量的在线课程平台的涌现，其中Coursera, Udacity和edX是目前MOOC领域的三座大山[[[4]](#endnote-5)]。Coursera和Udacity都源于斯坦福大学；edX是由麻省理工学院和哈佛大学于2012年5月共同创建的大规模开放在线课堂平台，属于非盈利组织。edX 于2013年开放源码，Open edX是一个基于Web的平台，它可以生成、发布、分析在线课程，为edX.org以及其它在线教育网站提供软件框架。edX.org是edX平台的门户网站，该网站上可以进行很多高校的在线课程学习，如MIT，Harvard，Berkeley以及我国的北京大学和清华大学的部分课程都可以在edX.org网站上学习。我国的“学堂在线”平台也是一个基于Open edX的平台，它是国内首个MOOC中文平台，面向全球提供精品课程[[[5]](#endnote-6)]。在线课程平台的出现预示着高校课程教学方式将发生历史性的转变，在线课程平台使学习方式更加灵活，学生和老师都可以更加灵活的分配时间。

Open edX是一个开源的在线课程平台，由Open edX平台驱动的在线课堂平台有很多。例如：与很多知名高校合作的教育门户网站edX.org，该门户网站用户数量十分庞大，而且在不断增长中，edX.org为用户提供了非常好的在线学习体验。此外，斯坦福大学为在校学生和远程学生的在线学习服务的网站class.stanford.edu同样基于Open edX平台。Open edX不只为高校的学习教学提供支持，世界经济论坛以及与强生公司(Johnson & Johnson)旗下Ethicon Endo-Surgery等企业也创建了自己的open edX平台。我国的“学线在线”是一个基于Open edX的大规模开放在线课程平台，目前我国很多高校如清华大学，复旦大学，北京理工大学等都与“学堂在线”合作进行在线学习与教学。

Open edX平台引领了未在在线学习与教学的趋势，未来的发展中，不只在高校的课程学习方便会有广泛的应用，在医疗健康领域、企业以及政府部门的学习交流都有良好的应用空间与发展前景。因此，对Open edX平台进行扩展以适应不同领域的功能显得犹为重要，这也是Open edX于2013年开放源码的一个原因之一，源码开放之后Open edX平台的应用会更广，任何人都可以布署自己的Open edX平台，为Open edX平台扩展功能。在线课堂平台功能越丰富，学生的在线学习兴趣越强烈，随着在线课程学习的推广，在线课堂平台的功能将会越来越复杂多样。

自edX平台开源后，针对edX平台研究主要集中下三个方面：

一是对互评系统的优化。何升[[[6]](#endnote-7)]等人对同伴互评系统算法进行优化，提高了同伴互评系统的准确性与可靠性。文献[[[7]](#endnote-8)]研究了影响评分准确度可靠度的因素，构建了三种同伴互评的概率统计模型。

二是对学习行为的分析。学习行为的分析研究相对多一些，蒋卓轩[[[8]](#endnote-9)]等人针对学习行为与学习效果之间的关系进行了研究；王萍从学习者类型分析、学习者特征分析、学习者行为分析三个方面对ed X平台的中外学习者进行了研究[[[9]](#endnote-10)]；李曼丽[[[10]](#endnote-11)]等人分别对MOOC 学习者的课程参与和完成情况进行深入分析。

三是各类课件组件的开发。这些课件组件开发针对性较强，往往针对某个特定的课件需要而设计。

而针对edX平台讨论区研究方面主要集中在基于己有的讨论区进行教学研究[[[11]](#endnote-12)]，或直接在单独的第三方问答平台进行讨论而不依托于edX平台的讨论区[[[12]](#endnote-13)]，而edX平台的讨论区在功能上有很大局限性，目前并没有得到有效的改进。

中国大学MOOC设置了3个讨论区—教师答疑区、课堂交流区、综合讨论区。学生依据讨论区板块的划分，在对应板块发表相关主题。由于主题多而且繁杂，使得整个讨论区看起来比较混乱，学生无法快速找到对自己有用的信息。而Piazza平台在这点上做的比较好。在Piazza平台上学生和老师可以为自己的班级或者为一门课程在Piazza平台上建一个问答中心，学生在该问答中心内提问问题时会给问题打上不同的标签，对于一个问题老师和学生都可以参与回答，而且老师的回答和学生的回答会分开显示。由于每一个问题会被打上不同的标签，因此如果学生想要寻找一个问题的答案可以根据标签筛选问题，再在筛选出的问题中寻找答案，以这样的方式学生可以快速知道论坛上有没有自己想要的答案，如果确实没有人提过相关的问题，学生可以自己再新建一个问题，并为自己的问题打上标签，很快就会得到学生和老师的回复。由于Piazza平台上的标签设置使问题能够有效分类，学生能快速找到要想的答案，大大降低了相同问题的重复提问率，提高了学生的学习效率。另外，针对一个问题能把老师的回答和学生的回答分开，教师和学生能够有效互动。

本文把Piazza平台与在线课程学习平台Open edX相结合之后，能利用Pizza 平台的优势，把Piazza平台的讨论历史记录在Open edX平台上展示出来，方便学习者直接在Open edX平台上查看历史讨论记录，并且本文开发的课件组件还新增了piazza平台不具备的多级标签筛选功能，能够对历史讨论记录进行按标签多次筛选，快速找到学习者遇到的问题的答案。既解决了原有edX平台讨论区主题杂乱无章，不易查找的问题，又能把发展成熟的第三方问答系统与edX平台有机结合。

## edX平台与piazza平台系统介绍

### edX平台系统结构

Open edX平台的服务器端主要通过Python实现，部分基于Ruby on Rails。浏览器端主要通过Javascript实现，部分代码基于CoffeeScript。Web应用层框架使用Django框架下的Mako模版系统。使用SASS和Bourbon框架编写CSS。

Open edX平台有两个主要的系统：

* **LMS**(Learning Management System, 学习管理系统,以下简称LMS)，学习者和教师通过LMS进行交互，学习者通过LMS看到学习界面，教师通过LMS进行对课程的相关操作。课程存储在Mongo数据库中，学习者的信息存在MySQL数据库中。
* **CMS**(Content Management System,内容管理系统，以下简称CMS),CMS主要管理课程安排，进行课程发布，设置分级策略等，CMS和LMS的课程内容存储在同一个Mongo数据库中。

