

# 2020年夏季Java小学期大作业

## 实验报告

刘畅 2018011315

任一 2018011423

2020 年 9 月 11 日

实验环境	
minSdkVersion	23
targetSdkVersion	30
Android Studio Version	4.0.1

# 1 代码结构

本项目中，较为重要的文件夹有：java, assets, res。

- java文件夹下的com.java.renyi包为项目的java源码，它主要分为两个部分：首先是db与LDA包，其中的代码实现了后端逻辑；其次根目录下是实现前端逻辑的所有代码。

db包中是数据库的相关代码，包括Repository, ViewModel, Database等各层级的代码，以及知疫学者和各国疫情数据的类。这部分的实现者为任一。

LDA包中，是实现聚类算法的代码。这里的代码主要使用了开源仓库中的实现<https://github.com/jskxsxs360/HanLDA>

根目录下的前端代码主要分为两类：用于实现界面的activity和fragment，用于配置布局显示的adapter, viewbinder和viewfactory。该部分实现者为刘畅。

- assets文件夹下是聚类模型。即在本机上已经训练好的模型，在程序运行时需要加载。
- res文件夹下是前端实现的所有资源文件。其中，子文件夹drawable和mipmap存放图片文件，layout存放各界面的布局文件，values存放尺寸、颜色、字符串等文件，menu存放菜单栏有关文件，xml存放配置文件。该部分实现者为刘畅。

## 具体分工

刘畅：负责前端部分。包括前端页面跳转逻辑设计、前端各页面之间通信、各页面各控件的布局设计、后端数据的展示以及前端少量数据的存储。

任一：负责后端部分。设计数据库并实现数据库逻辑(上下拉和搜索逻辑、离线数据存储)、解析API数据、实现并调优聚类功能、申请微信SDK。

## 2 具体实现

### 2.1 前端部分

#### 2.1.1 总体逻辑

前端总体跳转逻辑如下：点击app后直接进入主页面，主页面通过顶部工具栏实现搜索功能，工具栏下方为标签栏，通过滑动或点击实现标签页切换，点击标签栏最右方按钮跳转至标签编辑页面。

点击工具栏左侧的抽屉按钮，可以打开应用抽屉，其中放置了多个选项，包括疫情数据、疫情图谱、知疫学者和夜间模式。点击可跳转至相应的activity或执行相应功能。

#### 2.1.2 新闻分类与新闻列表

市面上的新闻资讯app主要采用滑动标签页的效果展示分类和对应分类下的新闻列表。实现这种效果的方法很多，我采用的方法是，用tablayout实现顶部标签页，viewpager实现滑动效果，并通过viewpager绑定新闻列表展示的fragment。fragment中就是对应类别的新闻列表了，采用smartfreshlayout嵌套recyclerview能够轻松实现表项内容的定制以及上拉更多与下拉刷新的功能。通过对每一项添加点击事件监听，可以进入对应的新闻详情页，它们之间通过intent通信。

此页面的点击变灰功能配合后端进行实现，在用户点击进入详情页并返回后，前端会执行操作更改数据库对应数据项的viewed字段。而在新闻列表recyclerview的数据绑定中则根据该字段的值设置文字颜色。

