# 源码分析。

此次文本标注工具主要由5个Activity和7个class完成，分别包括MainActivity、ReadBookActivity、FileBrowserActivity、TitleActivity和SplashActivity（class不一一列举），

下面将对比较重要的部分进行详尽的阐述。

1. MainActivity

MainActivity是这款app的主界面，视图如下：



该页面绑定了三个控件，分别是TextView、Button以及ListView，TextView用于提示用户文本列表如下，Button用于添加新的文本，ListView用于显示当前已经添加的文本。

FileReader fr = new **FileReader**(file);*//声明并实例化FileReader类，路径为file*

BufferedReader br = new **BufferedReader**(fr);*//声明并实例化BufferedFile类，缓冲读取数据*

data = new ArrayList<>();*//实例化List类，用于添加数据，加载在适配器中*

pathData = new ArrayList<>();*//记录路径*

String line = "" ;*//用于存储读取到的数据*

while((line = br.**readLine**()) != null){*//如果还有下一行数据*

int sub = line.**lastIndexOf**(File.separator);*//截取文件名使用*

String strName = line.**substring**(sub+1, line.**length**());*//截取到文件名*

data.**add**(strName);*//添加文件名*

pathData.**add**(line);*//添加整个文件路径*

}

br.**close**();*//关闭BufferedReader类*

fr.**close**();*//关闭FileReader类*

*//加载数据在适配器*

adapter = new ArrayAdapter<>(MainActivity.this, android.R.layout.simple\_list\_item\_1, data);

bookList.**setAdapter**(adapter);*//设置适配器*

这个页面会读取保存的当前文本目录，并将其显示在ListView上。

private class **OnClick** implements **View.OnClickListener**{

@Override

public void **onClick**(View v) {

*//Intent intent;//跳转到文件浏览选择界面*

Intent intent = new **Intent**(MainActivity.this, FileBrowserActivity.class);

**startActivity**(intent);*//跳转界面*

}

}

Button的点击会跳转到FileBrowserActivity，即开始添加新的文本。

private class **OnItemClick** implements **AdapterView.OnItemClickListener**{

@Override

public void **onItemClick**(AdapterView<?> parent, View view, int position, long id) {

*//用于跳转Activity使用，此处跳转到读取图书界面*

Intent intent = new **Intent**(MainActivity.this, ReadBookActivity.class);

intent.**putExtra**("txtFilePath", pathData.**get**(position));*//传送要打开的txt文件路径*

**startActivity**(intent);*//跳转界面*

}

}

图书列表子项的点击会跳转到ReadBookActivity，即开始阅读文本。

private class **OnItemLongClick** implements **AdapterView.OnItemLongClickListener**{

@Override

public boolean **onItemLongClick**(AdapterView<?> parent, View view, int position, long id) {

longClickPosition = position;*//记录长按了列表哪个子项*

return false;

}

}

监听图书列表的选项长按事件。

@Override

public void **onCreateContextMenu**(ContextMenu menu, View v, ContextMenuInfo menuInfo) {

menu.**add**(0, REMOVE\_BOOK, 0, "移除读书");*//添加菜单子项*

super.**onCreateContextMenu**(menu, v, menuInfo);

}

@Override

public boolean **onContextItemSelected**(MenuItem item) {

if(item.**getItemId**() == REMOVE\_BOOK){*//移除图书处理*

**removeBook**();

}

return super.**onContextItemSelected**(item);

}

创建上下文菜单，长按图书列表选项即弹出移除图书选项，一旦选择移除图书，则调用removeBook方法。

1. ReadBookActivity

一旦点击了某个文本项，则跳转至阅读标注文本界面，视图如下：



可以看到标题栏上的两个Button可以上下页切换，TextView用于显示文本，底部的一些选项是用于标注实体和关系，以及调节文本字体大小的。

这里只阐释上下页切换和标注实体的操作，其余控件的代码类似。

public void **onClick**(View v) {

if(myBook.**GetCurLine**() == 0) {

Intent intent = new **Intent**(ReadBookActivity.this, MainActivity.class);

**startActivity**(intent);

} else {

if(myBook.**GetCurLine**() == 1)

**showBackwardView**(R.string.text\_back, true);

myBook.**SubCurLine**();

bookContent.**setText**(myBook.**GetSpannableStringBuilder**());