图3-1是简单Open edX的系统架构图。

图3-1 Open edX系统架构图

**Courses**

**(Mongo)**

**LMS**

**(MySOL)**

**XML Courses**

**(File system)**

**LMS**

**(Django)**

**CMS(Studio)**

**(Django)**

### 3.2 Open edX平台组件

#### 3.2.1 LMS

主代码库edX-paltfrom主要包括两个组件：LMS和Studio。

LMS学习管理系统面向学生显示内容以及进行一些交互式的应用，LMS能为学生提供多种多样的功能。通过LMS学生可以进行课程的注册，以及进行讨论，分级等。LMS对于数据的存储使用了两种数据库，课程的数据都放在Mongo数据库中，学生的数据都放在MySQL数据库中。Studio是edX的课程制作工具，教师可以通过Studio制作形式多种多样的课程，发布课程安排，发布练习等。

LMS具有四个特性，分别是运行时特性，类特性，成绩以及限制。

1. Runtime Features(运行时特性)

self.runtime有一个属性anonymous\_student\_id和一个方法publish()。

①anonymous\_student\_id: 匿名学生id，它是一个学生在特定课程中的唯一标识符，同一个学生在两个不同的课程中会有两个不同的id。

②publish(block,event\_type,event): 发射事件到周围的系统。XBlock能通过调用self.runtime.publish(self,event\_type,event)来发布事件，事件是包含任意形式的字典。参数event\_type使事件的下游处理可用，因为它唯一标志了模式。这个函数的调用将会引起运行时保存应用程序事件流中的事件数据。每当一个重要的状态发生改变的时候，XBlock应该发布事件。从事件流的事后分析可以得到XBlock是怎样在应用程序的环境中被使用的。通过仔细分析事件流，XBlock的状态能在任何一个历史点被重新建立。

1. Class Features(类特性)

通过XBlock提供的类属性或方法，可以在LMS中自定义行为。

①student\_view(XBlock view): student\_view()方法定义LMS中展示XBlock时将被呈现的视图。也被用来在Studio中的“preview”模式中渲染XBlock。

②has\_score(class property):如果这个XBlock应该出现在LMS 的progress页面则has\_score()方法返回值为true。

③icon\_class(class property):返回值可能是（other,video,problem）中的一个，返回值决定了在edX队列头中显示哪一个icon。

1. Grading(成绩)

为了加入课程成绩，一个XBlock要把has\_score设置为true.当成绩改变时应发布一个成绩事件。成绩事件是下面形式的字典：

{

‘value’:<number>

‘max\_value’:<number>

‘user\_id’:<number>

}

成绩事件代表了当前用户value/max\_value 的成绩，user\_id可选，如果它被省略则使用当前登录的用户ID

1. Restrictions(限制)

当User属性是UserScope.NONE时，XBlock不能修改field的任何值。

#### 3.2.2 Studio（课程制作工具）

课程制作工具有两个特性，分别是类特性和限制：

1. Class Features(类特性)

Studio\_view(XBlock.view):该视图会在Studio上生成一个编辑框。编辑器和LMS 的student\_view完全不同，该编辑框当且仅当在Studio中选择“Edit”时才会出现。

①Author\_view(XBlock.view):是和student\_view类似的一个可选视图，但是有内嵌编辑的能力。该视图与studio\_view不同，但student\_view类似。当使用Studio 预览XBlock时，比起student\_view,Studio更倾向于使用author\_view.

②non\_editable\_metadata\_fields(property):是XBlock.fields.Field对象的一个list.它不会在默认的Studio编辑视图中出现。

1. Restrictions(限制)

User属性不是UserScope.NONE时，XBlock不能修改field的任何值。

#### 3.2.3 XQueue

XQueue为LMS定义了一个和外部分级服务进行交流的接口。当一个学生在LMS中提交了一个问题时，LMS会把问题提交到XQueue，然后把问题通过外部服务分级，并且把返回响应到LMS。图2-2是XQueue与LMS和外部分级服务的关系图：

**LMS**

**XQueue**

**Passive Graders**

HTTP POST

HTTP POST

**Active Graders**

等待XQueue发送消息

响应已分级的消息

从XQueue 拉取消息

推送已分级的消息

学生提交的消息

图2-2 XQueue与LMS和外部分级服务关系图

LMS和XQueue的交互：首先LMS把学生提交的URL为 “/xqueue/submit”的HTTP POST请求的内容发送到XQueue。提交的内容包含一个回调URL，它标明了分级后的响应应该要发送到的地方。然后，当提交被分级后，XQueue会向LMS返回一个HTTP POST请求的响应到前面回调的URL。

XQueue和外部分级服务交互：有两种方式，第一种是Passive Graders,在这种方式下，passive graders等待XQueue发送给它提交，passive graders 把已分级的答复进行同步响应。具体的步骤是：首先LMS把消息发送到一个特定的队列; 然后XQueue检查它的设置，找到和消息相关联的URL的队列，把消息转发给该URL；接着，passive graders从XQueue接收一个POST请求并且把已分级的答复进行同步响应；最后，XQueue转发已分级的响应发送到LMS在它的原始消息中所提供的回调URL。另一种是Active Graders，在这种方式下，active graders把消息从XQueue中主动拉下来，而不是被动的等待，然后把响应推送回XQueue。具体的步骤是：首先，测试客户把消息发送到一个特定的队列；然后active graders 使用一个REST-like接口对XQueue调查。当它接收到一个提交，就会把一个响应返回给XQueue，返回时也是使用REST-like的接口；最后XQueue把响应返回到LMS。

#### 3.2.4 XBlcok 课件组件

XBlock是 edX.org 为了创建课件而产生的组件体系结构。edX课件是由分层联系的组件构建出来的。这些组件包括视频播放器，CAPA（The LearningOnline Network with CAPA）[[13]](#footnote-1)，以及像学习序列这样的复杂组件。XBlock是为这些组件开发的第二代API。XBlock和像LTI（Learning Tools Interoperability）[[14]](#footnote-2)这样已经存在的行业标准的区别是XBlock是一个Python语言水平的API，它提供了诸如存储数据的合理的默认值。XBlock可以封装在LTI中，可以把它写成一个LTI-XBlock,写一个XBlock最主要的目的是可扩展，可以把XBlock直接嵌入到课件中，而如果用LTI的话需要提供一个虚拟机镜像。