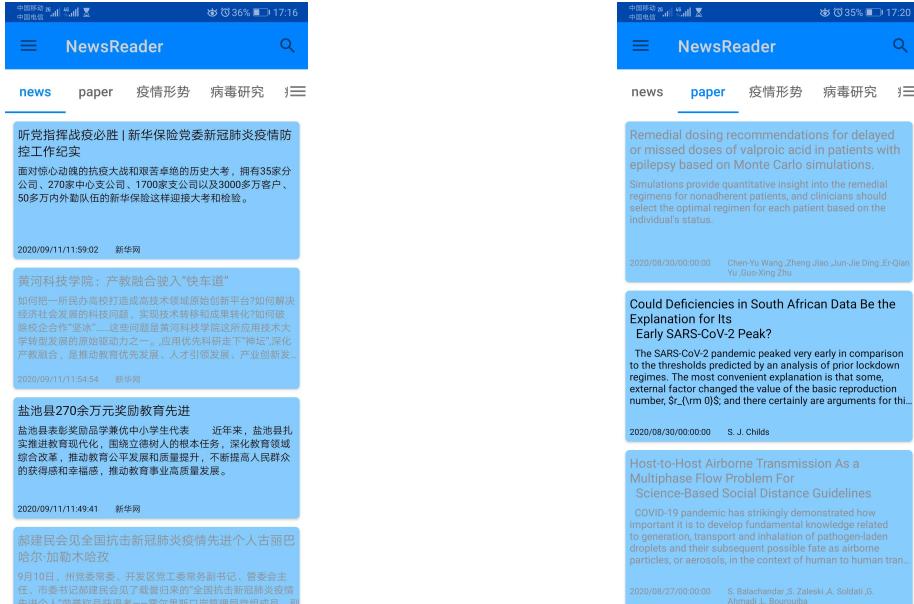


图 1: 新闻分类与新闻列表

**难点：**在实现的过程中这部分出现过3个问题。

- 标签编辑后标签与fragment的内容不对应。比如首次进入应用时第一个标签是"news"，第二个标签为"paper"，删除第一个标签后，显示第一个标签为"paper"，但对应fragment中为"news"的

内容。通过重写fragment的onResume方法，发现当前标签为本应为"paper"的fragment输出的当前tag值竟然是"news"。

这个问题与framgent生命周期管理有关。默认情况下，FragmentPagerAdapter没有在onDestroy方法中添加销毁fragment实例的逻辑，因此自定义的HomePagerAdapter应该继承FragmentStatePagerAdapter。

- 当标签多于3个时，标签页来回切换会出现"IllegalStateException: Bindings already cleared"错误，这是由于viewpager有回收fragment的机制，如果被回收，再次回到该标签页会导致视图bind失败。最简单的解决办法是给viewpager添加缓存机制，这里我根据需要把缓存的fragment数设置为了7。
- 测试时发现点击变灰没有问题，但是某些项也会出现“未点击变灰”。调试输出后发现后端数据正常，但是前段这个玄学问题依然存在。困惑之际队友发现我的NewsListAdapter代码中少了对默认未点击情况的处理，增加之后便一切正常。看来在java代码与xml混用的时候还是要小心。

### 2.1.3 标签编辑

标签编辑页要实现拖拽等动态效果，我找到了一个第三方Layout，类名为：DragGridLayout。该类已经封装好了拖拽、删除等常用功能，根据本处需求对代码和资源文件加以修改即可。

**难点：**值得一提的是如何获得用户的修改结果并返回给主页面呢？第三方Layout本身并没有提供这样的功能。这里我采用了一种原始的方式，通过根节点rootView递归遍历所有的子view/viewGroup，并提取出父节点为GridLayout的TextView中的文字。为了让用户退出程序后依然能够看到标签修改后的结果，每次修改完成返回主页面时都将标签列表保存到了SharedPreferences中，创建页面时则从中读取标签列表。



图 2: 标签编辑界面与编辑成功效果

## 2.1.4 新闻搜索与历史记录

谷歌官方的SearchView控件对这些功能有强大的支持。通过在应用清单中进行配置，可以隐式地向搜索展示页面传递搜索字符串query。搜索展示页面的逻辑与新闻列表页十分类似。而历史记录则继承了官方提供的SearchRecentSuggestionsProvider，可以方便地将近期浏览记录保存到本地。



图 3: 搜索结果页面

## 2.1.5 新闻分享

通过查看微信开发者文档，实现了基于WXWebPageObject的网页分享，能分享到用户好友。消息标题为新闻标题，消息内容为新闻摘要，消息url为对应的url。值得注意的是微信要求摘要文字不能过长。



