

# Report of Dragable Notebook

Lu Guandong, Qi Zhenlin, Mao Yong, Han Bing

May 18, 2017

## Abstract

This report shows you why we develop the app, the function of the app and the features of the app. We include some source code in this report.

## Contents

<b>1</b>	<b>A brief introduction of the app</b>	<b>1</b>
<b>2</b>	<b>The motivation of developing the app</b>	<b>2</b>
2.1	The exist problems in our life . . . . .	2
2.2	Our solution to this situation . . . . .	2
<b>3</b>	<b>The function of the app</b>	<b>2</b>
3.1	The evaluate system . . . . .	2
3.2	The sorting system . . . . .	2
3.3	The drag algorithm . . . . .	4
<b>4</b>	<b>The feature of the app</b>	<b>12</b>
<b>5</b>	<b>The things we want to do next</b>	<b>13</b>
5.1	The more beautiful UI . . . . .	13
5.2	The learning system . . . . .	13
<b>6</b>	<b>The feeling about the experiment</b>	<b>13</b>

## 1 A brief introduction of the app

This app tends to manage things you should do in the future. You can add things in natural language. Then the app can sort by the importance of the things. The most important things will be listed at the top. Finally, if you are not satisfied with the result, you can slide the items and change the importance of the things.

You can also rebuild the data and in this way the app will try to resort and give a better management of the items.

## 2 The motivation of developing the app

### 2.1 The exist problems in our life

We are often overwhelmed by endless things. Especially in the university, there are infinity things to do. And the only way to do everything best is to try to manage time well.

However, many university students cannot manage time well. For example, some students cannot measure what is important, what should do immediately and what is less important. This can lead to a mess of a student's study, living and so on.

Unfortunately, in app store there are so many notebooks, but none of them can help a student to manage time. Most of them only give the deadline and how many days left.

### 2.2 Our solution to this situation

To deal with the situation I mentioned, we develop a sort system. This system sort by the "importance" of every things. Here "importance" considers many aspects such as time, that is the deadline, the things we should do("Math Exam" or "Throw Trash" and so on) and some words represents importance("immedately", "must to do").

After evaluate the importance of every things, the sort begins. After the sort, the most important things is put in the top. Then the user can do things according to the list.

However, maybe the user are not so satisfied with the sorting result. For example, I can throw my trash immediately but take exam later. At this point, users can drag the things to change the importance to make it more reasonable.

## 3 The function of the app

### 3.1 The evaluate system

The evaluate system gives each sentence a score. Every words in the sentence can affect the score of importance. The words include time, things, and other words and each word has a different score.

We use matching method to find the words. The orginal score of each sentence is 1. And every time we find a matching word, we multiply the score of the words(and typically, the score is greater than 1).

And after the matching process completed, every sentence has a score, and the score represents the importance the app calculated.

The source code shows below:

### 3.2 The sorting system

The sorting algorithm is quite simple. Since we have got every sentence a score, we can get every sentence's importance. We sort by importance, that is, sort by the score. The larger value will be put on the top.

We use the quick sort algorithm to sort the score. The source code shows below:

```

1  import java.util.*;
2  public class autoSort
3  {
4      private static vitalTask[] storeTasks = new vitalTask[11];
5
6      public static void loading()
7      {
8          storeTasks[0] = new vitalTask("    ",1.73);
9          storeTasks[1] = new vitalTask("    ",1.65);
10         storeTasks[2] = new vitalTask("    ",2.2);
11         storeTasks[3] = new vitalTask("    ",2.0);
12         storeTasks[4] = new vitalTask("    ",2.4);
13         storeTasks[5] = new vitalTask("    ",1.77);
14         storeTasks[6] = new vitalTask("    ",2.05);
15         storeTasks[7] = new vitalTask("    ",1.16);
16         storeTasks[8] = new vitalTask("    ",1.14);
17         storeTasks[9] = new vitalTask("    ",1.75);
18         storeTasks[10] = new vitalTask("    ",1.25);
19     }
20
21     public static void autosort(ArrayList<String> tasklist)
22     {
23         loading();
24         oneOfTask[] list = new oneOfTask[tasklist.size()];
25
26         for (int i = 0; i < tasklist.size(); i++)
27         {
28             String tmp = tasklist.get(i);
29             list[i] = new oneOfTask(tmp,i);
30         }
31
32         for (int i = 0; i < tasklist.size(); i++)
33             autoSetValue(list[i]);
34
35         SortTasks(list , 0, list.length);
36
37         for (int i = 0; i < tasklist.size(); i++)
38         {
39             tasklist.set(i, list[i].getName());
40         }
41     }
42
43     public static void autoSetValue(oneOfTask atask)
44     {
45         String name = atask.getName();
46         for (int i = 0; i < storeTasks.length; i++)
47             for (int j = 0; j < name.length() - storeTasks[i].getNa
48                 {
49                     String piecename = name.substring(j,j + storeTa
50                     if (piecename.equals(storeTasks[i].getName()))

