Tiger AppStore

Recommender System

Guangcheng Lu

Recommender - architecture

All data about Apps

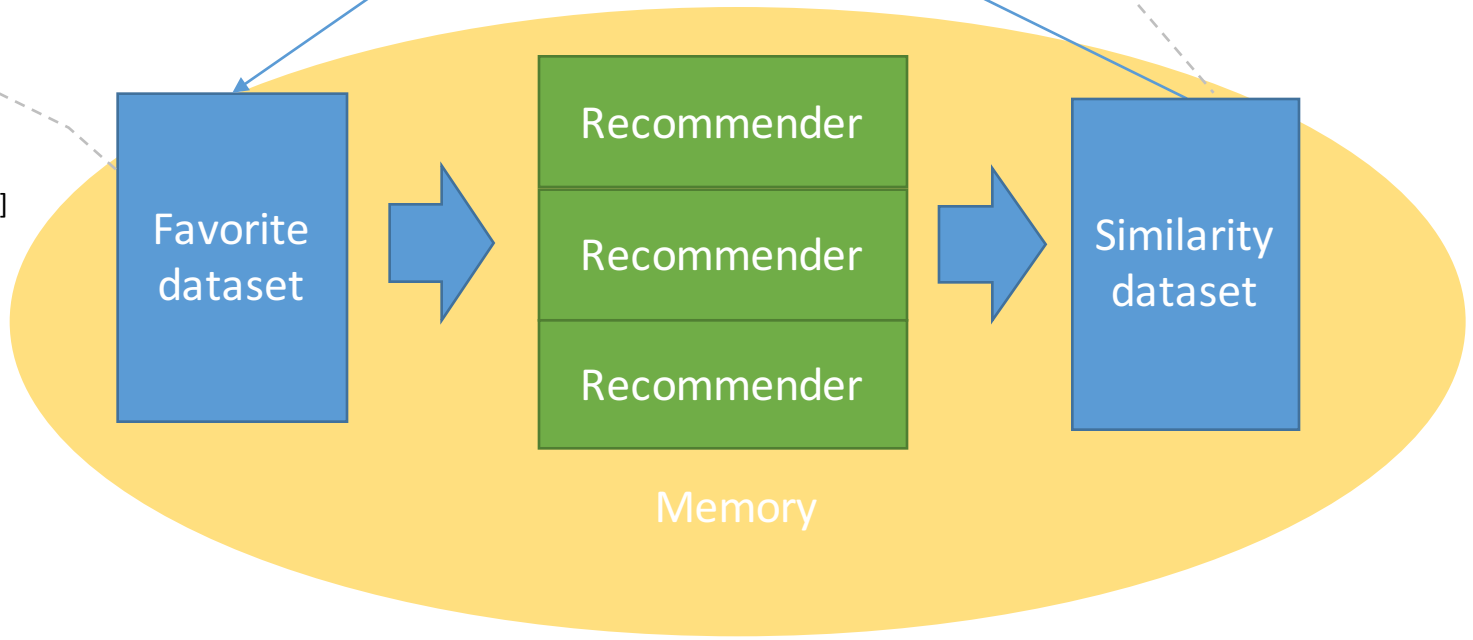
MongoDB

The lists of each App's most similar Apps

- WeChat similar to [WhatsApp, Facebook messenger…]
- Sohu similar to [Yahoo news, Medium …]
- AngryBirds similar to [Candy crush sage, Cut the rope …]
- …

The lists of each user's favorite Apps

- U1 likes [WeChat, Sohu, Sina, Facebook]
- U2 likes [WhatsApp, Google, Chrome, WeChat]
- U3 likes [Chrome, AngryBirds, WeChat, Slack]
- …



