2020 夏季 Java 小学期大作业
实验报告

罗弈桢 2018013381
周知谦 2018013380

2020 年 9 月 11 日

1 代码结构

1.1 java 部分

前端代码（由罗弈桢实现）：

- MainActivity：启动 app，显示封面图片并在三秒后进入主页。
- HomePage：交互主页，提供了新闻显示、搜索、分类切换等功能。
- CategoryActivity：用于设置顶部功能与分类列表。
- VisualizeActivity：省市与全球疫情数据的可视化。
- ClusterActivity：显示聚类结果。
- ScholarActivity：显示学者信息列表。
- ItemNewsActivity：显示一条新闻的正文内容。
- ItemScholarActivity：显示学者的详细信息。

后端代码（由周知谦实现）：

- News: 一条新闻，包含了标题、url、日期、来源等信息。
- NewsList: 新闻列表，支持从接口中获取更多新闻、将新闻浏览记录进行本地存储。
- Entity：实体，包含了名称、关系等信息。
- Expert：学者，包含了姓名、职位、主页 url 等信息。
- CovidData：疫情数据，包含了各省和各国家的确诊人数。
- Server: 服务端，支持存储历史记录、获取新闻、进行关键词和实体搜索、获取学者和疫情数据

1.2 python 部分

聚类（由罗弈桢实现）：

- cluster.py：通过预处理得到的新闻标题和相关新闻信息进行聚类
- topic.py：针对每一类寻找关键词

2 具体实现

2.1 前端具体实现

图 1: 初始页面

图 2: 主界面

图 3: 设置分类列表

2.1.1 初始界面

进入初始界面时，设置一个三秒的计时器，计时器结束后跳转至主界面。

2.1.2 主界面

主界面主要由搜索栏、分类列表和新闻列表构成。搜索栏使用的是 androidx 的 SearchView，通过监听 OnQueryTextListener 事件来向后端发送搜索请求。分类列表采用了横向的 LinearLayout，每次加载时动态地向其中填入 TextView，并使用 HorizontalScrollView 实现横向滚动。

新闻列表最开始采用了 ListView+SimpleAdapter 的实现方式，但这种方法并不能方便地实现读过的新闻置为灰色的操作，因此还是改用 LinearLayout 的动态填入方式。为了监听下拉事件及显示动画，使用了 SwipeRefreshLayout 作为父控件。上拉事件则采用了 ScrollView 的 onScrollChange 方法，当滚动的 Y 坐标 +scrollview 的高度与新闻列表的高度相等时即滚动到底端，此时向服务端发送加载更多的请求。

2.1.3 设置分类列表界面

左右两个列表分别显示了已选择和未选择的分类，点击分类时会出发修改动画。动画采用了 ObjectAnimator，该类能够方便地实现平移、淡入淡出等效果。

2.1.4 可视化界面

使用 MPAndroidChart 中的柱状图来显示各个省市和国家的数据，点击两个按钮能在全国/全球数据之间切换。对于 x 轴的标签，使用 ValueFormatter 来将数值转化为字符串。由于国家和地区的名称长度较大，即使对 x 轴采用了 RotationAngle 也不能完全显示，因此改用了 HorizontalBarChart 来进行展示。

图 4: 可视化

图 5: 新闻正文

图 6: 等待中

2.1.5 新闻正文界面

使用了 webView 来显示新闻的具体内容。一开始我使用了 loadDataWithBaseUrl 来显示获取到的 Html 信息，然而该方法无法正确显示微信公众号的图片，因此改用了 loadUrl 函数。

对于新闻缓存，采用了 webView 自带的 cache 功能，当处于离线模式时，将 CacheMode 改为 LOAD_CACHE_ELSE_NETWORK 即可。

2.1.6 异步加载与等待动画

由于部分网络资源加载时间较长，因此需要采用异步的方法获取数据，防止主线程被阻塞。在实现中，我采用了 Thread + Message + Handler 的方法，即当网络请求发生时，创建一个新线程获取数据。由于只有主线程能对 UI 元素进行修改，因此当线程取得数据后，需要新建一条 message 通知主线程。而主线程则通过 handler 的 handleMessage 方法读取通知，并对 UI 元素进行修改。

