MorphlingNews（新冠肺炎疫情 APP）

2018011351 计83 何雨泽 2018011114 计83 陈祖浩

1. 基础功能

功能	子功能	总占比(%)(功能+性能)	是否实现
系统支持	要保证程序在安卓机上正常运行，测试过程中程序不崩溃。	8+2	√
页面布局	布局合理，点击处理正确	8+2	√
新闻列表	删除和添加操作	4+1	√
	正确显示新闻列表的消息，布局和展示，点击进入新闻详情页面正确	8+2	√
	实现新闻的本地存储，看过的新闻列表 在离线的情况下也可以浏览。新闻是否看过的页面灰色标记。	8+2	√
	上拉获取更多新闻，下拉刷新最新新闻	4+1	√
	显示新闻的来源和时间	4+1	√
	新闻关键词搜索，历史记录	4+1	√
分享收藏	使用微博等 SDK分享，新闻详情页面点击分享可以分享到常用的 app，分享内容带有新闻摘要	4+1	√
疫情数据	全国各省疫情统计，全球各国疫情统计可视化，以表格或折线图或柱状图形式呈现	8+2	√
疫情图谱	对新冠疫情图谱内容进行查询和展示，需展示实体词条的描述，包含的关系和属性	8+2	√

2. 扩展功能

功能	子功能	总占比(%)（功能+性能）	是否实现
疫情新闻聚类	对疫情相关的新闻事件进行聚类、展示关键词和聚类新闻	8+2	√
知疫学者	显示在疫情领域的高关注学者和追忆学者，点开可以查看	4+1	√

3. 前端简介

3.0 整体架构

主界面 MainActivity 共包含 4 个 Fragments，分别为实时新闻、疫情数据、知识图谱、知疫学者，可以通过底部的导航栏切换这 4 个 Fragments。

3.1 实时新闻部分

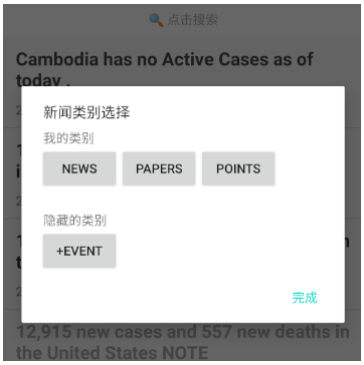
新闻界面展示如下：

主页使用了 ListView 展示了当前分类下的所有新闻，上方使用一个 TabLayout 进行不同分类新闻的切换，每次切换时更新 ListView 中的数据。特别的，当切至 Event 栏目时，会出现第二行 TabLayout，并展示当前聚类结果和关键词。

点击新闻后进入详情页（打开一个新的 Activity），返回后可以看到已看过的新闻条目变灰。离线环境下也可以看到之前加载好的新闻列表。

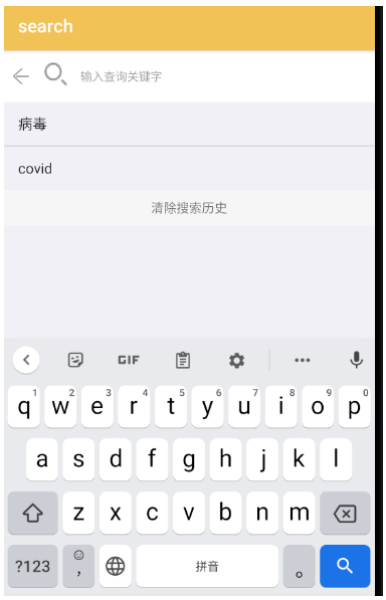
点击右上角的加号进入分类的添加与删除，当展示分类只剩 1 个时不允许删除，增删时有动画特效。

界面如下：

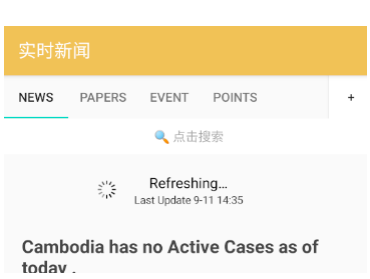


在搜索功能上我使用 SearchView 组件。点击搜索栏后进入搜索页面，可以看到历史记录功能，搜索后（进入新 activity）页面无误，点击后进入新闻详情。

我也申请了微信 SDK 并进行了微信相关 Activity 的设置，点击右下角的微信小图标可以将新闻内容分享到微信：



同时使用了 refreshLayout 实现了上拉下拉刷新功能，可以看到也有动态特效：



3.2 疫情数据

上部一个 TabLayout 用于切换国内和全球疫情，中间一个 GridLayout 用于显示总的数据（阴影 panel 使用 png9 实现），下部表格使用 ListView 实现。