Basic Setup for Development

Caveat - This is the exact steps on Windows 10

- Install Python 2.7
- Install PyMongo
  - http://api.mongodb.org/python/current/installation.html
- Install and setup MongoDB
  - https://docs.mongodb.org/manual/installation/
  - If it doesn't run, remember to set the path in environment variable
  - Example: add "C:\Program Files\MongoDB\Server\3.2\bin" to path
- The program assume MongoDB instance is running on the default host:port
  - $ mongod --> start MongoDB instance

Outline

- Load data into MongoDB
- Calculate an App's top-5 related apps
- Save data into MongoDB

Load Data into MongoDB

- Load data into MongoDB
  - Import JSON file
    - $ cd <path_to_json_file>
    - $ mongoimport --db appstore --collection user_download_history --drop --file user_download_history.json
    - $ mongoimport --db appstore --collection app_info --drop --file app_info.json
  - Data stored in
    - Hierarchy: database/collection/document
    - Database: appstore
    - Collections: user_download_history, app_info

Result

```
C:\WINDOWS\system32>mongo
MongoDB shell version: 3.2.0
connecting to: test
> use appstore
switched to db appstore
> db.app_info.findOne()
{
        "_id" : ObjectId("568976f2bcb96fc5c0d62f19"),
        "score" : "9",
        "title" : "泡泡龙亚特兰蒂斯",
        "url" : "http://appstore.huawei.com:80/app/C10145675",
        "app_id" : "C10145675",
        "thumbnail_url" : "http://appimg.hicloud.com/hwmarket/files/application/icon144/cb3c6ce12b73424990921097fe20a7b1.png",
        "intro" : "泡泡龙亚特兰蒂斯:一款令人着迷的泡泡龙游戏。在经典的游戏模式中增加了独有的BOSS战，每个场景都 有独特的守护者等待你的挑战。多种多样的奇趣道具供你使用。打BOSS，秀操作，耍道具让你爱不释手。欢乐之旅由一段美丽的故事带你进入。前所未有的体验，带给你神奇的亚特兰蒂斯之旅。",
        "developer" : "深圳市灵游科技有限公司"
}
> db.user_download_history.findOne()
{
        "_id" : ObjectId("568976d7bcb96fc5c0d620c0"),
        "user_id" : 2,
        "download_history" : [
                "C10148546",
                "C10237091",
                "C10187028",
                "C10189589",
                "C10167895",
                "C10141383",
                "C10136202"
        ]
}
```

Dataset - app_info

{
  "score":"8",
  "title":"京东",
  "url":"http://appstore.huawei.com:80/app/C20252",
  "app_id":"C20252",
  "thumbnail_url":"http://appimg.hicloud.com/hwmarket/files/application/icon144/365b65540dff47619efd1ec5bd682dc8.png",
  "intro":"【掌上京东 实惠轻松 2亿用户购物首选客户端】 1、新人特权：客户端新人首单满79元送79元大礼包 2、专享特权：每天多款客户端专享价商品 3、闪电到货：京东自营商品211限时达 【真实用户评价】 京东商城网购评价相对比较好。货真价实，物流也比较快，客服人员比较热情，最重要就是活动多。京东商城值得信赖——by“mC烟味",
  "developer":"北京京东叁佰陆拾度电子商务有限公司"
}

Dataset - user_download_history


```
{
  "user_id":1,
  "download_history":[
    "C2217",
    "C20252",
    "C2682",
    "C183901",
    "C41529",
    "C10220136",
    "C5373",
    "C34075",
    "C5683",
    "C3466"
  ]
}
```

```
{
  "user_id":2,
  "download_history":[
    "C2217",
    "C57804",
    "C10047107",
    "C77434",
    "C5373",
    "C6015363",
    "C5165",
    "C34075",
    "C10196888",
    "C10191116"
  ]
}
```