}

}

一旦点击上一页，首先判断是第一页，如果满足条件，则跳转回主界面，如果不是第一页，则myBook的索引自减1，并重新设置bookContent上的文本内容。

MyBook是自定义类（所有的自定义类的方法都已省略），定义如下：

public class **MyBook** {

private String[] lines;

private int lineNum, curLine, bookId;

private SpannableStringBuilder[] spannableStringBuilders;

}

myBook存储了当前文本的所有句子，并且由curLine来定位当前页面该显示哪一句，这里的spannableStringBuilders是用来存储标注后的所有句子。

**markEntityText**(lable, sent, myBook, bookContent, dirFile, name, redSpans, redSpansIndex);

一旦点击标注实体按钮，则调用markEntityText方法。

List<Entity> elist = sent[myBook.**GetCurLine**()].**getElist**();

Entity e = new **Entity**(label, start, end - start);

elist.**add**(e);

sent[myBook.**GetCurLine**()].**setElist**(elist);

**writeData**(sent[myBook.**GetCurLine**()].**SentToJSON**(), dirFile, name + myBook.**GetCurLine**() + ".json");

myBook.**GetSpannableStringBuilder**().**setSpan**(spans[spansIndex.**GetIndex**()], start, end, Spannable.SPAN\_INCLUSIVE\_INCLUSIVE);

bookContent.**setText**(myBook.**GetSpannableStringBuilder**());

其核心代码如上，即对sent类进行赋值操作，并转成json串写出保存，最后对所选择的文本进行样式修改。

Sent是自定义类，定义如下：

public class **Sent** {

private String articleID;

private int sentID;

private String sentText;

private List<Relation> Rlist;

private List<Entity>Elist;

}

其包括文本ID，句子ID，句子内容以及用列表存储这句话上的所有实体和关系。

Entity和Relation定义如下：

public class **Entity** {

private int label;

private int offset;

private int length;

}

public class **Relation** {

private Entity entity1;

private Entity entity2;

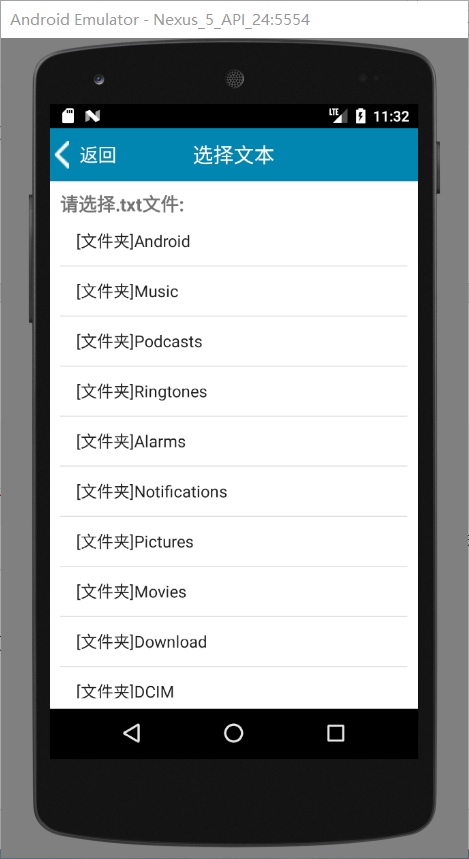
private int RelationStyle;

}

实体包括实体类型label，位移offset和长度length用于标记实体在句子中的位置，关系包括两个实体以及关系类型RelationStyle。

1. FileBrowserActivity

FileBrowserActivity用于打开手机文件目录并选择需要添加的文本，视图如下：



返回按钮不再赘述，这里分析显示文件夹目录的方法。

public void **openFile**(ArrayAdapter adapter, ListView list){

List<SubFile> ndata = new ArrayList<SubFile>();*//用于存放文件名，加载到适配器中*

String strPath = **getIntent**().**getStringExtra**("filename");*//获取上个界面传来的值*

if(strPath == null){*//没有有数据*

strPath = Environment.**getExternalStorageDirectory**().**getPath**();*//根目录*

}

File pathFile = new **File**(strPath);*//要显示的目录*

if(pathFile != null){*//有这个目录*

File[] files = pathFile.**listFiles**();*//获取目录下的所有文件夹与文件*

for(File file : files){*//全部遍历*

if(new **SubFile**(file).**toString**()!=null){*//如果不是筛选掉的文件*

ndata.**add**(new **SubFile**(file));*//添加到mdata对象中*

}

}

*//配置适配器*

adapter = new **ArrayAdapter**(FileBrowserActivity.this, android.R.layout.simple\_list\_item\_1, ndata);

list.**setAdapter**(adapter);*//设置列表的适配去*

}else{*//空文件处理*

Toast.**makeText**(FileBrowserActivity.this , "查找文件为空！", Toast.LENGTH\_SHORT).**show**();