图 4: 分享结果页面

## 2.1.6 疫情数据展示

疫情数据可视化我使用了第三方库MPAndroidChart，另外由于手机屏幕宽度的限制，全球疫情方面只显示了确诊人数前五的国家，用柱状图显示。使用过程中发现MPAndroidChart的主要问题是没有提供修改显示图表坐标文字颜色的接口。

对于全国疫情，我参考了今日头条，使用了表格的形式进行展示。为了更方便地对表项进行定制，我没有使用TableLayout而是直接使用了RecyclerView。展示效果也令人满意。

在实现过程中，发现RecyclerView始终只展示第一个item，经过仔细比对后发现问题出在被我忽略的item的布局文件上，`layout_height`属性应该取值为`wrap_content`而非`match_parent`。容易理解，`match_parent`会导致第一项直接占用全部的空间，遮挡住后续各项。

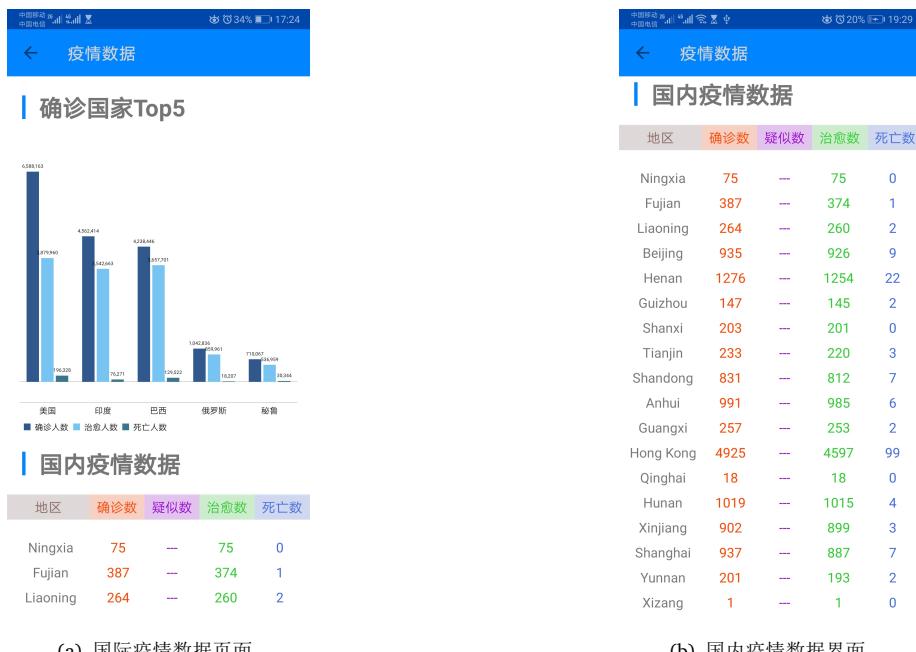


图 5: 疫情数据界面

## 2.1.7 疫情实体搜索

本功能分为搜索和展示两个子功能。

搜索功能依然使用SearchView实现，由于发起搜索和展示在一个页面实现，有关的配置相反更简单了。这里不再赘述。

实体信息展示页面的实现花费了比较多的时间。这里我想实现类似于网页端的折叠与展开效果，最开始我的朴素想法是：RecyclerView的item布局里再嵌套一个RecyclerView？但是经过一番搜索后发现这样实现的性能会出大问题（RecyclerView本身的性能就可能随着项数的增多而降低，如果采取多层RecyclerView嵌套的方式实现，不仅会出现滑动错乱的问题，还可能会导致页面卡顿，影响使用体验），推荐的做法是只使用一层RecyclerView实现展开效果，将不同级的项在代码中“视为”同一年级，只不过通过展开和隐藏的方式响应点击事件。