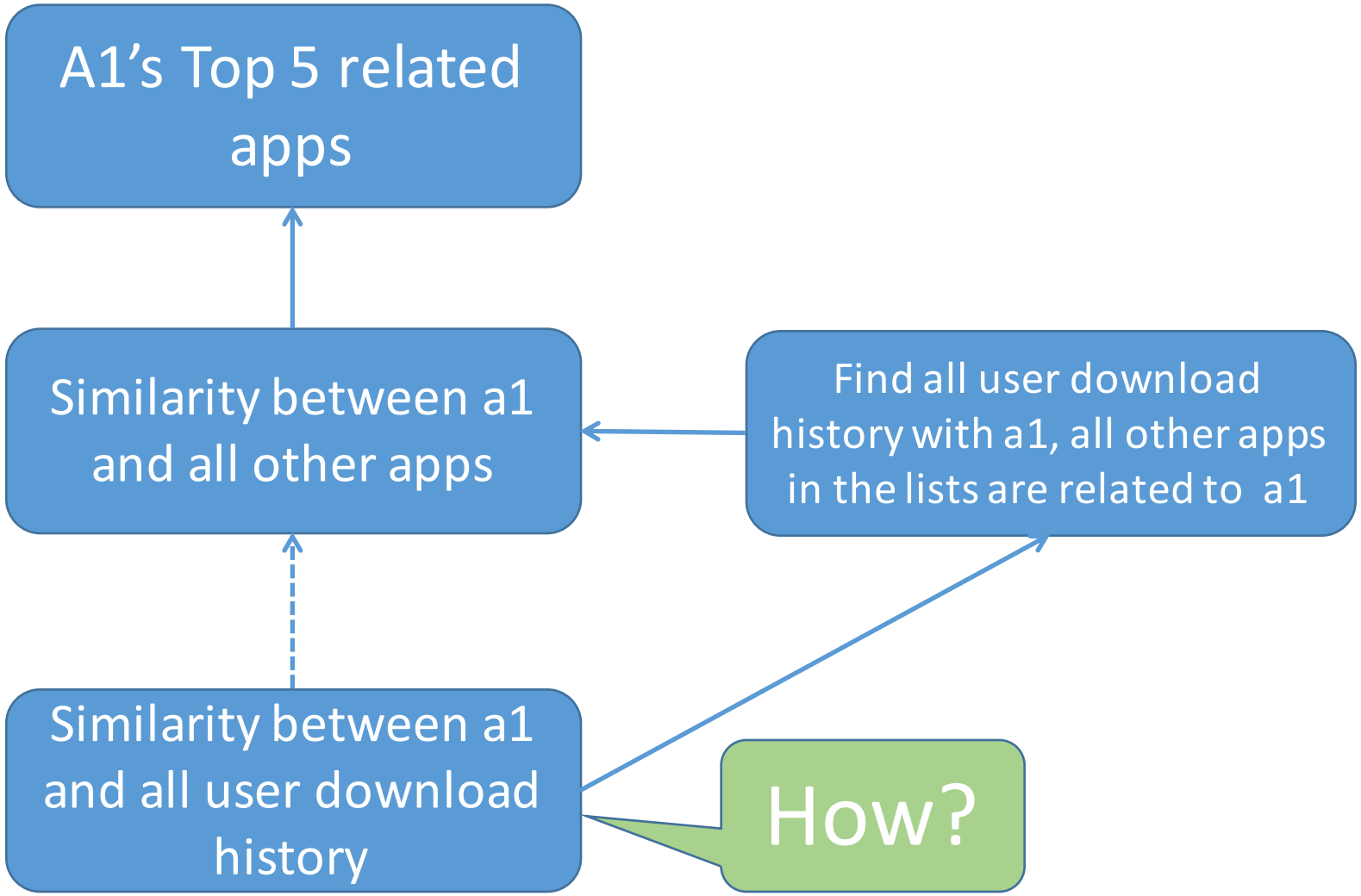
```
{
  "user_id":3,
  "download_history":[
    "C174391",
    "C10129690",
    "C10114178",
    "C5980",
    "C5683",
    "C10191116"
  ]
}
```

Calculate an App’s Top-5 realated apps

- Load data into MongoDB
- Calculate an App's top-5 related apps

How to calculate an APP's top-5 related App

App a1's Top 5


A1's Top 5 related apps
Similarity between a1 and all other apps
Find all user download history with a1, all other apps in the lists are related to a1
Similarity between a1 and all user download history
How?