点击右侧详情，可以进入该省份/国家的详细数据界面，其中包含一个图表，红色、绿色、黑色曲线分别代表了累计确诊、累计治愈、累计死亡人数的趋势：

3.3 知识图谱

进入知识图谱后，能看到这样的一个搜索框，搜索后即可查看相关实体的结果：

点击条目的标题，可以看到详情页面如下，火的个数代表词条的热度。

关系条目通常会非常多，为了避免这种情况，我设计了展开和收缩的按钮。可以看到图片、描述、关系、属性都展示无误。点击右侧的搜索图标，可以跳转到其他条目的搜索页面：

3.4 知疫学者

界面如下所示，点击条目后可以看到人物的详细信息，再次点击后隐藏。

顶部 TabLayout 用于切换两种不同类型的学者。当切换至 “追忆学者” 栏目时，可以看到条目中多了【逝世日期】等额外的信息：

4. 后端介绍

4.0 整体架构

从今年大作业的需求来看，APP仅仅通过HTTP请求就可以完成所有的网络交互，也没有大规模的数据需要处理（整个新闻列表文件大概只有20M左右），因此不需要特地实现服务端，所以后端只需要在本地对数据进行处理即可，当然多线程是有必要的，在APP中也有被大量运用。

本地端中，我选择以一个单例**InfoManager类**作为中枢，为前端提供必要的数据接口，同时也实现了各种基本的数据处理操作。对于一些复杂的数据处理，则放在其他不暴露给前端的类中实现，例如**Kmeans类**主要实现聚类算法，**NewsDictionary类**保存了关键词的倒排索引和出现在文章的频率，为新闻搜索接口提供支持。

4.1 疫情数据 & 新闻数据

4.1.1 数据拉取

疫情数据和新闻数据都可以通过GET请求由给定的接口直接获得，由于请求得到的均为JSON字符串，因此我们需要对其进行解析。这里我们选择了Fastjson这一Java库来进行JSON解析，主要是用到了JSONObject和JSONArray两个类分别来处理**字典**和**列表**，通过反复迭代来爬取需要的信息，其中新闻类信息被封装在了**NewsData类**，而疫情数据信息被封装在了**InfectData类**中。**InfoManager类**对前端提供了getNewsData和getInfectData两个接口用于获取数据，还提供了

4.1.2 离线存储

我们认为新闻数据和疫情数据是APP中最重要的数据，因此有必要让用户在离线情况下也能访问，所以实现了这两类数据的离线存储（当然作业需求中也要求能离线访问新闻列表），这里我们选择在本地存储JSON串的方式来进行离线存储

当用户打开APP时，无论联网与否，都会优先读取本地的JSON文件，然后系统才会在后台更新这两类信息的最新数据，同时将更新的数据保存至本地——这样保证了用户在使用过程中不会在开启程序界面等待太久，提高了用户体验。

4.1.3 新闻搜索

我们实现了对新闻标题的排序搜索功能，使用了Java版的Jieba(https://github.com/huaban/jieba-analysis) 对标题和搜索关键词进行分词，根据搜索关键词的倒排索引定位到若干篇文章，再计算每种关键词对于每篇相关文章的TF-IDF值，加权求和后按权值由大到小排序并展示。

上述倒排索引以及词频字典都存储在**NewsDictionary类**中，这些数据会在新闻数据刷新后在后台刷新（因此搜索功能在最新新闻方面相比新闻列表呈现的有一定时间滞后）

4.2 知识图谱 & 知疫专家

这两类数据的拉取方式和疫情信息和新闻信息完全一致，只不过内容更多更复杂一些。前端可以通过**InfoManager类**的getKnowledge和getScholarData两个接口获得数据。这两类数据并不会被离线保存在本地，而是实时访问和实时加载的，因此在离线情况下无法正确被加载。

4.3 新闻聚类

按照要求APP实现了对event类型的新闻的聚类。聚类使用的是K-Means算法（位于**KMeans类**），时间复杂度为 $O(lnkm)$ ，其中l为迭代次数，n为样本个数，m为每个元素的字段个数，k为类别数。在聚类之前我们需要构造文本特征向量——每一篇event新闻都有一个特征向量，我使用的是**词袋模型 + TF-IDF概率统计模型**，先统计所有event新闻出现过的词（不重复）作为向量基底（这里我没有用之前做搜索功能使用的**Jieba**分词，而是直接使用了接口JSON文件中的**seg_text**属性里提供的分词），然后每个词关于这篇新闻的TF-IDF值就是向量这一维的值。这样处理的好处是能比较全面的反映出文章的整体特征，坏处就是特征向量的长度太大，导致聚类计算的效率大大降低。

聚类的主体算法是先随机选取k个样本点作为初始的质心点，然后计算每个样本点到各个质心点的欧氏距离，选择距离最短的质心点归为一类，于是就把样本分为了K簇。然后对于每一簇内所有样本点取**向量平均值**，作为新的质心点，然后再重新计算距离、归类...如此反复迭代，直至连续两次分类完全一致后。经实验，在当前的模型下反复迭代3~4轮即可收敛。

