Demo设计

程序文件架构

```
|- data
|    |- song_info.json                    # 歌曲数据
|    |- user_info_without_moments.json    # 用户数据
|    |- SongData.py                       # 歌曲数据的加载和查找器
|    -- UserData.py                       # 用户数据的加载和查找器
|- model
|    |- DefaultRecom.py                   # Demo当前所用的推荐模型
|    -- ...                               # 可以与其他模型相结合
|- ui
|    |- MainWindow.py                     # 主窗口
|    |- SongButton.py                     # 展示歌曲信息的按钮
|    |- SongWindow.py                     # 歌曲卡片
|    -- UrlLabel.py                      # URL链接
|- util
     -- UrlToPixmap.py                   # 从URL获取图片
```

运行环境与方法

本UI的构建依赖于Python3 / PyQt5

进入`ui`文件夹下使用如下命令进行运行：

```
python MainWindow.py
```

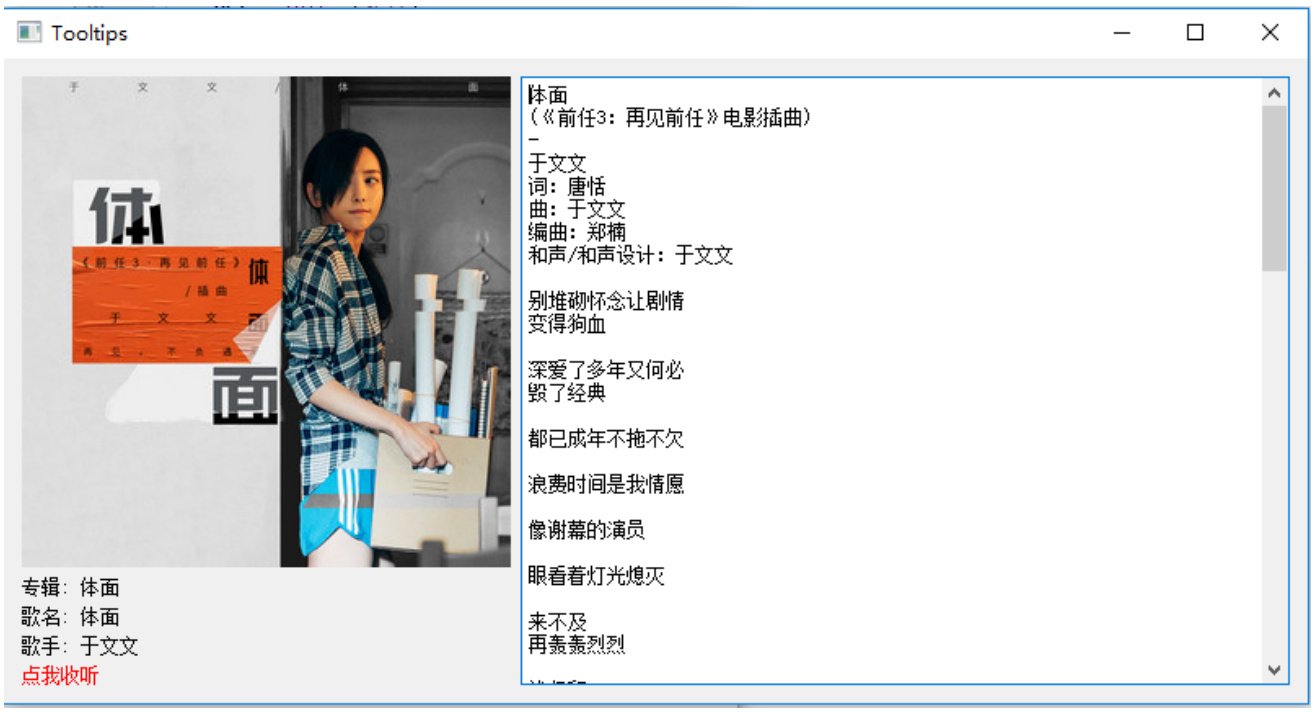
弹出窗口后使用搜索按钮进行查找用户

运行图像

运行时弹出如下窗口：

在输入栏中输入用户的id，点击"搜索用户"按钮，查看对应id用户的结果。

点击右侧的推荐歌曲按钮，可以打开一个新的窗口如下所示：



歌曲卡片会给出歌曲的专辑封面，专辑名称，歌曲名称，歌手，歌词等。点击红字可以打开浏览器访问相应的网页进行收听歌曲。

程序功能介绍

Song/User Data

两类的功能主要是读取歌曲和用户的数据，整理为list of dictionary，分别留有一个查找接口，通过用户和音乐的ID获取详细的信息。

DefaultRecom

这是一个简单的用于测试的推荐器，可以被替换为其他算法的推荐器。需要实现一个接口`get_recom`，输入为用户ID，输出为推荐的前5首歌曲ID。

MainWindow

```
init_data: 载入用户数据和歌曲数据，便于在本地进行查找
init_recommender: 载入推荐器
init_list: 载入默认推荐列表
init_ui: 载入UI界面设置
init_connect: 载入信号-槽设置
search_user: 对于输入的用户进行查找
update_ui: 根据查找结果更新界面
```

具体的搜索过程为：

1. 输入用户ID，点击按钮进行查找
2. 在用户信息中根据对该用户进行搜索，更新界面的用户信息
3. 通过推荐器接口，得到被推荐的5首歌曲ID
4. 分别根据歌曲ID，查找到歌曲的详细信息
5. 根据歌曲详细信息，更新界面上相应按钮

SongButton

一个基于`PyQt5`元件库`QPushButton`原件的类，其中额外包含了歌曲的具体信息，根据歌曲名设置按钮上的文字信息，点击后会打开一个歌曲卡片界面对于歌曲的详细信息进行展示。

SongWindow

点击`SongButton`后弹出的音乐卡片界面。包含了图中所示的若干信息。

UrlLabel

一个基于`PyQt5`元件库`QLabel`原件的类，重新设置了其颜色，并记录了URL的地址，在点击时设置为打开链接。

UrlToPixmap

为了根据URL生成歌曲封面图，需要进行一些处理。