Cosine Similarity

$$u_1 = \{a_3, a_5, a_7, a_{11}\}$$
$$u_2 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$$

$$u_1 \cap u_2 = \{a_3, a_5, a_7\}$$

$$CosineSimilarity(u_1, u_2) = \frac{|u_1 \cap u_2|}{\sqrt{|u_1| * |u_2|}} = \frac{3}{\sqrt{4*9}} = 0.5$$

Key Idea: More same Apps & shorter lists -> More similar

Code - Cosine Similarity

- Call the function in helper class when needed
  - helper = Helper()
  - similarity = helper.cosine_similarity(app_list1, app_list2)

```python
import math

class Helper(object):

    @classmethod
    def cosine_similarity(cls, app_list1, app_list2):
        match_count = cls.__count_match(app_list1, app_list2)
        return float(match_count) / math.sqrt( len(app_list1) * len(app_list2) )

    @classmethod
    def __count_match(cls, list1, list2):
        count = 0
        for element in list1:
            if element in list2:
                count += 1
        return count
```

Similarity between an App and a user's history

$$App_3 = \{a_3\}$$
$$u_1 = \{a_3, a_5, a_7, a_{11}\}$$
$$u_2 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$$

$$CosineSimilarity(App_3, u_1) = \frac{|App_3 \cap u_1|}{\sqrt{|App_3| * |u_1|}} = \frac{1}{\sqrt{1*4}} = 0.50$$

$$CosineSimilarity(App_3, u_2) = \frac{|App_3 \cap u_2|}{\sqrt{|App_3| * |u_2|}} = \frac{1}{\sqrt{1*9}} = 0.33$$

Find all the users related to an App

$App_3 = \{a_3\}$

$u_1 = \{a_3, a_5, a_7, a_{11}\}$
$Similarity(App_3, u_1) = 0.50$

$u_2 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$
$Similarity(App_3, u_2) = 0.33$

$u_8 = \{a_1, a_3, a_4, a_5, a_{16}, a_{37}, a_{44}\}$
$Similarity(App_3, u_2) = 0.38$

$u_9 = \{a_3, a_7, a_{14}, a_{44}, a_{52}\}$
$Similarity(App_3, u_2) = 0.45$

Calculate similarity between a3 and a5

$App_3 = \{a_3\}$

$u_1 = \{a_3, a_5, a_7, a_{11}\}$
$Similarity(App_3, u_1) = 0.50$

$u_2 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$
$Similarity(App_3, u_2) = 0.33$

$u_8 = \{a_1, a_3, a_4, a_5, a_{16}, a_{37}, a_{44}\}$
$Similarity(App_3, u_2) = 0.38$

$u_9 = \{a_3, a_7, a_{14}, a_{44}, a_{52}\}$
$Similarity(App_3, u_2) = 0.45$

$Similarity(a_3, a_5)$
$= Similarity(App_3, u_1) + Similarity(App_3, u_2) + Similarity(App_3, u_8) = 0.5 + 0.33 + 0.38 = 1.21$

Find a3's top-5 related Apps by similarity

$App_3 = \{a_3\}$

$u_1 = \{a_3, a_5, a_7, a_{11}\}$
$Similarity(App_3, u_1) = 0.50$

$u_2 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$
$Similarity(App_3, u_2) = 0.33$

$u_8 = \{a_1, a_3, a_4, a_5, a_{16}, a_{37}, a_{44}\}$
$Similarity(App_3, u_2) = 0.38$