另外我想到多级折叠展开效果在安卓中其实非常常见，比如我们熟悉的文件管理器。这实质上就是一种树形结构。通过这些关键词的搜索，我找到了一个实现简洁、文档易理解的第三方库：TreeView。这个库的使用人数实在太少，因此网上根本没有对应的教程。我通过查看该库的源码后发现，使用该库需要自定义每层的ViewBinder用于绑定每层布局所需要的数据。但该控件不支持单一ViewBinder绑定多个布局文件，这不符合本处第三级展开时布局不同的要求。在第三级展开中，描述/关系/属性三级对应的布局肯定是不同的，但是又只能包含在同一个布局中。上课时老师介绍过FrameLayout默认情况下会导致重叠效果，我觉得正好能够实现本处需求：通过一个FrameLayout包裹三个不同的布局，并通过java代码设置它们是否可见。

由于RecyclerView计算高度并移动item相对耗时，而返回的实体信息中某些热门实体如“药物”的关系非常之多，而大部分实体的关系比较少，因此前端截取了前20条（如果大于20条）关系进行展示，以免造成应用的性能问题。



图 6: 搜索结果页面

## 2.1.8 新闻聚类

新闻聚类页面合并到了主页面进行展示。主页面将聚类关键词展示到标签栏中，当切换到对应标签时显示对应关键词下聚类的结果。



图 7: 聚类展示页面

## 2.1.9 知疫学者展示

知疫学者页面的实现逻辑和新闻列表页面类似。同样是通过RecyclerView实现了学者列表的展示、点击进入对应的学者详情页。学者详情页展示了学者的姓名、照片、职位、简介、教育经历、工作经历等。

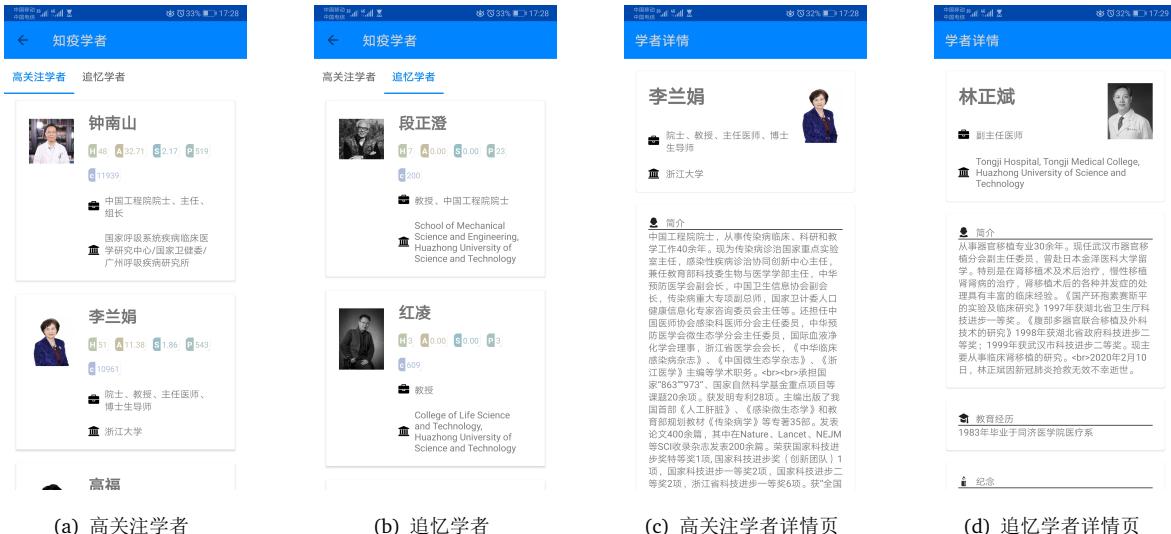


图 8: 知疫学者页面

## 2.1.10 夜间模式

夜间模式并非本次作业的硬性要求，我在这里进行了一些简单的学习和尝试。对于非官方控件需要自行配置values文件夹下的color(night)文件，才能够实现夜间模式时的颜色变化。