```

```

51                                     atask.changeValue(storeTasks[i].getValu
52                                     }
53     }
54
55     public static void SortTasks(oneOfTask[] list, int begin, int end)
56     {
57         if (end - begin <= 1) return;
58         else
59         {
60             double tailnum = list[end-1].getValue();
61             oneOfTask tailobj = list[end - 1];
62             int index = begin;
63             for (int i = begin; i < end - 1; i++)
64             {
65                 if (list[i].getValue() > tailnum)
66                 {
67                     oneOfTask tmp1 = list[i];
68                     list[i] = list[index];
69                     list[index] = tmp1;
70                     index++;
71                 }
72             }
73             for (int i = end - 1; i > index; i--)
74             {
75                 list[i] = list[i - 1];
76             }
77             list[index] = tailobj;
78             SortTasks(list, begin, index);
79             SortTasks(list, index, end);
80         }
81     }
82 }
83 }

```

### 3.3 The drag algorithm

Our drag algorithm focus on the drag process. When you click on the item for a while, the item will be activated, then you can drag it over to other line. And of course, this operation can be done only after the sort.

The source code shows below:

```

1 import android.content.Context;
2 import android.os.Bundle;
3 import android.support.annotation.Nullable;
4 import android.support.v4.app.Fragment;
5 import android.support.v4.content.ContextCompat;
6 import android.support.v4.util.Pair;
7 import android.support.v4.widget.SwipeRefreshLayout;
8 import android.support.v7.app.AppCompatActivity;
9 import android.support.v7.widget.GridLayoutManager;

```

```

10 import android.support.v7.widget.LinearLayoutManager;
11 import android.view.LayoutInflater;
12 import android.view.Menu;
13 import android.view.MenuInflater;
14 import android.view.MenuItem;
15 import android.view.View;
16 import android.view.ViewGroup;
17 import android.widget.TextView;
18 import android.widget.Toast;
19
20 import com.woxthebox.draglistview.DragItem;
21 import com.woxthebox.draglistview.DragListView;
22 import com.woxthebox.draglistview.swipe.ListSwipeHelper;
23 import com.woxthebox.draglistview.swipe.ListSwipeItem;
24
25 import java.util.ArrayList;
26
27 public class ListFragment extends Fragment {
28
29     private ArrayList<Pair<Long, String>> mItemArray;
30     private DragListView mDragListView;
31     private ListSwipeHelper mSwipeHelper;
32     private MySwipeRefreshLayout mRefreshLayout;
33
34     public static ListFragment newInstance() {
35         return new ListFragment();
36     }
37
38     @Override
39     public void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setHasOptionsMenu(true);
42     }
43
44     @Override
45     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
46         View view = inflater.inflate(R.layout.list_layout, container, false);
47         mRefreshLayout = (MySwipeRefreshLayout) view.findViewById(R.id.swipe_refresh_layout);
48         mDragListView = (DragListView) view.findViewById(R.id.drag_list_view);
49         mDragListView.getRecyclerView().setVerticalScrollBarEnabled(true);
50         mDragListView.setDragListListener(new DragListView.DragListListenerAdapter());
51         @Override
52         public void onItemDragStarted(int position) {
53             mRefreshLayout.setEnabled(false);
54             Toast.makeText(mDragListView.getContext(), "Start drag at position: " + position, Toast.LENGTH_SHORT).show();
55         }
56
57         @Override
58         public void onItemDragEnded(int fromPosition, int toPosition) {
59             mRefreshLayout.setEnabled(true);