$u_9 = \{a_3, a_7, a_{14}, a_{44}, a_{52}\}$
$Similarity(App_3, u_2) = 0.45$

$Similarity(a_3, a_1) = 0.71$
$(a_3, a_2) = 0.33$
$(a_3, a_4) = 1.21$
$(a_3, a_5) = 1.21$
$(a_3, a_6) = 0.33$
$(a_3, a_7) = 1.28$
$(a_3, a_8) = 0.33$
$(a_3, a_9) = 0.33$
$(a_3, a_{11}) = 0.50$
$(a_3, a_{14}) = 0.45$
$(a_3, a_{16}) = 0.38$
$(a_3, a_{37}) = 0.38$
$(a_3, a_{44}) = 0.83$
$(a_3, a_{52}) = 0.45$

Output: $\{a_7, a_4, a_5, a_{44}, a_1\}$

Code - calculate Apps' top-5 related Apps

```
42  def main():
43      try:
44          # get MongoDB client and set it in DataService
45          client = MongoClient('localhost', 27017)
46          DataService.init(client)
47
48          # work flow
49          user_download_history = DataService.retrieve_user_download_history()
50          calculate_top_5('C10107104', user_download_history.values())
51      except Exception as e:
52          print(e)
53      finally:
54          # clean up work
55          if 'client' in locals():
56              client.close()
57
58  if __name__ == "__main__":
59      main()
60
```

Code - calculate Apps' top-5 related Apps

```python
from pymongo import MongoClient
import random


class DataService(object):

    @classmethod
    def init(cls, client):
        cls.client = client
        cls.db = client.appstore
        cls.user_download_history = cls.db.user_download_history
        cls.app_info = cls.db.app_info

    @classmethod
    def retrieve_user_download_history(cls, filter_dict={}):
        # return a dict {user_id: download_history} containing user download history data
        # return all data in the collection if no filter is specified
        result = {}
        cursor = cls.user_download_history.find(filter_dict)
        for user_download_history in cursor:
            result[user_download_history['user_id']] = user_download_history['download_history']
        return result
```

Code - calculate Apps' top-5 related Apps

```python
from pymongo import MongoClient
from dataservice import DataService
import operator
import math

class Helper(object):
    @classmethod
    def cosine_similarity(cls, app_list1, app_list2):
        return float(cls.__count_match(app_list1, app_list2)) / math.sqrt( len(app_list1) * len(app_list2) )

    @classmethod
    def __count_match(cls, list1, list2):
        count = 0
        for element in list1:
            if element in list2:
                count += 1
        return count

def calculate_top_5(app, user_download_history):
    # create a dict to store each other app and its similarity to this app
    app_similarity = {} # {app_id: similarity}

    for apps in user_download_history:
        #calculate the similarity
        similarity = Helper.cosine_similarity([app], apps)
        for other_app in apps:
            if app_similarity.has_key(other_app):
                app_similarity[other_app] = app_similarity[other_app] + similarity
            else:
                app_similarity[other_app] = similarity

    # There could be app without related apps (not in any download history)
    if not app_similarity.has_key(app):
        return

    # sort app_similarity dict by value and get the top 5 as recommendation
    app_similarity.pop(app)
    sorted_tups = sorted(app_similarity.items(), key=operator.itemgetter(1), reverse=True) # sort by similarity
    top_5_app = [sorted_tups[0][0], sorted_tups[1][0], sorted_tups[2][0], sorted_tups[3][0], sorted_tups[4][0]]
    print("top_5_app for " + str(app) + ":\t" + str(top_5_app))
```

Result

Console output (Sublime Text 3)

top_5_app for C10107104: [u'C10129690', u'C5341', u'C20252', u'C10191382', u'C183901']

[Finished in 0.7s]

Outline

- Load data into MongoDB
- Calculate an App's top-5 related apps
- Save data into MongoDB
  - Instead of print, we do persist
  - Persist top 5 recommended apps in app_info

