系统关键模块实现

推荐模块实现

1. 概述

推荐模块包含的排行榜和图像、用户、标签推荐子模块以综合美学评分、点赞数、评论数计算的权重为导向为用户进行信息推荐，这是该社区“美学导向“这一宗旨的主要体现。

在设计图像权重的过程中，为了保证美学评分这一百分制分数和点赞数、评论数这两个和社区总体用户数目挂钩的数值的平衡，同时让系统在点赞数和评论数过少时去推动美学评分的影响力，利用以下公式对图像权重进行计算：

假定当前社区最大用户为100人。

这样的权重设计具有以下目的：

* 美学评分最大可占比总权重50%，保证美学评分的主要导向性；
* 点赞数和评论数占比比例为4比1，因为用户最多可以点一个赞，而可以多次评论；
* 使用点赞数、评论数和社区总数乘以权的比例来替代点赞数和评论数，有效避免点赞数评论数在无限增长时产生的负面影响；
* 动态中所有图片的点赞数和评论数依赖于动态，都相同，但权重有所差异。

1. 排行榜子模块
2. 核心算法

按图片权重排序获取一周内或一日内所以动态中图片权重排行前十的图片。权重设计见概述。获取函数如下：

1. **def** get\_by\_weight(self, start, end):
2. **from** Stack.config **import** RANK\_LIMIT
3. **return** self.query\
4. .filter(start < Post.ptime, Post.ptime <= end)\
5. .filter(self.pid == Post.pid)\
6. .order\_by(self.weight.desc())\
7. .limit(RANK\_LIMIT)\
8. .all()
9. 前端设计

排行榜分为日榜和周榜两个Tab，每个Tab内容相同，内部有总数不超过十的子图像模块，模块包含排行图片、下方的发布人基本信息和点赞数以及点赞按钮。如图。



1. 后端实现

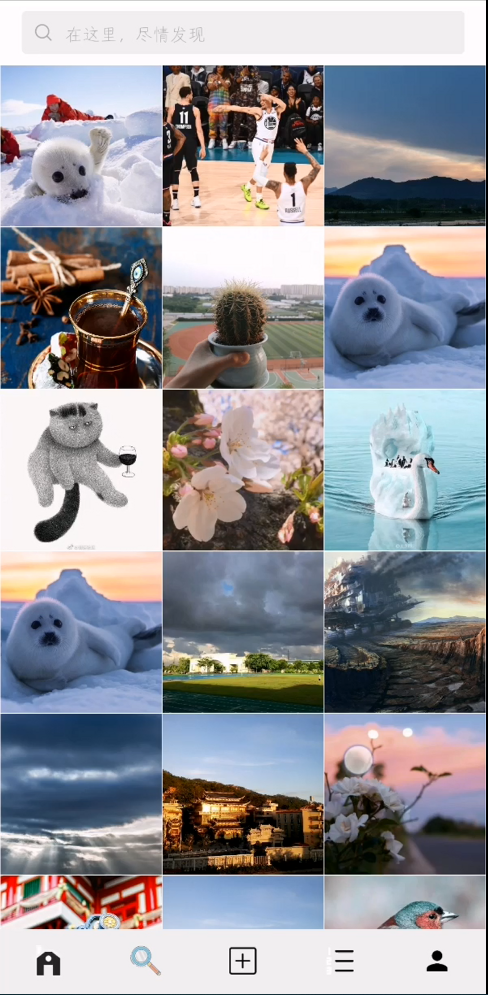
由于大部分的后端代码相似，这里只展示排行榜部分后端代码。

1. 获取表单参数
2. @app.route('/rank/get\_rank', methods=['POST'])
3. **def** get\_rank():
4. data = {}
5. phonenum = request.form.get('phonenum')
6. period = int(request.form.get('period'))
7. cur = request.form.get('time')
8. 后端验证
9. **try**:
10. schema(
11. {
12. "phonenum": phonenum,
13. "ptime": cur,
14. "period": period
15. }
16. )
17. conforms\_to\_schema = True
18. **except** MultipleInvalid as e:
19. data['status'] = 400
20. conforms\_to\_schema = False
21. **if** "expected" **in** e.msg:
22. data['message'] = e.path[0] + " is not in the correct format"
23. **else**:
24. data['message'] = e.msg + " for " + e.path[0]
25. 验证成功则获取数据库中信息
26. **if** conforms\_to\_schema:
27. **try**:
28. images = Image.get\_by\_weight(Image, start=pre, end=cur)
29. res = list()
30. **for** image **in** images:
31. imgdic = Image.out(Image, image)
32. post = Post.get(Post, image.pid)
33. imgdic['likes'] = LikeTable.get\_count\_by\_pid(LikeTable, pid=post.pid)
34. imgdic['isliked'] = True **if** LikeTable.get\_by\_pp(LikeTable, pid=post.pid,
35. phonenum=phonenum) **is** **not** None **else** False
36. usr = User.partly\_out(User, User.get(User, post.phonenum))
37. imgdic['user'] = usr
38. res.append(imgdic)
39. data['message'] = res
40. data['status'] = 200
41. **except** Exception as e:
42. data['status'] = 406
43. data['message'] = str(e)
44. 返回JSON格式结果
45. **return** jsonify(data)
46. 图像推荐子模块
47. 核心算法