通过AppCompatDelegate.setDefaultNightMode函数可以实现基础的夜间模式（会自动对不同色彩进行匹配）。



图 9: 夜间模式

## 2.2 后端具体实现

后端部分由任一完成，下面是后端的具体实现介绍。

### 2.2.1 后端框架

在对比Room, Realm等数据库之后，我们使用了Android官方推荐的Room作为后端数据库，原因在于Google在Room数据库中，实现了很多针对Android APP开发的特性，这些特性可以简化APP的开发，同时保证数据库的高效。

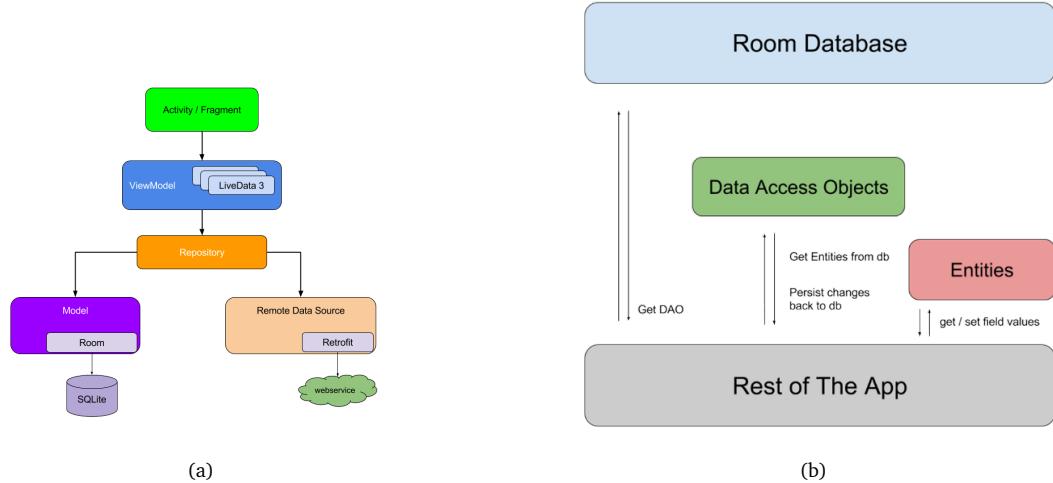


图 10: Room数据库框架示意图

上图展示了Room数据库的总体结构。<sup>1</sup> 在图10(a)中，我们可以看到Room数据库大体分为前端、ViewModel, Repository这几层。前端的Activity和Fragment只需要调用ViewModel层的接口，即可通知后端对数据进行更新或从后端获取更多内容。Repository层负责对数据进行访问，它可以通过图10(b)中的Data Access Objects(简称DAO) 对数据库进行访问，同时也可选择从网络上请求更多数据加入数据库并显示。这样的结构的优势在于，能够很大程度上实现前后端的解耦合。前端只调用ViewModel的接口即可获取数据，后端复杂的逻辑全部在Repository中进行处理，降低了前后端对接的难度。

此外，这个框架中还有一个亮点，即LiveData.<sup>2</sup> LiveData很大的优点在于，能够方便地保持前端数据与后端数据的同步。例如，前端需要后端刷新新闻时，在子线程后端数据库的确加入了最新的新闻，但前端如何知道后端数据已经更新了呢？LiveData的作用在这里就是，在数据库层面随时监控后端的数据变化，当后端数据出现变化时，可以立刻通知前端刷新页面，这样就保持了前后端数据的同步性。同时，LiveData在底层也处理了前端各组件生命周期、内存泄露等问题，大大方便了开发。

<sup>1</sup> 图源<https://developer.android.com/jetpack/guide>, <https://developer.android.google.cn/training/data-storage/room>

<sup>2</sup> 关于LiveData的更多信息<https://developer.android.com/topic/libraries/architecture/livedata>