Code – Persist Top 5 Apps in MongoDB

```
19  def calculate_top_5(app, user_download_history):
20      # create a dict to store each other app and its similarity to this app
21      app_similarity = {} # {app_id: similarity}
22
23      for apps in user_download_history:
24          #calculate the similarity
25          similarity = Helper.cosine_similarity([app], apps)
26          for other_app in apps:
27              if app_similarity.has_key(other_app):
28                  app_similarity[other_app] = app_similarity[other_app] + similarity
29              else:
30                  app_similarity[other_app] = similarity
31
32      # There could be app without related apps (not in any download history)
33      if not app_similarity.has_key(app):
34          return
35
36      # sort app_similarity dict by value and get the top 5 as recommendation
37      app_similarity.pop(app)
38      sorted_tups = sorted(app_similarity.items(), key=operator.itemgetter(1), reverse=True) # sort by similarity
39      top_5_app = [sorted_tups[0][0], sorted_tups[1][0], sorted_tups[2][0], sorted_tups[3][0], sorted_tups[4][0]]
40      # print("top_5_app for " + str(app) + ":\t" + str(top_5_app))
41
42      # store the top 5
43      DataService.update_app_info({'app_id': app}, {'$set': {'top_5_app': top_5_app}})
```

Code – Persist Top 5 Apps in MongoDB

```
3   class DataService(object):
4
5       @classmethod
6       def init(cls, client):
7           cls.client = client
8           cls.db = client.appstore
9           cls.user_download_history = cls.db.user_download_history
10          cls.app_info = cls.db.app_info
11
12      @classmethod
13      def retrieve_user_download_history(cls, filter_dict={}):
14          # return a dict {user_id: download_history} containing user download history data
15          # return all data in the collection if no filter is specified
16          result = {}
17          cursor = cls.user_download_history.find(filter_dict)
18          for user_download_history in cursor:
19              result[user_download_history['user_id']] = user_download_history['download_history']
20          return result
21
22      @classmethod
23      def update_app_info(cls, filter_dict, update):
24          cls.app_info.update_one(filter_dict, update, True)
```

Persist Top 5 Apps in MongoDB

To check the DB update

- Enter MongoDB console
  - Query like other database
  - Example given in next page

Result

```
C:\Users\lugua_000>mongo
MongoDB shell version: 3.2.0
connecting to: test
> use appstore
switched to db appstore
> db.app_info.findOne({"app_id":"C10063783"})
{
        "_id" : ObjectId("5689979cbcb96fc5c0d63d74"),
        "score" : "8",
        "title" : "3D终极狂飙3",
        "url" : "http://appstore.huawei.com:80/app/C10063783",
        "app_id" : "C10063783",
        "thumbnail_url" : "http://appimg.hicloud.com/hwmarket/files/application/icon144/b03f7e1fa3b04fb089fdfb3c0da88010.png",
        "intro" : "《3D终极狂飙3》第三代华丽来袭，强势登陆！ ☆全球5000万狂热玩家翘首以待！堪称移动平台赛车游戏的霸主！ ☆延续超小的体积和炫酷的3D引擎！加入360度的3D真实名车体验极度疯狂！ ☆融合了流行而火爆的改装车竞速文化，满足玩家个性需求！ ☆真实的漂移、加速特效和动态模糊，逼真的画质，让您身临其境 ☆标准、挑战、生存等赛车模式",
        "developer" : "北京中科奥科技有限公司",
        "top_5_app" : [
                "C10179074",
                "C10136197",
                "C10203802",
                "C10221865",
                "C10281290"
        ]
}
```