XBlock基于Django框架。Django是一个高级Python Web框架，鼓励快速开和清洁，务实的设计。原始的使用Python开发Web的方式是使用CGI标准，例如：先写一个输出html代码的python脚本，然后保存为CGI文件，通过浏览器访问该文件即可出结果。但是这种方法在处理数据库，代码重用等方面存在缺陷，开发者往往重复做了很多在框架设计上事，浪费大量精力，应运而生的是Django框架，它为开发者省去了很多浪费在框架上时间。Django使用MVC方法，即模型-视图-控制器。图3-3是MVC方法的体系结构简图：

**Model**

**模型**

**Controler**

**控制器**

**View**

**视图**

**数据库**

**用户**

用户输入

用户输入的指令和数据

存取数据

选择不同的视图

返回结果给用户

图3-3 MVC方法的体系结构简图

从图3-3中可以看出，MVC使得开发者，设计师和数据库管理员可以独立更改自己的模块而不影响其它部分。Django其实是一组用Python写的类库，它支持四种数据库：PostgreSQL, SQLite3, MySQL, Oracle. 使用Django开发项目时，它一般包含四个文件：

* \_init\_.py:把目录当成开发包
* manage.py:使你能与该Django项目进行交互的命令行工具
* settings.py:与项目相关的设置
* urls.py：该项目的URL设置

一般会把视图文件写在views.py中， urls.py中设置url和视图的对应关系。

#### 3.2.5 其它组件

1. CS Comments Service

CS Comments Service是一个独立的评论系统，它支持支票和嵌套的评论，它也支持也教育为目的的教员代言讨论平台。

①ORA（Open Response Assessor）

ORA从xqueue 安装提交，通过适当的机器学习分级，对等分级和成员分级，把结果返回给LMS。ORA会使用到edX-paltform和xqueue，它允许在edX平台开启响应问题的评价。

1. Discern

Discern是一个封装好的API,为任意自由文本分级响应提供服务，目的是提供一个高性能，可扩展的解决方案，能够有效的帮助学生学习，其中的反馈系统是重要的组成部分。

1. XServer

XServer从LMS接受学生代码的提交，使用课件分级运行代码。

## Piazza数据获取与分析

### 4.1 整体思路

从Piazza讨论区平台提取信息。提取的内容包括：

1. 所有的讨论记录。由于每一个讨论记录都对应一个唯一的cid，可以通过遍历所有的cid，向API发送POST请求从而获得所有讨论记录的json数据。
2. 所有标签的第一次筛选结果。先向API发送一个POST请求，从获得的json数据中提取出标签列表，根据标签列表再向API发送POST请求，获得按相应的标签筛选后的json数据，该数据作为以后实现多级筛选的基础。

提取信息的途径是通过piazza的API获得json数据。具体的实现思路如下：

首先，通过python脚本实现piazza平台的登录操作。分析浏览器登录piazza所需的请求头（header），根据分析的结果利用python脚本实现模拟登录piazza平台，并保存cookie，供以后直接访问piazza使用。

其次，通过API获得所需的json数据。利用python脚本实现模拟登录piazza平台之后，分析浏览器进行“点击某一条记录的快照，网页中显示该记录详细信息”这一行为所发送的POST请求。分析后发现：浏览器采用新的请求头向piazza的API发送POST请求，请求结果是返回json数据到浏览器端，浏览器端对json数据解析，实现讨论记录详细信息的显示。由此可以通过python脚本实现向piazza的API发送POST请求，实现json数据的获取。

最后对获得的json数据解压。

下面详细说明这三个过程。

### 4.2 实现方法

#### 4.2.1 python脚本实现piazza平台的登录操作

用流程图表示浏览器直接登录piazza以及python脚本模拟登录piazza的流程如图4-1和图4-1所示：

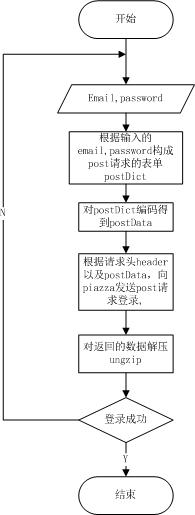
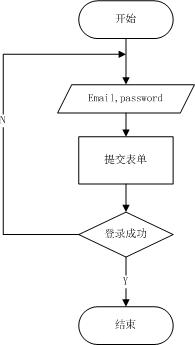


图4­-1 浏览器登录piazza 图4-2 利用python脚本登录piazza

下面对python脚本登录piazza所涉及的关键内容进行分析：

利用urllib.request, http.cookiejar两个库创建opener实现自动在发出的POST请求中加上自定义的请求头，以及自动处理使用opener过程中遇到的cookies的功能，创建opener以及利用opener发送post请求的代码如下：

//创建opener

cj = http.cookiejar.CookieJar()

pro = urllib.request.HTTPCookieProcessor(cj)

opener = urllib.request.build\_opener(pro)

urllib .request .install\_opener(opener)

//利用opener发送post请求，接收返回的数据并解压

req = urllib.request.Request (url,postData,header)

response=urllib .request .urlopen(req)

data=response.read()

data = ungzip(data)

urllib.request.Request (url,postData,header) 的参数说明：

“url”: 发送POST请求的目标url，url=” <https://piazza.com/class/>”,

“postData”: POST请求的参数，登录时所需的参数包括：email, password, from, remember四项，其中email, password即用户登录的邮箱和密码，from：“signup”, remember: ”on” 。这四项内容以json格式组合即：postDict={ 'from': '/signup','email':email ,'password': password , 'remember': 'on' }，然后要对postDict编码才能作为urllib.request.Request的参数。调用urllib.parse包中的urlencode方法对postDict编码。postData = urllib.parse.urlencode(postDict).encode()。

“header”: 自定义的请求头。Header通过分析浏览器行为获得，python脚本中发送POST请求时利用此header让服务器端认为是由浏览器发出的POST请求，以达到模拟浏览器操作的效果。