## 2.2.2 数据库逻辑

数据库主要储存了News, Paper和Events类的词条。这部分主要介绍对News和Paper词条的处理。



图 11: News,Paper以及搜索后端处理逻辑示意图

应用刚启动时，数据库会向API请求一个PAGE\_SIZE的news和paper数据<sup>3</sup> 加入到本地的数据库中。这样应用刚启动时，数据库中就有PAGE\_SIZE数量的news和paper了。在此之后，数据库将ADD\_SIZE条<sup>4</sup>的news和paper设置到NewsList和PaperList当中。由于NewsList和PaperList使用了 LiveData的机制，这两个List有了新的内容后，立即通知前端刷新界面，也就是说前端与LiveData就是完全同步的。

当前端上拉想要加载更多时，会告知后端。后端则会检查当前数据库中的新闻或论文是否足够给前端更多，如果仍有前端未显示的新闻，则把尚未显示的部分加入NewsList或PaperList的LiveData中，LiveData负责与前端保持同步。若当前数据库中的新闻不够给前端更多了，则数据库会向API请求更多的新闻或论文，加入数据库并设置LiveData，LiveData负责与前端保持同步。

当前端想要下拉刷新最新新闻时，同样会告知后端，后端则会直接向API请求最新的新闻加入数据库，并设置LiveData，LiveData负责与前端保持同步。

当前端设置想要搜索时，只需传给后端搜索关键词，后端调用SELECT语句以及LIKE子句，即可在数据库所有的news, paper和events中匹配标题和内容，将含有相应关键词的词条设置到LiveData中，LiveData保证前端与后端返回的搜索结果同步。

当前端点击一条新闻时，会告知后端将其设置为已读状态，后端通过UPDATE语句即可设置该词条的已读状态，同样通过LiveData的机制保证前端得到的该词条被标记为已读，以便将其显示为灰色。

这样的后端设计的优点在于：

1. 简化了前端逻辑，有利于前后端解耦合。后端负责所有上拉加载更多、下拉刷新逻辑，前端只需要向后端发送请求即可。
2. 加入LiveData作为前端与数据库中间的缓存，可实现本地新闻大量存储。如果没有LiveData作为中间的缓存，我们可能需要将数据库中的所有新闻都发给前端，这样既不必要，也会增加前端处理的难度。或者说前端每请求一次数据，后端都要到API去请求数据，这样会导致频繁的网络连接，降低APP响应速度。有了LiveData作为中间的缓存，我们的逻辑和实现都更为自然和简洁，APP的响应也更加快速。同时数据库可以储存大量的新闻在本地，方便离线浏览以及搜索。

<sup>3</sup>PAGE\_SIZE在APP中设置为100

<sup>4</sup>ADD\_SIZE在APP中设置为10.

### 2.2.3 聚类逻辑

在本APP中，我们实现了对Event类词条的在线聚类，即在APP中实时运行聚类并得到结果。聚类用到的模型是隐含狄利克雷分布(Latent Dirichlet allocation, 简称LDA).<sup>5</sup>

该算法的主要过程分为两个部分，训练和预测。在训练过程中，这个算法可以预先指定聚类的类个数k，即训练出的模型可以从训练集中学到k个类。训练集是由大量已经做好分词的文章组成的，这里对文本使用的模型是词袋模型(Bag of Words)。在训练时，该算法通过吉布斯采样的方法(或EM算法)，对LDA模型的参数进行估计。训练的结果是得到k个类，每个类有20个关键词，每个关键词都有与该类关联程度的系数。通过每个类的关键词，即可判断出该类的主题。在预测过程中，我们会计算每篇文档与模型中k个类的相似程度，选择相似程度最大的类，作为当前文档的类即可。