}

}

通过Environment.**getExternalStorageDirectory**().**getPath**()方法得到文件系统的根目录，然后遍历该目录下的所有文件夹和文件，并将它们的名字配置到适配器中，显示在ListView上。

SubFile subFile = (SubFile) parent.**getAdapter**().**getItem**(position);*//获取子项存储的SubFile类，该类可以获取到子项所在路径*

String filename = subFile.**getFile**().**getPath**();*//filename赋值为subFile对象中的路径*

Intent intent = null;*//声明Intent类，用于跳转界面*

if(subFile.**getFile**().**isDirectory**()){*//如果路径为文件夹*

intent = new **Intent**(FileBrowserActivity.this, FileBrowserActivity.class);*//还是跳转到该Activity*

intent.**putExtra**("filename", filename);*//传入路径*

}else{*//文件的话*

File file = new **File**(MainActivity.MYPATH+File.separator+"bookPath.txt");

intent = new **Intent**(FileBrowserActivity.this, MainActivity.class);*//跳转到主界面*

**writeData**(filename);*//调用writeData方法，该方法用于写入数据*

}

**startActivity**(intent);*//跳转界面*

**finish**();*//清除界面*

一旦点击了其中的一项，如果是文件夹，则继续跳转FileBrowserActivity，并将路径传入，如果是文本文件，则跳转MainActivity，并更新已添加文本列表（这些文本的文件名都存储在一个外部txt中，更新列表即对这个外部文件进行修改）。

1. SplashActivity

SplashActivity是用于打开app时显示启动页面，等待其他页面加载完成后，才进入MainActivity，视图如下：



protected void **onCreate**(Bundle savedInstanceState) {

super.**onCreate**(savedInstanceState);

**getWindow**().**requestFeature**(Window.FEATURE\_NO\_TITLE);

**setContentView**(R.layout.activity\_splash);

handler = new **Handler**();

*// 延迟SPLASH\_DISPLAY\_LENGTH时间然后跳转到MainActivity*

handler.**postDelayed**(new **Runnable**() {

@Override

public void **run**() {

Intent intent = new **Intent**(SplashActivity.this, MainActivity.class);

**startActivity**(intent);

SplashActivity.this.**finish**();

**overridePendingTransition**(R.anim.fade\_in, R.anim.fade\_out);

}

}, SPLASH\_DISPLAY\_LENGTH);

}

该Activity的xml文件就是一个ImageView，显示一张图片，延迟一段时间后跳转到MainActivity，这里的overridePendingTransition()方法是用于图片的淡入与淡出。

<alpha

android:fromAlpha="0"

android:toAlpha="1"

android:duration="500" />

通过设置fade\_in.xml和fade\_out.xml的alpha变化和时间，以上面的代码为例，从完全透明到完全不透明，历时500ms，即实现淡入效果。

1. TitleActivity

TitleActivity是继承Activity所写的一个自定义Activity，目的是添加一个标题栏，上述的MainActivity、ReadBookActivity以及FileBrowserActivity都是继承的TitleActivity，比较简单，只是在Activity顶部加上一个TextView用于显示标题，两端各一个Button实现点击跳转等操作，仅仅是为了简化代码所写。

1. TextPage

TextPage是继承EditView所写的一个自定义TextView，因为默认的TextView是不支持选取文本的，对文本标注的实现造成了很大的困难，也会影响用户体验，所以通过继承EditView，可以实现光标选取一段文本。

@Override

public boolean **onTouchEvent**(MotionEvent event) {

int action = event.**getAction**();