本APP中一共对新闻分了6类，而这6类关键词的选择是通过统计同一类的新闻所有出现词的词频得到的，我手动对出现的高频词进行了筛选，去掉了所有类共性较高的关键词，保留了特征度较高的关键词。

4.4 微信分享

微信分享功能的实现是直接使用了微信开放平台官方提供的SDK，创建了一个**WXShare类**，其中调用了微信API的注册接口，需要注意的是要正确填写微信开放平台审核通过提供的APP_ID，并且要把apk的签名正确填写在微信开放平台上，否者分享功能会无法使用。

这部分代码基本上都是照着文档和样例实现的，就不多做描述了。

4.5 一些优化

4.5.1 并行优化

由于后端需要对APP的流畅程度负责，因此我对程序做了一些优化来保证用户的使用体验。第一点优化在于多线程的使用，由于新闻信息和疫情信息更加需要在第一时间呈现给用户，因此我把它们放在了初始化的主后台线程中，在打开APP的同时就立即开始尝试读取本地JSON文件。而在新闻被加载完全之前，搜索功能（需要先构造各种词典）的加载和更新无法被加载，于是我就让APP在加载新闻的同时先加载Jieba分词（这个包启动也需要若干秒时间），等到新闻被完全加载/更新后在加载/更新。整个搜索功能是位于另一个次优先级的线程上完成的。

这样的并行协同加载，经测试的确比之前单后台线程的串行效率要高很多，用户体验也要好很多。

4.5.2 聚类优化

由于观察到接口提供的event新闻数量是确定且不会再更新的，于是在最终版本的APP中我们选择不对新闻进行在线聚类，而是把聚类的结果硬编码到APP本身，实现读取event类新闻后无需在线聚类只要查询分类参数即可立即显示分类结果。

5. 总结与心得

后端（陈祖浩）：

这次大作业的经历让我学到了很多，除了对于Java语言本身和后端开发更加熟悉外，也让我体会到了Android设备编程和以往Windows/发行版Linux的不同，遇到了很多前所未见还不好调试的bug，也遇到了很多性能的挑战，所幸这些困难最终都克服了，这也让我感到很欣慰。

除了Debug之外，在实现层面令我印象最深也花费最多时间的是搜索功能和聚类功能的实现，其中搜索功能因为有去年小学期的实现经验，所以相对还比较顺利，主要花费的时间在提高性能上；而聚类功能则是我之前完全没有实现过甚至没有接触过的领域，在算法实现层面我先花了很多的时间上网找资料学习，完全弄明白后又花了很多时间按照我们的APP的需求从头实现一遍，当然虽然困难，聚类无疑也是整个大作业过程中让我感觉自己知识和能力提高最多的需求。

总而言之，虽然这次大作业也是在百忙之中和队友赶出来的产品，还有很多细节没有时间再实现或者优化，有些遗憾，但整体来说，起码也算是写出点有模有样的东西来了，相信如果以后再有机会开发类似的产品，我们能有把握做的更好更完善。最后还是要感谢一下我的队友何雨泽，我们的分工一开始就很明确，因为我自己前端基础薄弱，也知道他比较擅长，事实也证明他负责的前端部分确实无需担心，让我可以在对AS前端仍然一无所知的情况下仍然顺利的完成这次大作业（笑

前端（何雨泽）

这次大作业让我在最短的时间内体会到了一个安卓 APP 的开发流程，也让我的前端技能提升了许多。

之前其实有一点 Vue 网页前端开发的经验，以为这次前端开发也会很顺手，没想到还是遇到了相当多的麻烦。安卓 APP 前端相比 html5 开发逻辑复杂一些，而且会产生相当多的意外错误，我为此花掉了不少时间。

即使 AS 这个开发平台已经相当成熟，但上手还是较为困难，几乎所有 api 和类定义都需要上网查。初期遇到困难后，我调整了开发策略，边学习边使用而不是一直学习，以及尽可能多的了解现有 api 的现状，用最合适的轮子造出最好的 APP。

前端不只是开发界面，更重要的是界面的简洁与美观，为此我也花了许多时间在控件的布局上，虽然这些可能最后体现不在分数上，但我觉得这是一个前端开发者应该做的。而且看到赏心悦目的界面时自己（与助教）的心情也会更加舒畅。最后还是有些遗憾用于调整 ui 的时间太少了。

由于暑假小学期、星火、实验室以及突如其来的 4 科考试几乎同时开干，这次大作业留给我的时间相当紧张，算下来完整的只有三到四天，与队友全力冲刺，最后刷了两天夜才完成了所有功能。虽然最后很疲劳，但看到自己倾注心血亲手设计的 APP，还有随之而来的在开发过程中提升的技能，自己还是相当欣慰。同时也要再次感谢负责后端的陈祖浩同学，他提供的后端接口非常完备，让我能专注于前端开发。