随机获取权重大于一定阈值的一定数目的所有动态的图片。权重设计见概述。获取函数如下：

1. **def** get\_recommended(self):
2. **from** Stack.config **import** RECOMMEND\_LIMIT
3. **return** self.query\
4. .filter(self.weight > 10) \
5. .order\_by(func.rand()) \
6. .limit(RECOMMEND\_LIMIT) \
7. .all()
8. 前端设计

图像推荐通过如下图的照片墙来展示，由于快应用暂时无法显示瀑布流式设计，采用固定大小的方块瓷砖式排列，通过cover来显示不同大小的图片。

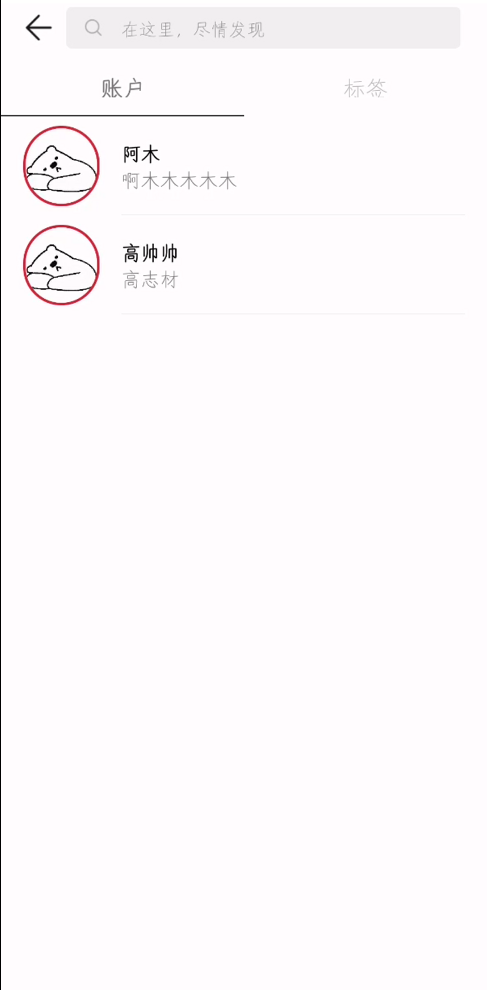


1. 用户推荐子模块
2. 核心算法

通过统计当前登录用户关注的所有用户的共同关注用户，建立共同关注用户到共同关注数的映射表，依序为用户推荐。具体设计如下：

1. followees = FollowTable.get\_followees(FollowTable, follower=phonenum)
2. res = dict()
3. **for** followee **in** followees:
4. # print(FollowTable.out(FollowTable, followee))
5. user = User.get(User, phonenum=followee.followee)
6. fs = FollowTable.get\_followees(FollowTable, follower=user.phonenum)
7. **for** f **in** fs:
8. u = User.get(User, phonenum=f.followee)
9. **if** u.phonenum **not** **in** res:
10. res[u.phonenum] = 1
11. res[u.phonenum] += 1
13. res = sorted(res.items(), key=**lambda** x: x[1], reverse=True)
14. **from** Stack.config **import** U\_RECOMMEND\_LIMIT
15. count = 0
16. ans = list()
17. **for** pn, cnt **in** res:
18. **if** pn != phonenum **and** FollowTable.get\_by\_ff(FollowTable, phonenum, pn) **is** None:
19. userdict = User.out(User, User.get(User, phonenum=pn))
20. userdict['count'] = cnt
21. ans.append(userdict)
22. count += 1
23. **if** count >= U\_RECOMMEND\_LIMIT:
24. **break**
25. 前端设计

在后台加载推荐，点击搜索栏后自动显示。



1. 标签推荐子模块
2. 核心算法

统计用户已发布动态的最常见标签，为用户推荐同质图像。具体实现如下：

1. posts = Post.get\_all\_posts(Post, phonenum)
2. res = dict()
3. **for** post **in** posts:
4. images = Image.get\_by\_pid(Image, post.pid)
5. **for** image **in** images:
6. labels = image.label.split()
7. **for** label **in** labels:
8. **if** label **not** **in** res:
9. res[label] = 1
10. res[label] += 1
12. res = sorted(res.items(), key=**lambda** x: x[1], reverse=True)
13. **from** Stack.config **import** L\_RECOMMEND\_LIMIT
14. count = 0
15. ans = list()
16. **for** label, cnt **in** res:
17. **if** label != '其它':
18. images = Image.search(Image, keyword=label)
19. **for** image **in** images:
20. **if** Post.get\_by\_pp(Post, image.pid, phonenum) **is** None:
21. ans.append(Image.out(Image, image))
22. count += 1
23. **if** count >= L\_RECOMMEND\_LIMIT:
24. **break**
25. 前端设计

在后台加载推荐，点击搜索栏后自动显示。