```

```

60         if (fromPosition != toPosition) {
61             Toast.makeText(mDragListView.getContext(), "End--position:
62         }
63     }
64 });
65
66     mItemArray = new ArrayList<>();
67     String [] p = {"", "B p p t", "
68     for (int i = 0; i < 5; i++) {
69         mItemArray.add(new Pair<>((long) i, p[i]));
70     }
71
72     mRefreshLayout.setScrollingView(mDragListView.getRecyclerView());
73     mRefreshLayout.setColorSchemeColors(ContextCompat.getColor(getContext(),
74     mRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshLi
75     @Override
76     public void onRefresh() {
77         mRefreshLayout.postDelayed(new Runnable() {
78             @Override
79             public void run() {
80                 mRefreshLayout.setRefreshing(false);
81             }
82         }, 2000);
83     }
84 });
85
86     mDragListView.setSwipeListener(new ListSwipeHelper.OnSwipeListenerAdapt
87     @Override
88     public void onItemSwipeStarted(ListSwipeItem item) {
89         mRefreshLayout.setEnabled(false);
90     }
91
92     @Override
93     public void onItemSwipeEnded(ListSwipeItem item, ListSwipeItem.Swipe
94         mRefreshLayout.setEnabled(true);
95
96     // Swipe to delete on left
97     if (swipedDirection == ListSwipeItem.SwipeDirection.LEFT) {
98         Pair<Long, String> adapterItem = (Pair<Long, String>) item;
99         int pos = mDragListView.getAdapter().getPositionForItem(adapterItem);
100        mDragListView.getAdapter().removeItem(pos);
101    }
102 }
103 });
104
105     setupListRecyclerView();
106     return view;
107 }
108
109 @Override

```

```

110     public void onActivityCreated(@Nullable Bundle savedInstanceState) {
111         super.onActivityCreated(savedInstanceState);
112         ((AppCompatActivity) getActivity()).getSupportActionBar().setTitle("Task")
113     }
114
115     @Override
116     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
117         super.onCreateOptionsMenu(menu, inflater);
118         inflater.inflate(R.menu.menu_list, menu);
119     }
120
121     @Override
122     public void onPrepareOptionsMenu(Menu menu) {
123         super.onPrepareOptionsMenu(menu);
124         menu.findItem(R.id.action_disable_drag).setVisible(mDragListView.isDragEnabled());
125         menu.findItem(R.id.action_enable_drag).setVisible(!mDragListView.isDragEnabled());
126     }
127
128     @Override
129     public boolean onOptionsItemSelected(MenuItem item) {
130         switch (item.getItemId()) {
131             case R.id.action_disable_drag:
132                 mDragListView.setDragEnabled(false);
133                 getActivity().supportInvalidateOptionsMenu();
134                 return true;
135             case R.id.action_enable_drag:
136                 mDragListView.setDragEnabled(true);
137                 getActivity().supportInvalidateOptionsMenu();
138                 return true;
139             case R.id.action_list:
140                 setupListRecyclerView();
141                 return true;
142             case R.id.action_grid_vertical:
143                 setupGridVerticalRecyclerView();
144                 return true;
145             case R.id.action_grid_horizontal:
146                 setupGridHorizontalRecyclerView();
147                 return true;
148             case R.id.add_task:
149                 ADD_TASK();
150                 return true;
151         }
152         return super.onOptionsItemSelected(item);
153     }
154
155     private void ADD_TASK() {
156
157     } //To Be Realized
158
159     private void setupListRecyclerView() {

```

```

160         mDragListView.setLayoutManager(new LinearLayoutManager(getContext()));
161         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.list_item);
162         mDragListView.setAdapter(listAdapter, true);
163         mDragListView.setCanDragHorizontally(false);
164         mDragListView.setCustomDragItem(new MyDragItem(getContext(), R.layout.list_item));
165     }
166
167     private void setupGridVerticalRecyclerView() {
168         mDragListView.setLayoutManager(new GridLayoutManager(getContext(), 4));
169         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.grid_item);
170         mDragListView.setAdapter(listAdapter, true);
171         mDragListView.setCanDragHorizontally(true);
172         mDragListView.setCustomDragItem(null);
173     }
174
175     private void setupGridHorizontalRecyclerView() {
176         mDragListView.setLayoutManager(new GridLayoutManager(getContext(), 4, LinearLayoutManager.HORIZONTAL));
177         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.grid_item);
178         mDragListView.setAdapter(listAdapter, true);
179         mDragListView.setCanDragHorizontally(true);
180         mDragListView.setCustomDragItem(null);
181     }
182
183     private static class MyDragItem extends DragItem {
184
185         MyDragItem(Context context, int layoutId) {
186             super(context, layoutId);
187         }
188
189         @Override
190         public void onBindDragView(View clickedView, View dragView) {
191             CharSequence text = ((TextView) clickedView.findViewById(R.id.text));
192             ((TextView) dragView.findViewById(R.id.text)).setText(text);
193             dragView.findViewById(R.id.item_layout).setBackgroundColor(dragView.getResources().getColor(R.color.drag_item));
194         }
195     }
196
197 }
198 import android.content.Context;
199 import android.os.Bundle;
200 import android.support.annotation.Nullable;
201 import android.support.v4.app.Fragment;
202 import android.support.v4.content.ContextCompat;
203 import android.support.v4.util.Pair;
204 import android.support.v4.widget.SwipeRefreshLayout;
205 import android.support.v7.app.AppCompatActivity;
206 import android.support.v7.widget.GridLayoutManager;
207 import android.support.v7.widget.LinearLayoutManager;
208 import android.view.LayoutInflater;
209 import android.view.Menu;