#### 4.2.2通过API获得所需的json数据

流程图4-3和4-4分别表示浏览器获得某条讨论记录的详细信息以及通过python脚本获得讨论记录详细信息：

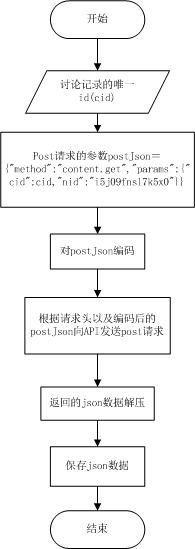


图4-3 浏览器获得某一条讨论记录详细信息流程图 图4-4 python脚本获得某一条讨论记录详细信息流程图

从获得某一个讨论记录详细信息的json数据出发，然后把方法推广到获得所有讨论记录详细信息数据，以及获得按标签筛选后的讨论记录快照列表的数据。

通过在浏览器的Piazza页面上进行“点击讨论记录快照列表中某一项”这一操作,分析浏览器处理这该动作所涉及到的网络请求,得到如下的信息:当点击讨论记录快照表表中某一项时,浏览器会向服务器端发送一个POST请求,请求的url为: <https://piazza.com/logic/api/>. 另外，POST请求的请求头header也和登录时不一样, 请求的参数包括两项内容”method”和”params”.参数格式为{"method":"content.get","params":{"cid":nr,"nid":"i5j09fnsl7k5x0"}}.其中,”method”:”content.get”表示的是向API请求讨论的详细内容,如果向API请求的是讨论记录快照则”method”:”network.get\_my\_feed”. ”params”是相应的参数,”cid”:表示的是点击的讨论记录快照所对应的ID, nid表示的是课程id.

接下来用Python实现上述操作。通过API向服务器端获得json数据实质上也是向服务器端发送POST请求，因此基本的代码和登录piazza时一样，只是改变了urllib.request.Request(url,postData,header\_new)中的参数的值。此时的url变为：”https://piazza.com/logic/api/”，header\_new是新的请求头可以通过浏览器行为分析中获得，postData是编码后的postJson ， postJson={"method":"content.get","params":{"cid":nr,"nid":"i5j09fnsl7k5x0"}}，对postJson编码调用的是json.dumps(postJson).encode()方法。POST请求的返回结果需要进行预处理以便于下一步的操作。预处理时需要对返回结果进行解压，解码，然后处理中文。关键的代码如下。

data = ungzip(data)

data=data.decode()

#处理data的中文

myjson=json.loads(data)

newjson=json.dumps(myjson,ensure\_ascii=False)

data是POST请求的返回结果，首先调用ungzip()对其解压，然后解码，解码后得到的数据中文不能正常显示，通过json.loads()以及json.dumps()处理中文。最后得到的newjson即为预处理后的json数据。

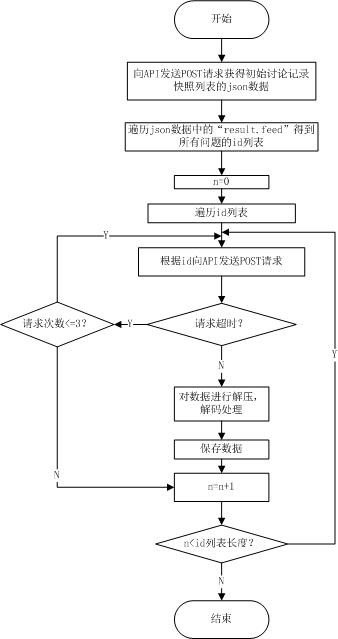
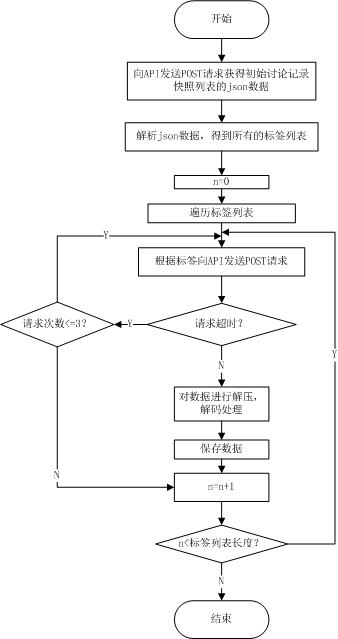
有了上述获得某一条指定的讨论记录json数据这一核心的方法后，就可以推而广之来获得所有的讨论记录json数据。首先需要得到一个所讨论记录的id列表，然后遍历这个列表，向API发送POST请求，获得讨论记录的详细信息数据。为了获得所有讨论记录的id列表，先向API发送POST请求获得初始时的讨论记录快照列表数据，数据中包含了所有讨论记录的快照，对数据解析可以得到id列表。另外，数据请求时有时因为网络问题请求时间太长，这里通过设定一个默认的请求时限，超过时限没有响应则显示超过再次发起本次请求，如果连续三次请求超时则跳过该请求。请求时限的设定由socket.setdefaulttimeout()实现，脚本中设定的值为20s。具体的流程见图4-6。

图4-6 获得所有讨论记录详细信息流程图

下面讨论获得按标签筛选后的讨论记录快照列表的json数据的方法。仍然通过分析在浏览器piazza页面进行“点击某一个受欢迎的标签列表中的标签”这一动作，浏览器的所涉及的操作。分析后得知，进行这一操作时浏览器仍然向同一个url发送post请求，只是请求时的参数不一样了。与”点击讨论记录快照列表中某一项”这一操作的区别在于postJson的构成不同。例如要获得按标签”lab1”筛选后的讨论记录快照列表，则postJson={"method":"network.filter\_feed","params":{"nid":"i5j09fnsl7k5x0","sort":"updated","filter\_folder":"lab1","folder":1}}。”filter\_folder” 的值即为标签的名称。

通过上述分析可以知道，获得json数据的实质是向“<https://piazza.com/logic/api/>”这个url发送POST请求，根据请求的参数不同，会返回不同的数据，然后对数据进行解压，解码处理得到json数据。具体的流程图如图4-7所示。