在加载新闻列表和新闻正文内容时，使用了 ZLoadingView 来显示等待动画。当页面加载完成时，同样使用 Message 和 Handler，将该元素设置为 Invisible。

2.1.7 微博与微信分享

使用了微博和微信的 SDK 进行分享，但未通过审核，无法对该功能进行调试。

图 7: 申请截图

2.2 后端具体实现

2.2.1 新闻列表获取

新闻包括最新推荐列表和分类列表。最新推荐列表即不加过滤地获取所有新闻：在启动的子线程中，使用给定的分页新闻列表 API 获取新闻列表，解析 JSON 得到新闻的标题、ID 等。对于每一个新闻，同时启动子线程，使用事件详细信息的 API 获取事件具体的来源、URL、相关新闻等信息。对于具有特定分类的列表（如国内新闻、前沿相关新闻、国际新闻）使用聚类算法得到的结果对新闻列表进行过滤。同时对于每一个新闻列表，在本地实时保存到文件，以供离线查看。

2.2.2 历史记录存储

每一个新闻点击查看后自动调用该新闻条目的 view 方法，将该新闻记为已读并保存至本地数据。使用一个列表维护当前已查看的所有新闻的列表，当查看历史记录时，使用该列表构造一个 Newslist 用来显示历史列表。显示其它新闻列表时，需要根据历史已读记录判断每条新闻是否已读，以使得所有列表在重新加载后也能正确显示新闻的已读状态。

2.2.3 新闻搜索

预先利用所有新闻列表的 API 将所有新闻的标题保存到本地。搜索使将关键词与新闻标题匹配，对于匹配结果启动子线程，使用事件详细信息的 API 获取事件具体的来源、URL、相关新闻等信息，得到搜索结果。

2.2.4 疫情数据获取

请求所有地区的汇总疫情数据，根据地区名的字符串格式判断其是否是国家、身份，以及身份所属的国家。根据每天的累计数据，可以计算出累计感染人数、现存感染人数等数据。

2.2.5 学者信息获取

使用给定的疫情相关学者 API 获取学者列表，解析 JSON 得到学者的姓名、职位、单位、头像链接、主页链接等信息。注意其中部分信息可能缺少，需要相应判断并给出默认值。部分外国学者没有中文名，应优先显示中文名，没有中文名时显示英文名。

2.2.6 实体搜索

搜索时同时给出相关新闻条目和相应实体的图谱信息（如果可用）。调用实体详细信息 API 获取一个列表，其中每个实体的信息包括百科解释、性质、图片、与其他实体的关系等。选取相关度较高的实体显示其图谱。

2.3 聚类算法实现

聚类算法需要满足语义相近的新闻被聚到一类，有相关联系的新闻被聚到一类，因此考虑采用 embedding + graph smoothing 的方法。

对获取到的 38240 条新闻进行了聚类。首先使用了 Bert Server 和 Client，通过预训练的词向量得到新闻

标题的 768 维向量表示，并通过连续 64 维取均值的方法将其压缩至 12 维。接着将每条新闻看作点，将新闻的相关性看作是边，相关性的大小即为边的权重，可以得到一张图，利用该图可以对新闻向量进行平滑化：

$$a_u^{(k+1)} = \lambda a_u^{(k)} + \frac{1-\lambda}{deg(u)} \sum_{v \in \mathcal{N}(u)} a_v^{(k)} w_{uv}$$