```



```

210 import android.view.MenuInflater;
211 import android.view.MenuItem;
212 import android.view.View;
213 import android.view.ViewGroup;
214 import android.widget.TextView;
215 import android.widget.Toast;
216
217 import com.woxthebox.draglistview.DragItem;
218 import com.woxthebox.draglistview.DragListView;
219 import com.woxthebox.draglistview.swipe.ListSwipeHelper;
220 import com.woxthebox.draglistview.swipe.ListSwipeItem;
221
222 import java.util.ArrayList;
223
224 public class ListFragment extends Fragment {
225
226     private ArrayList<Pair<Long, String>> mItemArray;
227     private DragListView mDragListView;
228     private ListSwipeHelper mSwipeHelper;
229     private MySwipeRefreshLayout mRefreshLayout;
230
231     public static ListFragment newInstance() {
232         return new ListFragment();
233     }
234
235     @Override
236     public void onCreate(Bundle savedInstanceState) {
237         super.onCreate(savedInstanceState);
238         setHasOptionsMenu(true);
239     }
240
241     @Override
242     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
243         View view = inflater.inflate(R.layout.list_layout, container, false);
244         mRefreshLayout = (MySwipeRefreshLayout) view.findViewById(R.id.swipe_refresh_layout);
245         mDragListView = (DragListView) view.findViewById(R.id.drag_list_view);
246         mDragListView.getRecyclerView().setVerticalScrollBarEnabled(true);
247         mDragListView.setDragListListener(new DragListView.DragListListenerAdapter());
248
249         @Override
250         public void onItemDragStarted(int position) {
251             mRefreshLayout.setEnabled(false);
252             Toast.makeText(mDragListView.getContext(), "Start -- position: " + position, Toast.LENGTH_SHORT).show();
253         }
254
255         @Override
256         public void onItemDragEnded(int fromPosition, int toPosition) {
257             mRefreshLayout.setEnabled(true);
258             if (fromPosition != toPosition) {
259                 Toast.makeText(mDragListView.getContext(), "End -- position: " + fromPosition, Toast.LENGTH_SHORT).show();
260             }
261         }
262     }

```

```

260     }
261     });
262
263     mRecyclerView = new ArrayList<>();
264     String [] p = { " ", " ", " B p p t ", "
265     for (int i = 0; i < 5; i++) {
266         mRecyclerView.add(new Pair<>((long) i, p[i]));
267     }
268
269     mRecyclerView.setLayoutManager(mDragListView.getRecyclerView());
270     mRecyclerView.setColorSchemeColors(ContextCompat.getColor(getContext(),
271     mRecyclerView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
272         @Override
273         public void onRefresh() {
274             mRecyclerView.postDelayed(new Runnable() {
275                 @Override
276                 public void run() {
277                     mRecyclerView.setRefreshing(false);
278                 }
279             }, 2000);
280         }
281     });
282
283     mDragListView.setSwipeListener(new ListSwipeHelper.OnSwipeListenerAdapter() {
284         @Override
285         public void onItemSwipeStarted(ListSwipeItem item) {
286             mRecyclerView.setEnabled(false);
287         }
288
289         @Override
290         public void onItemSwipeEnded(ListSwipeItem item, ListSwipeItem.SwipeDirection direction) {
291             mRecyclerView.setEnabled(true);
292
293             // Swipe to delete on left
294             if (swipedDirection == ListSwipeItem.SwipeDirection.LEFT) {
295                 Pair<Long, String> adapterItem = (Pair<Long, String>) item.getAdapterItem();
296                 int pos = mDragListView.getAdapter().getPositionForItem(adapterItem);
297                 mDragListView.getAdapter().removeItem(pos);
298             }
299         }
300     });
301
302     setupListRecyclerView();
303     return view;
304 }
305
306 @Override
307 public void onActivityCreated(@Nullable Bundle savedInstanceState) {
308     super.onActivityCreated(savedInstanceState);
309     ((AppCompatActivity) getActivity()).getSupportActionBar().setTitle("Task")