图4-7 获得所有标签筛选后讨论记录快照流程图

通过上面的方法已经得到了从piazza平台提取的数据。

### 3.3 piazza平台

Piazza平台上以每一门课程为单位的页面组织方式基本上分为两大部分：第一部分，导航栏。导航栏中包含了课程信息，用户设置等导航栏所应具备的基本内容。第二部分，主页面。主页面是需要研究的页面组织结构，主页面分为两大块：第一块是标签列表。标签列表位于导航栏下方，这里列出了该课程的所有标签，当用户点击一个特定的标签时浏览器会向Piazza的logic/API发出过滤请求,从API获得被打上该标签的所有问题的json数据，并把问题列表显示在主页面的左边栏，即图4-2中主页面的第二块中的第一小块，显示的是过滤后的索引列表，并按时间排序。主页面的第二块是视图，它分别两小块，第一小块是前面提到的问题列表，第二小块是每一个问题的详细信息，在第一小块中选中某一个问题时同样会向/logic/API发出请求，这时候请求的获得一个问题的信息。返回的结果会在第二小块显示该问题的所有信息，包括标题，内容，学生的回答，老师的回答以及其他的所有回复。其中,每一个讨论记录只对应唯一的一个学生的回答(the students’ answer);每一个讨论记录也只对应唯一的一个教师的回答(the instructors’ answer);而后续的讨论(follow up discussions)可以有多个，而且针对每一个后续讨论可以有多个回复。图3-4是piazza平台的页面组织结构简图

图3-4 piazza页面组织结构图

其html结构为：

第一部分导航栏对应的是<div id="top\_bar" class="navbar top\_bar">；

第二部分主页面<div id="page\_main" class="has\_popular\_tags\_bar">。在第二部分里分为两大块，第一块标签列表<div id="bars" class="bars">，第二块是视图<div id="views" class="views">。视图模块分为问题列表和问题详细信息两小块：<div id="feed" class="page\_feed" style="width: 350px;">是问题 列表，问题详细信息<div id="page\_center" style="left: 350px; bottom: 0px;">。

### 4.3 数据存储

这些数据都是以json字符串的形式保存的。由于数据量不是特别大，为了节省开支不需要存放在一个特点的数据库中，以文件的格式保存即可，为了便于后续在Open edX平台上的集成，以及数据的存取方便，把这些数据保存在github的一个仓库中。每次更新后的数据都会同步到github仓库中，数据的保存格式为.json文件，以讨论记录的id为文件名。

### 4.4 数据更新

利用存放于edX服务器中的python脚本自动执行，实现对piazza平台更新数据的检测与获取，脚本的功能为检测有更新的post，然后获取json数据，并用获取到的新json数据覆盖仓库的旧数据，而那些没有更新的post数据则不会重新获取，这样大大减少了系统开销。保证edX课件组件中浏览到的信息与piazza平台上的信息的一致性。

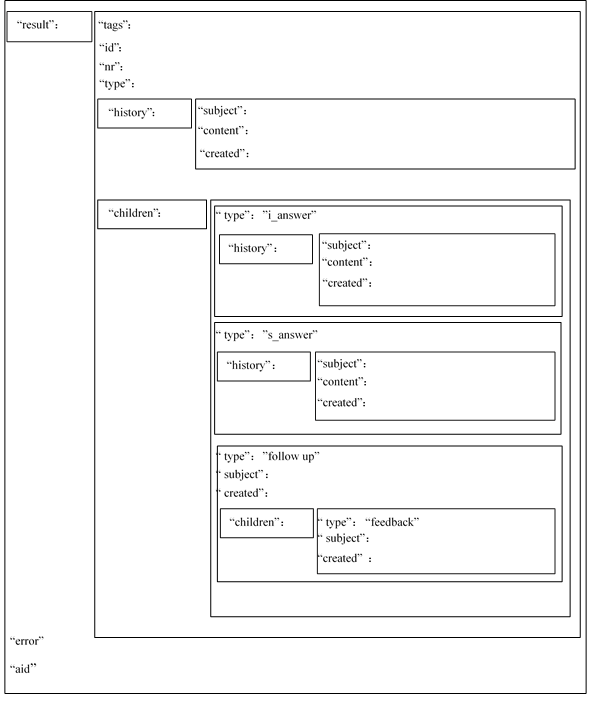
### 4.5 数据解析与展现

对json数据进行解析是最重要的一部分内容。首先对所获得的json数据的结构进行分析。然后根据json数据的结构，以json数据为数据源利用js模板引擎――handlebar进行数据解析，生成js模板，并利用css渲染，最后以html页面的形式展现出来。

#### 4.5.1 json数据结构分析

提取到的数据主要有两种：一种是讨论记录的详细信息，另一种是按标签筛选后的讨论记录快照。这两种json数据的结构存在差别，而每一种数据各自的结构是相同的。

讨论记录的详细信息。其结构如图4-1所示。

图4-1讨论记录详细信息数据结构简图

一个讨论记录详细信息数据以json字符串的形式保存在文件中，进行解析的时候先把json字符串转化为json对象，每一个json对象的格式如图4-1所示，注意图4-1中只列出了有用信息的名称，从图中可以清楚的了解到json对象的格式。所需的信息存在于“result.history”以及”result.children”这两个字面量中。

Result.history是一个对象数组，这个数组又由对象字面量组成，每一个{}所代表的是一个对象字面量。对象字面量中存储的是post的内容，每当post的发表者进行修改的时候，数组中会相应的增加一个对象字面量。在进行json解析时，只需要读出数组中第一个对象字面量的内容即可，第一个对象字面量的内容就是post内容的最新版本。 Result.children是一个对象数组。它包括三种对象，这三种对象按”type”的值不同来区分，”type”的值为” i\_answer”时该对象存储的是instructor针对该post的回答。”type”的值为” s\_answer”时该对象存储的是student针对该post的回答。”type”的值为” followup”时该对象存储的是后续的讨论。当”type”的值为” followup”时，对象中又包含数组，后续的讨论中会有回复，回复的内容会存储在数组中。