其中 deg(u) 为 u 的点权（与其相连的边权和），λ 为超参数，实现中设置为 0.4。

进行 k（实现中 k=3）轮平滑后，对得到的向量执行 KMeans 算法得到聚类结果，并使用 LDA 算法得到每个类对应的关键词。

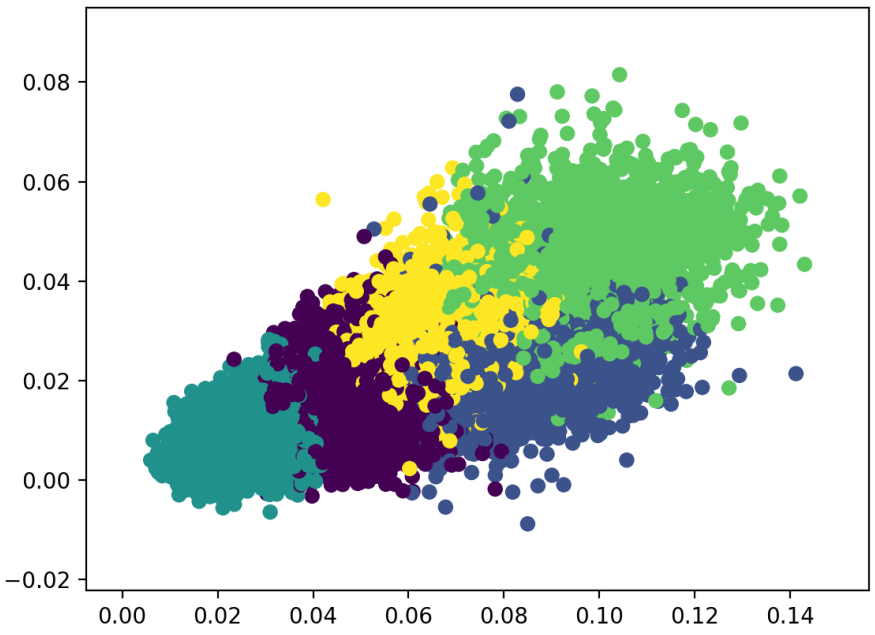

图 8: 聚类结果

3 总结与心得

3.1 前端：罗弈桢

本次大作业我主要完成前端的工作。前端的界面设计和事件响应的实现难度相比于后端并不是很高，但是想要做到完善还是有很大的难度。首先是在设计 XML 布局文件的时候需要掌握各种 Layout 的使用方法，而其中的布局特性十分丰富，而且经常会出现使用不当的情况，需要多次调整。例如使用 ListView 显示列表时，我将 height 设置为 wrap-content，有时列表的高度会充满整个屏幕，有时则只有一行文字的大小。此外，在许多功能和特效（如等待动画、下拉刷新）的实现中需要用到其他库和工具，这时就非常考验程序员的文档阅读能力和调试能力。

此外，页面之间的事件响应、逻辑转移也是一项重要的工作。尤其是异步加载的功能，一开始我将数据拉取写在了主线程中，导致加载新闻列表时 APP 直接锁死，极大地降低了用户体验。在查阅了大量资料后发现

可以使用 handler 来实现异步的加载和子线程向主线程的通讯。遗憾的是，在功能逐渐叠加时我发现使用一个 Activity 内嵌套多个 Fragment 的写法更优，但由于时间原因已经无法重构了。

最后，前端与后端的协作也是十分重要的，在工程设计的阶段就需要定义一些合适的类和接口进行前后端的交互。将前后端解耦合，也能够极大地提升开发和调试的效率。

本次大作业加入了新冠疫情的主题，但由于新闻的分类不是很明确，导致和往年的修改分类列表的需求有一些冲突，开发的时候较为混乱。

3.2 后端：周知谦

后端涉及大量网络操作和外存读写，必须要注意的是性能问题，耗时的 IO 必须放在子线程中，并且创建后不能立刻 join，否则会导致主线程卡住。此外，多个不具有依赖关系的网络请求可以全部发出后再逐个处理返回，这样有利于大幅减少响应时间。此外多线程可能导致竞争的问题，例如多个线程同时修改同一数据时容易写出错误数据甚至直接导致程序崩溃。因此要注意对所有可能被多个线程同时读写的数据加锁。

4 参考资料

- [1]https://github.com/PhilJay/MPAndroidChart
- [2]https://github.com/omadahealth/SwipyRefreshLayout
- [3]https://github.com/zyao89/ZLoading
- [4]https://github.com/hanxiao/bert-as-service
- [5]Arthur D , Vassilvitskii S . K-Means++: The Advantages of Careful Seeding[C]// Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana, 2007. ACM, 2007.
- [6]Blei D M , Ng A Y , Jordan M I , et al. Latent Dirichlet Allocation[J]. J. Mach. Learn. Res, 2012, 3:993-1022.