在我的实现中，我首先在电脑上下载了所有699条event，并对其进行分词和本地的训练。经过我对聚类类数目的尝试，以及聚类结果的观察，设置聚类数目为5是一个合适的选择。聚类数目太多，某些类就会太小或者与关键词的关联系数不够大，而聚类数目太少又会降低聚类的意义。

topic0 :	topic1 :	topic2 :
研究=0.038029001049676985	新冠=0.01432087000425283	病毒=0.06361322052934702
病毒=0.031163638624813833	疫情=0.01357684535161157	冠状=0.022957303110138366
sar=0.019486435870929308	卫生=0.01048550218183694	冠状病毒=0.022509821358245922
感染=0.018434663483404314	美国=0.010243216475924705	新型=0.012814154582898261
cov=0.018071246009642807	传播=0.008055731803450284	大学=0.011488219048035939
发现=0.01761979608860049	病例=0.007814722981984433	团队=0.009721087324277143
人员=0.01521775927512594051	感染=0.007795927512594051	蛋白=0.009569490264534223
新冠=0.01518802813143326	中心=0.007443695249143481	研究=0.008314034062760605
细胞=0.011876932607069447	流行=0.007376908102778518	基因=0.007614365621886081
大学=0.009567473800620709	肺炎=0.006145006369860555	药物=0.006932392316471321
topic3 :	topic4 :	
疫苗=0.02069659719245641	患者=0.029580454767273747	
临床=0.017204200189676538	研究=0.024493715324988775	
中国=0.01463949878578981	19=0.019858784533148568	
试验=0.01349956711002891	covid=0.019647199219058088	
新冠=0.011011501451041007	肺炎=0.014810921970394818	
临床试验=0.009992812701854583	新冠=0.013050126272837848	
院士=0.009505230469532971	治疗=0.010611406394336245	
研发=0.007366756006403241	发现=0.009839885497686825	
科学=0.007230969541230677	医院=0.00811425482427015	
肺炎=0.00687390300413448	大学=0.008097300895007927	

图 12: 聚类训练得到各topic及topic关键词

上图为我的聚类训练得到的topic以及topic的关键词，每个关键词后面的数字都是该关键词与这个topic的关联系数，关联系数越大代表与topic联系越紧密。通过对这些topic关键词的观察，我把topic0和topic2命名为"病毒研究"类，topic1命名为"疫情形势"类，topic3命名为"疫苗药物"类。在预测时，对于一条event，我们选择其相关系数最大的topic所对应的类，作为这个event的聚类结果。

这样的聚类方法，实现了不错的结果。下表是这4类所对应的部分新闻，可以看出聚类效果很好。

聚类这部分实现的难点有以下两点：

1. 弄懂LDA的聚类原理。<sup>6</sup> LDA的数学原理较为复杂，需要详细的学习才能使用。
2. 与开源代码做对接。我参考的开源LDA实现<sup>7</sup>，是通过绝对路径访问文件中的内容作为输入进

<sup>5</sup>感谢刘雪怡同学向我推荐了这个算法及该算法在Github上的开源代码，我使用的该算法开源代码链接为<https://github.com/jsksxs360/HanLDA>

<sup>6</sup>学习时主要参考了[https://en.wikipedia.org/wiki/Latent\\_Dirichlet\\_allocation](https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation)

<sup>7</sup>参考了<https://github.com/jsksxs360/HanLDA>

表 1: 聚类结果分析

	内容
病毒研究	杜兰大学医学院罗伯特 加里教授等研究发现新型病毒不是在实验室中构建的，也不是有目的性的人为操控的病毒，而是自然产生的。
	deCODE基因公司通过对冰岛确诊病例进行基因测序分析，发现40个新冠病毒的变体。病毒学家认为新冠病毒最终可能会进化为更具传染性但低致病性的病毒
疫苗药物	上海市科委主任张全称在疫苗研发方面，多路线部署推进的疫苗研发均取得良好进展。 mRNA疫苗，已启动灵长类动物毒理和药效预实验，预计4月中旬临床试验
	以色列科技部27日宣布，该国研究人员正在加快开发一种口服的新型冠状病毒疫苗，研究人员希望能在8到10周内制造出疫苗，有望在90天内开始临床试验。
疫情形势	国家卫健委新闻发言人米锋介绍，目前国内疫情防控形势持续向好，但是境外疫情却在加速扩散，“外防输入”已经成为当前我国疫情防控的重中之重。
	首都机场设置处置专区，全部国际及来自港澳台地区进港航班，均停靠首都机场处置专区，3月16日起所有境外进京人员均需转送至集中观察点隔离观察14天
患者治疗	托珠单抗（Tocilizumab）对重症COVID-19患者临床验证有效果，其机制是阻断IL-6抑制炎症因子风暴，由于病人数量有限，后续仍需大量研究。
	首都医科大学附属北京同仁医院发表文章指出高龄和男性是COVID-19预后较差的危险因素，男性更倾向于具有更高的疾病程度和死亡率。