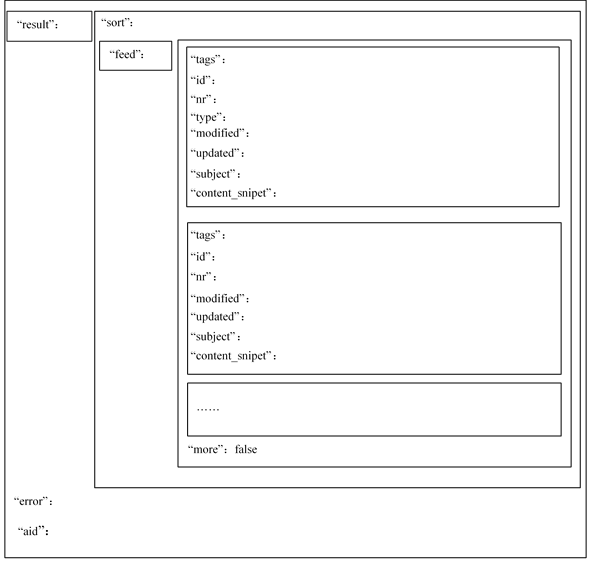
按标签筛选后的讨论记录快照，其结构如图4-2所示。

图3-2 按标签筛选后讨论记录快照json数据结构简图

标签筛选后讨论记录快照的json对象的格式则简单得多，每一个快照以对象数组”result.feed”中的一个对象的形式存在。每一个对象包含一系列的“名称－值”。对这些“名称”的含义作说明。”tags”的值表示的是该快照所包含的标签，是一个数组；”id”的值表示的是该快照的id，由14个字符的字符串表示；”nr”的值表示的是该快照的id，是一个整数；”modified”的值表示的是修改时间；”updated”:的值表示的是更新时间；”subject”的值表示的是快照的主题，与图4-1中“result.history.subject”的值相同；”content\_snipet”的值表示的是快照的内容，是图4-2中“result.history.content”中的一部分内容。

#### 4.5.2 handlebars模板引擎

清楚了json数据的格式之后，下面需要做的是把json数据对象的内容展示在页面上。

关于页面的展示采用的是”json+ajax+html”的方式。这种方法是基于ajax请求，要求服务器端返回一个json类型的json字符串，这个json串中包含了界面所需的所有信息，界面拿到json串后，构造出html。

完成界面展示利用javascript模板引擎可以有效的组织和分离前端页面代码中的显示层和数据层两个部分。Javascript模板引擎有很多种，本文中使用的是handlebars模板引擎。handlebars以json对象为数据源，支持逻辑判断、循环等操作，同时具有非常好的扩展性。

下面介绍利用handlebars模板引擎完成数据的展示的方法。

1. 创建html页面

①在<head>部分添加js库，包括两个js库，一个是jquery的js 库（jquery-1.11.3.min.js）；另一个是handlebars的js 库(handlebars- v3.0.3.js)。

②添加handlebars模板。

<script id=”page-center-template” type=”text/x-handlebars-template”>

　　　Template content　//对json数据的解析

</script>

（2）创建js

使用handlebars.compile方法将模板编译为函数，生成的执行函数接受json数据作为参数，用来渲染模板

//获得html中定义的模板

var source = $("#page-center-template").html();

//将模板编译为函数

var template = Handlebars.compile(source);

//生成的执行函数接受context作为参数

var html = template(data\_json);

//渲染模板

$("#page\_center").html(html);

下面以第二部分——讨论记录快照列表模板的生成为例详细介绍如何利用handlebar模板引擎生成js模板实现动态加载html页面。

1. 文件结构

主要包括三类文件：.html，.js，.css 如表2所示。

表2：文件名与文件介绍对照表

|  |  |
| --- | --- |
| 文件名 | 文件介绍 |
| example.html | 实现讨论记录快照列表的html页面 |
| handlebars-v3.0.3js和jquery-1.11.3.min.js | 利用handlebars模板所需的js脚本 |
| piazza\_feed.js | 对模板进行编译以及其它函数功能实现，利用ajax获取json数据的js脚本 |
| dashboard\_feed.js | 渲染html页面的css文件 |

1. 具体实现

（1）创建html页面

①在<head>部分添加js库，包括两个js库，一个是jquery的js 库（jquery-1.11.3.min.js）；另一个是handlebars的js 库(handlebars-v3.0.3.js)。

②页面框架

<div id="page\_main" class="has\_popular\_tags\_bar">

<div class ="views" id="views">

<div id ="feed" class="page\_feed">

</div>

</div>

</div>

讨论记录快照列表的内容显示在<div id ="feed" class="page\_feed"></div>之内，具体的显示格式定义在下面的handlebars模板中。

③添加handlebars模板。

<script id=” feed-template” type=”text/x-handlebars-template”>

　　　　　　　 //template content

</script>

模板内容的编写是实现的重点部分之一，利用的是handlebars的语法规则，解析json数据，获取所需的内容使其在模板的对应位置显示。内容用{{{　}}}获得。其中进入到属性上下文的方法为{{#with }}{{/with}}，遍历数组的方法为{{#each}}{{/each}}。例如：获得快照相对应的时间，快照的标题，快照的内容：

　　 {{#with result}}

　　　{{!-- 按组显示（遍历所有组） --}}

{{#each feed}}

　　　　　　　　{{!--日期 --}}

　　　　　　　　{{{modified}}}

　　　　　　　　{{!--标题 --}}

　　　　　　　　{{{subject}}}

　　　　　　　　{{!--内容片段 --}}

　　　　　　　　{{{content\_snipet}}}

　　　　　　{{/each}}

　　　　{{/with}}

另外，当需要进行条件判断时，需要注册helper来实现。例如定义比较大小的helper 则可以在html进行使用，使用方法如下：

{{#compare @index 0}}

….

{{/compare}}

可以判断数组的下标是否为0,若为0则执行相应的操作。

该比较大小的helper的注册在js脚本中实现，实现方法为：

//注册一个比较大小的Helper,判断v1是否等于于v2

Handlebars.registerHelper("compare",function(v1,v2,options){

if(v1==v2){

//满足添加继续执行

return options.fn(this);

}

else{

//不满足条件执行{{else}}部分

return options.inverse(this);

}

});

（2）创建js

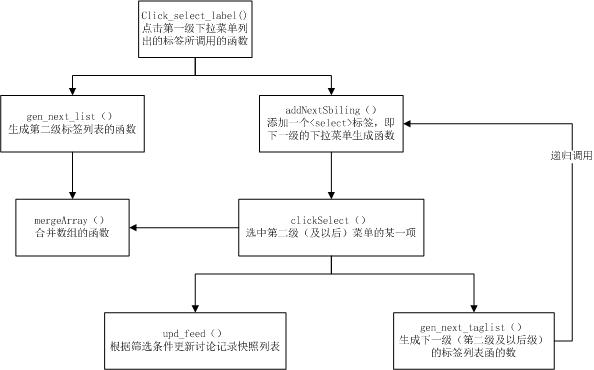
页面加载时的初始化操作包括：导航栏显示“popular tags”以及生成多级筛选的第一级下拉菜单；讨论记录快照列表部分显示的是按时间倒序排列的快照列表；注册一个比较大小的helper。前两项内容的显示都涉及到数据的动态获取以及显示的问题，通过ajax实现从github获得json数据，然后利用handlebar模板引擎实现json数据的解析与显示。当在前台的页面进行相应的操作时，页面需要根据操作进行响应，通过js中的函数实现对页面的响应