Layout layout = **getLayout**();

int line = 0;

switch(action) {

case MotionEvent.ACTION\_DOWN:

line = layout.**getLineForVertical**(**getScrollY**()+ (int)event.**getY**());

off = layout.**getOffsetForHorizontal**(line, (int)event.**getX**());

this.**setSelection**(off);

break;

case MotionEvent.ACTION\_MOVE:

case MotionEvent.ACTION\_UP:

line = layout.**getLineForVertical**(**getScrollY**()+(int)event.**getY**());

curOff = layout.**getOffsetForHorizontal**(line, (int)event.**getX**());

this.**setSelection**(off, curOff);

break;

}

return true;

}

通过触摸事件，一旦按下，则通过setSelection()设置光标位置，移动过程直到抬起，一直更新Selection，ReadBookActivity中会通过TextPage的getStart()和getEnd()方法，获取光标起始和结束的位置，从而对选取的文本进行标注。

1. Json文件的读取和保存

一旦添加了文本之后，便会在mybook目录下创建对应其文件名的文件夹，里面保存了每一句的标注Json文件，阅读该文本时会读取对应的Json文件，对文本进行标注还原，每次标注后都会更新对应的Json文件。

根据文件名找到对应的Json文件并读取这里不进行赘述。

Gson gson = new **Gson**();

sent[i] = gson.**fromJson**(jsonstr, Sent.class);

这里jsonstr即为从Json文件中读取的字符串，通过调用谷歌的API，可以很简单的通过fromJson()方法将jsonstr转为Sent类的实例。

**writeData**(sent[myBook.**GetCurLine**()].**SentToJSON**(), dirFile, name + myBook.**GetCurLine**() + ".json");

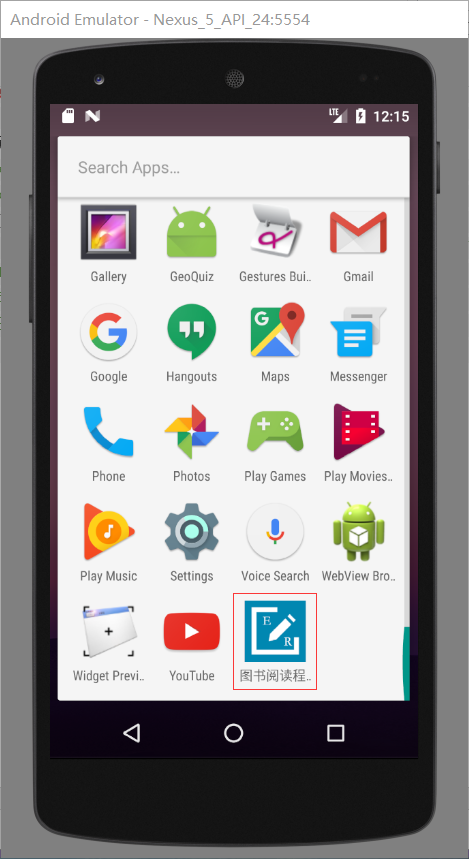
通过调用SentToJSON()方法，将sent转为Json字符串并写入外部文件。

{"Elist":[{"label":8,"length":2,"offset":11},{"label":8,"length":2,"offset":35},{"label":2,"length":5,"offset":41},{"label":1,"length":2,"offset":47},{"label":6,"length":2,"offset":66},{"label":1,"length":2,"offset":69},{"label":1,"length":4,"offset":71}],"Rlist":[{"RelationStyle":0,"entity1":{"label":8,"length":2,"offset":11},"entity2":{"label":8,"length":2,"offset":35}}],"articleID":"1.txt","sentID":0,"sentText":"海外网11月1日电 日本第195次特别国会于当地时间11月1日召开，日本自民党总裁安倍晋三在当日下午举行的众参两院全体大会首相提名选举中被选为第98任首相，全体阁僚和党高层留任。"}

Json文件内容如上。

# 实际操作测试。

APP图标如下：



打开后进入启动界面：



进入主界面：



此时还没有文本添加进来，点击右方+号添加文本：



我们选择1.txt，可以看到1.txt被添加到文本列表：



点击1.txt显示文本内容：



这里进行简单的标注，将通讯社标记为机构，将1日标记为日期，安倍晋三的国籍是日本，进行关系标注：



放大字体：



缩小字体：



退出查看Json文件，在对应的文件夹目录下找到Json文件：



查看刚才所标注的第四句话，文件名为1.txt3.json:



可以看到Json文件与刚才的标注是一致的，包含了四个实体和一个关系，因为一个关系里包含两个实体。