行预测的，此外该开源代码也只能从绝对路径加载训练好的模型，无法直接移植到Android APP中。因此，我对这份开源代码也进行了一定的学习，修改了文件的推断函数使之可以将字符串作为输入，并且学习在Android APP中加载本地文件，同时修改开源代码的构造函数使之成功适配Android APP.<sup>8</sup>

#### 2.2.4 其他逻辑

其他后端的需求主要有疫情数据、疫情图谱和知疫学者的逻辑。这部分的处理逻辑与news和paper的逻辑相仿。但由于疫情数据、疫情图谱、知疫学者具有较强的时效性和灵活性，不便像news和paper一样，做本地的大量存储。因此每次调用这些功能，都会向API发送请求，API返回数据后设置LiveData，LiveData保证前端同步显示。因此这部分在显示时，可能会先空白1-2s，然后再显示，这是后端在向API请求数据并解析数据的时间。

<sup>8</sup> 此处参考了<https://stackoverflow.com/questions/34521349/asset-file-directory-giving-java-io-filenotfoundexception-eclipse>的第一个回答。

### 3 总结与心得

#### 3.1 前端部分总结与心得-刘畅

本次大作业中我负责前端部分，最大的收获是java编程能力和前端开发能力得到了一定的提高。另外，快速获得所需要的学习资料、快速习得其中的要领、快速定位问题出现的位置是每一个开发者必备的能力。我认为本次开发可以起到锻炼这些能力的效果。另外还要特别感谢后端队友精妙的框架设计，让我能够专注地进行前端开发，解决了前端开发的后顾之忧。队友的出色和靠谱表现让我印象非常深刻。

#### 3.2 后端部分总结与心得-任一

在本次项目中，我负责了后端的设计，主要内容是设计数据库的组织结构、设计后端与前端的交互方式。得益于LiveData的功能，我们基本没有在前后端数据同步上花费过多精力，得益于Room数据库官方推荐的组织结构，整个项目的前后端分离也做的非常完善。<sup>9</sup> 总体来说开发体验非常好！虽然说由于这是我第一次设计后端，在一开始的结构选择上也有过犹豫和反复，但最后还是选择了FrontEnd-LiveData-Database这样的逻辑来展示news和paper，这样的结构有着很好的性能。

整个项目的进度我们也把握得很合理。在小学期第4周刚开始，我们在正常小学期课业与考试的压力中，从9月3日-9月6日进行了4天高强度的开发，每天工作时间在6-8小时，这段时间我们就基本完成了所有的基础功能和一部分扩展功能。9月10日中午，我们就全面完成了基础功能和扩展功能，在当天下午进行了充分的测试和调优，保证了APP的稳定性。9月11日提交当天，我们录制了展示视频、撰写实验报告，整个过程很艰辛，但进度非常稳健。

在这里我还想特别感谢我的队友刘畅。她是一位非常优秀、努力、认真并且可靠的合作伙伴。她在小学期开始第一周就完成了简要的前端框架，在之后的时间里一直对前端进行调优和增加新功能，她对这个项目的投入让我十分敬佩。此外也感谢老师和助教对我们的悉心指导和帮助。最后也感谢努力的自己，设计了一个不错的后端，让整个APP的效率和鲁棒性都很好，也希望我自己之后继续努力，进一步增强自己的工程能力。

---

<sup>9</sup> 参考了Room的组织架构教程<https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0>