#### 4.5.3 多级标签筛选

Piazza平台只提供了标签的一次筛选功能。为了能够快速，准确的筛选讨论记录，找到所需的讨论记录信息。实现了按标签多级筛选的功能。基本的思路是：第一级菜单中列出所有的标签列表，当选中某一个标签时，进行第一次筛选，本次筛选直接利用piazza平台已有的标签分类信息，并且列出经过第一次筛选后的第二级可选标签。以后的每次筛选都在上一次筛选的结果基础上进行筛选，并列出下一级可选的标签，可以不断进行筛选。筛选的形式为动态弹出多级下拉菜单。

例如：当选中第一级下拉菜单的标签中“lab1”标签时，会在feed部分以摘要的形式显示与lab1有关的所有记录（集合a1），同时会生成第二级下拉菜单（第二级下拉菜单所列出的标签为：集合a1包含的标签的并集-“lab1”。）当选中第二级下拉菜单的标签“lab2”时，会在feed部分以摘要的形式显示出同时与“lab1”, ”lab2”有关的所有记录（集合a2）, 同时会生成第三级下拉菜单（第三级下拉菜单所列出的标签为：集合a2包含的标签的并集－（”lab1”,”lab2”））.如此反复进行。

具体的实现函数调用关系如图4－1所示：当点击第一级下拉菜单列出的某一个标签时，调用函数click\_select\_label()，该函数会调用两个函数生成第二级下拉菜单， addNextSbling()生成下一级下拉菜单，下拉菜单中包含的标签列表由gen\_next\_list()生成。当点击第二级以及以后各级下拉菜单时，调用函数clickSelect(),该函数会调用两个函数生成下一级下拉菜单， addNextSbling()生成下一级下拉菜单，下拉菜单中包含的标签列表由gen\_next\_taglist()生成。递归调用关系是通过addNextSbling()函数中监听点击事件，触发clickSelect()函数执行实现的。

图4－1 多级筛选函数调用关系图

## 与Open edX平台的集成与测试

首先，编写XBlock，通iframe形式显示piazza历史记录的展示页面。

然后，把XBlock安装到Open edX平台，piazza历史记录展示页面即可作为该平台的一个课件组件使用。

### 5.1 编写XBlock

利用XBlock-sdk进行可以快速生成一个初始XBlock。然后在此基础上按照需要开发。

由于XBlock-sdk进行了更新，生成初始XBlock的方法与早先的版本有一些差别，在此说明利用更新后的XBlock-sdk生成xblock的步骤。

1. 安装标准开发库 (更新前的XBlock-sdk没有这一步)

sudo apt-get install python-dev libxml2-dev libxslt-dev lib32z1-dev libjpeg62-dev

1. 从github克隆一份 xblock-sdk

sudo git clone <https://github.com/edx/xblock-sdk.git>

1. 进入到xblock-sdk目录
2. 安装虚拟环境virtualenv

pip install virtualenv

1. 为XBlock创建虚拟环境

sudo virtualenv Piazza\_xblock

1. 激活虚拟环境

source Piazza\_xblock/bin/activate

1. 安装依赖，注册XBlock

make install

1. 运行bin下的workbench-make-xblock，创建新的xblock

cd /xblock-sdk/bin

./workbench-make-xblock

输入shortname:piazza

输入Classname:PiazzaXBlock

创建成功！

XBlock创建成功后，把piazza历史记录的html页面(example.html)，作为iframe嵌入到piazza.html中。然后把XBlock安装到Openedx平台。

### XBlock安装到OpenedX平台

1. 安装XBlock

sudo –u edxapp /edx/bin/pip.edxapp install /piazza-xblock/piazza/

1. 使XBlock可用（该步骤只需首次操作，之后可跳过）

①在edx-platform/lms/envs/common.py中去掉注释：

*# from xmodule.x\_module import prefer\_xmodules*

*# XBLOCK\_SELECT\_FUNCTION = prefer\_xmodules*

②在edx-platform/cms/envs/common.py,中去掉注释：

*# from xmodule.x\_module import prefer\_xmodules*

*# XBLOCK\_SELECT\_FUNCTION = prefer\_xmodules、*

③在edx-platform/cms/envs/common.py中把

*'ALLOW\_ALL\_ADVANCED\_COMPONENTS': False,*

*改成：*

*'ALLOW\_ALL\_ADVANCED\_COMPONENTS': True,*

（3）在Studio中把Xblock添加到课程的高级设置中。

①登录到Studio,打开课程

②settings->Advanced Setting

③把键”advanced\_modules”的值改piazza.

（4） 把piazzaXblock添加到课程，在studio中

①Edit编辑一个单元

②Advanced->piazza

### 结果展示

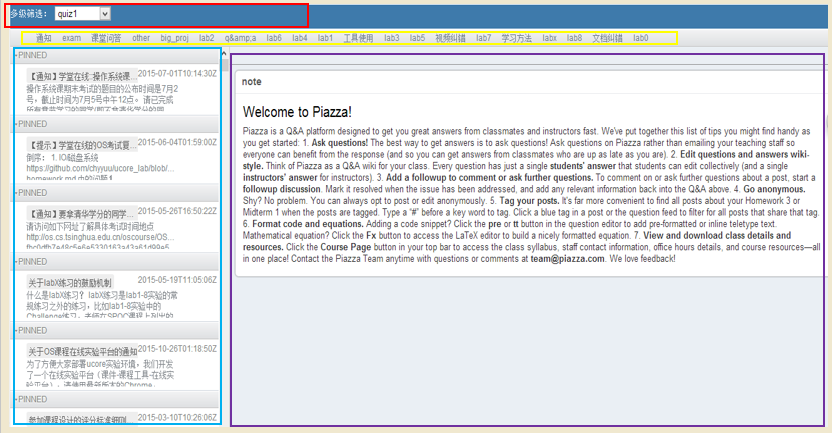
初始加载后如图5-1所示。页面最上方（红色框线部分）是多级筛选功能栏。可以进行多次筛选，筛选的结果会在feed部分（蓝色框线）显示，如图5-2是进行了两次筛选（quiz5,lab1）之后的结果，有三条记录满足条件。当选中feed部分的某一条记录时，会在主页面部分（紫色框线）显示选中记录的详细信息，如图5-3。黄色框线内是受欢迎的标签，点击其中的任何一个标签，会进行一次筛选，筛选结果也显示在feed部分（蓝色框线）。