```

```

310     }
311
312     @Override
313     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
314         super.onCreateOptionsMenu(menu, inflater);
315         inflater.inflate(R.menu.menu_list, menu);
316     }
317
318     @Override
319     public void onPrepareOptionsMenu(Menu menu) {
320         super.onPrepareOptionsMenu(menu);
321         menu.findItem(R.id.action_disable_drag).setVisible(mDragListView.isDrag
322         menu.findItem(R.id.action_enable_drag).setVisible(!mDragListView.isDrag
323     }
324
325     @Override
326     public boolean onOptionsItemSelected(MenuItem item) {
327         switch (item.getItemId()) {
328             case R.id.action_disable_drag:
329                 mDragListView.setDragEnabled(false);
330                 getActivity().supportInvalidateOptionsMenu();
331                 return true;
332             case R.id.action_enable_drag:
333                 mDragListView.setDragEnabled(true);
334                 getActivity().supportInvalidateOptionsMenu();
335                 return true;
336             case R.id.action_list:
337                 setupListRecyclerView();
338                 return true;
339             case R.id.action_grid_vertical:
340                 setupGridVerticalRecyclerView();
341                 return true;
342             case R.id.action_grid_horizontal:
343                 setupGridHorizontalRecyclerView();
344                 return true;
345             case R.id.add_task:
346                 ADD_TASK();
347                 return true;
348         }
349         return super.onOptionsItemSelected(item);
350     }
351
352     private void ADD_TASK() {
353
354     }//To Be Realized
355
356     private void setupListRecyclerView() {
357         mDragListView.setLayoutManager(new LinearLayoutManager(getContext()));
358         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.list_item
359         mDragListView.setAdapter(listAdapter, true);

```

```

360         mDragListView.setCanDragHorizontally(false);
361         mDragListView.setCustomDragItem(new MyDragItem(getContext(), R.layout.l
362     }
363
364     private void setupGridVerticalRecyclerView() {
365         mDragListView.setLayoutManager(new GridLayoutManager(getContext(), 4));
366         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.grid_item);
367         mDragListView.setAdapter(listAdapter, true);
368         mDragListView.setCanDragHorizontally(true);
369         mDragListView.setCustomDragItem(null);
370
371     }
372
373     private void setupGridHorizontalRecyclerView() {
374         mDragListView.setLayoutManager(new GridLayoutManager(getContext(), 4, I
375         ItemAdapter listAdapter = new ItemAdapter(mItemArray, R.layout.grid_item);
376         mDragListView.setAdapter(listAdapter, true);
377         mDragListView.setCanDragHorizontally(true);
378         mDragListView.setCustomDragItem(null);
379     }
380
381     private static class MyDragItem extends DragItem {
382
383         MyDragItem(Context context, int layoutId) {
384             super(context, layoutId);
385         }
386
387         @Override
388         public void onBindDragView(View clickedView, View dragView) {
389             CharSequence text = ((TextView) clickedView.findViewById(R.id.text)
390             ((TextView) dragView.findViewById(R.id.text)).setText(text);
391             dragView.findViewById(R.id.item_layout).setBackgroundColor(dragView
392         }
393     }
394 }

```

## 4 The feature of the app

We try our best to make the evaluate system more reasonable, so up to now it can evaluate how important the thing is very well. Only the evaluation system be more reasonable can we sort well and give the more reasonable solution.

As for the sort algorithm, we use the typical quick sort algorithm. It is more efficient than bubble sort.

The most important study is about the drag operation. We make the drag as smooth as possible, which gives the user a better and more convenient use.

## 5 The things we want to do next

We are going to make the app much more convenient, not just for this experiment. So next we are going to update in the following aspects:

### 5.1 The more beautiful UI

As at the first time we are try our best developing our algorithm, we are not focus as much on the UI and the interface.

So next time, we are going to develop a more beautiful UI and interface to satisfy the user's demand.

### 5.2 The learning system

Today our evaluate system is based on dictionary, that is, use dictionary to regulate the score of each words. However, this is not always the truth as the demand for each user is different.

So we are going to change our evaluate system into a data-based system, using users' sort and the big data to analyse what is more important.

## 6 The feeling about the experiment

From this experiment, we learn how to develop an app in the mobile phone, and try to solve the problems in our daily life.

At first time, we are not very clear about how to develop, but then we try to learn and try by hand, and finally we are able to manage some basic operation in Android Studio.

And in the experiment, we came across some difficulties. However, we didn't give up and try to find the solutions. Finally, we developed what we want. This is a great experience.

The experiment is coming to an end, but our development and study is not. So it is just a beginning to go and it encourages us to develop greater app in the future.