图5-1 初始加载结束后的页面

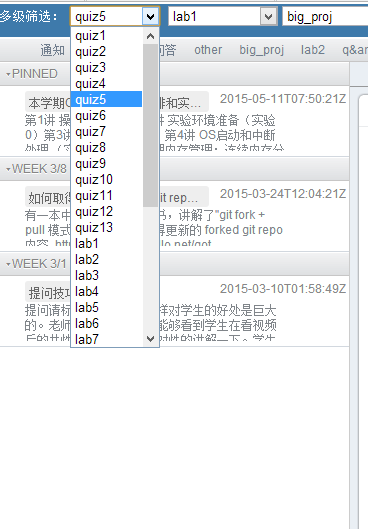
图5－2 利用多级筛选功能进行两次筛选结果图

图5－3 点击某一个记录，显示其详细信息

## 总结与未来工作

本文针对edX原有讨论区模块的主题繁杂，学生无法快速查找有用的信息的缺点，基于Open edX平台开发了新的课件组件，实现对第三方问答平台Piazza历史讨论记录的浏览以及按标签多级筛选的功能，有机结合了Piazza平台与Open edX平台。原有edX平台讨论区主题杂乱无章，无法快速准确查找到有用的讨论信息，不利用在线课程的学习。而Piazza问答平台专门针对学生和教师课堂问答及讨论而开发，具有讨论信息分类清晰明确，学生与教师回答分开，教师发贴标注等优点，学生可根据标签筛选出讨论记录信息，该平台广泛运用于高校课程教学中。本文基于Open edX平台开发的课件组件，能让学生直接在Open edX平台中浏览以及筛选piazza平台上的讨论信息，当学生在学习过程中遇到疑问时能够直接在Open edX平台查找相关的讨论，该课件组件不当能像Piazza平台上根据单个标签筛选出讨论信息，还增加了Piazza平台尚不支持的多级标签筛选功能，使讨论信息的查找更高效，还避免了相同的问题被重复提问。同时还能直接根据每个Post的唯一id编号，定位到单个特点的讨论记录。利用该课件组件，不仅解决了edX原有讨论区主题繁杂，学生无法快速查找有用的信息的缺点，还把发展成熟的第三方问答平台与Open edX平台相结合，有效达到资源的合理高效利用。

未来的工作主要有：一是完整该课件组件的功能。目前筛选主要是按标签进行筛选，想要进一步完善可实现按关键字进行全文筛选，这里涉及到计算开销等问题需要深入研究。另外还可以按日期进行筛选，目前筛选的结果按日期进行倒序排序，可进一步开发按日期筛选功能，筛选出特定日期的讨论记录，其他筛选方法如按发贴人筛选等扩展功能。二是对piazza数据分析，进行学生学习行为分析，得出学生讨论参与程度相关信息，帮助教师最后评分。还可以分析学生讨论参与程度与学生最后成绩的关系。

**参考文献**

1. [] 方静.MOOC的发展及其对传统教育的挑战[J].华北科技学院学报,2014，(5):90-94. [↑](#endnote-ref-2)
2. [] Piazza. “Ask.Answer. Explore.Whenever”. https://piazza.com/ouc.edu.cn/ [↑](#endnote-ref-3)
3. [] 王左利. MOOC:一场教育风暴要来了吗?[J].中国教育网络,2013,(4):11-15. [↑](#endnote-ref-4)
4. [] 陈小蒙.Coursera,Udacity,edX:MOOC(大规模在线开放课程)的三座大山[OL].http://www.36kr.com/p/170352html. [↑](#endnote-ref-5)
5. [] 王书瑶,王小根,晋步.中外MOOC课程对比研究――以edX和“学堂在线”为例.[J].软件导刊,2014,13(7):155-156. [↑](#endnote-ref-6)
6. [] 何升, 邓伟林, 肖体斌, . MOOC中基于二分图推荐的同伴互评系统优化[J/OL]. 计算机应用研究, 2016 (05): [↑](#endnote-ref-7)
7. [] Piech C, Huang J, Chen Z, et al. Tuned models of peer assessment in

   MOOCs[EB/OL]. [2015-03-02] . http: //www. stanford.edu/~cpiech/bio/papers/tuningPeerGrading. pdf [↑](#endnote-ref-8)
8. [] 蒋卓轩,张岩,李晓明. 基于MOOC数据的学习行为分析与预测. [J].计算机研究与发展.2015,52(3):614-628. [↑](#endnote-ref-9)
9. [] 王 萍. 基于 edX 开放数据的学习者学习分析.[J]. 现代教育技术.2015,25(4):86-93 [↑](#endnote-ref-10)
10. [] 李曼丽,徐舜平,孙梦嫽. MOOC 学习者课程学习行为分析

    ———以“电路原理”课程为例.[J].开放教育研究.2015,21(02):63-69 [↑](#endnote-ref-11)
11. [] 杨芳，魏兴，张文霞. MOOC edx 讨论区的协作学习模式探析

    ———以英语会话技巧的教学实践为例.[J].外语电化教学2015.116：60－68 [↑](#endnote-ref-12)
12. [] 郑海永，任新敏. 基于课程问答平台的项目教学探索与实践. [J]. 电气电子教学学报. 2014,36(5):95-97 [↑](#endnote-ref-13)
13. LON-CAPA（The LearningOnline Network with CAPA）是一个课程内容学习和管理系统。它提供了一个基本内容共享池，跨科室和制度的界限导师来集合丰富的课程学习内容 [↑](#footnote-ref-1)
14. LTI（Learning Tools Interoperability）是一个由IMS Global Learning Consortium创建的标准，它的主要目的是使一个标准化的方法通过学习系统连接像LMS这样的学习系统和外部服务工具。 [↑](#footnote-ref-2)