Persist Top 5 Apps in MongoDB

```
> mongo
> use appstore
> db.app_info.findOne({"app_id": "C10107104"})
{
    "_id" : ObjectId("568794f1b44031e7d3adbd6b"),
    "score" : "7",
    "title" : "新华字典",
    "url" : "http://appstore.huawei.com:80/app/C10107104",
    "app_id" : "C10107104",
    "thumbnail_url" : "http://appimg.hicloud.com/hwmarket/files/application/icon144/1d538d5861964bcaa136cc41d359a53a.png",
    "intro" : "免流量、免下载、免扩展，更新到20998个汉字，安卓最完美的新华字典！ 1、自带真人发音，无需再次下载发音补丁 包。 2、支持汉字、拼音、部首、五笔搜寻方式，例如：想要使用部首搜索，用手机手写功能输入部首即可查询。 3、根据《新华字典》目录增加了“汉语拼音检索”、“汉语拼音方案”、“汉语简繁对照表”和“标点符号用法”；4、同时支持9大网络知识库查询； 5、",
    "developer" : "广东凡跃计算机系统股份有限公司",
    "top_5_app" : [
        "C10129690",
        "C5341",
        "C20252",
        "C10191382",
        "C183901"
    ]
}
```

We have got top 5 apps for one app, need to do it for all apps

Persist Top 5 Apps in MongoDB

Next Step: replicate the work for all apps

- Loop through all apps, get the list from collection app_info
  - Add function in DataService
  - Add for loop in main function
- Call the same function calculate_top_5(app, user_download_history)

Code - Persist Top 5 Apps in MongoDB

```
45  def main():
46      try:
47          # get MongoDB client and set it in DataService
48          client = MongoClient('localhost', 27017)
49          DataService.init(client)
50
51          # work flow
52          user_download_history = DataService.retrieve_user_download_history()
53          app_info = DataService.retrieve_app_info()
54          for app in app_info.keys():
55              calculate_top_5(app, user_download_history.values())
56      except Exception as e:
57          print("Exception detected:")
58          print(e)
59      finally:
60          # clean up work
61          if 'client' in locals():
62              client.close()
63
64  if __name__ == "__main__":
65      main()
```

Code - Persist Top 5 Apps in MongoDB

```python
from pymongo import MongoClient

class DataService(object):

    @classmethod
    def init(cls, client):
        cls.client = client
        cls.db = client.appstore
        cls.user_download_history = cls.db.user_download_history
        cls.app_info = cls.db.app_info

    @classmethod
    def retrieve_user_download_history(cls, filter_dict={}):
        # return a dict {user_id: download_history} containing user download history data
        # return all data in the collection if no filter is specified
        result = {}
        cursor = cls.user_download_history.find(filter_dict)
        for user_download_history in cursor:
            result[user_download_history['user_id']] = user_download_history['download_history']
        return result

    @classmethod
    def retrieve_app_info(cls, filter_dict={}):
        # return a dict {app_id: {app_name, top_5_app}} containing user download history data
        # return all data in the collection if no filter is specified
        result = {}
        cursor = cls.app_info.find(filter_dict)
        for app_info in cursor:
            app_id = app_info['app_id']
            title = app_info['title']
            result[app_id] = {'title': title}
        return result

    @classmethod
    def update_app_info(cls, filter_dict, update):
        cls.app_info.update_one(filter_dict, update, True)
```

Reference & Read more

- http://api.mongodb.org/python/current/tutorial.html
- https://docs.mongodb.org/getting-started/python/
- https://en.wikipedia.org/wiki/Cosine_similarity

Homework

1. For each user in our dataset, calculate his/her top-5 recommended apps
2. Try to gain better performance
   - Evaluate the performance by time analysis
   - 200s? 100s? 20s? 10s? 1s?

```
1  import time
2
3  start = time.clock() # start time of processor time
4
5  # do some work here
6
7  end = time.clock() # end time of processor time
8  print "time elapsed = " + str(end - start)
```

Result

```
C:\Users\lugua_000>mongo
MongoDB shell version: 3.2.0
connecting to: test
> use appstore
switched to db appstore
> db.app_info.findOne()
{
        "_id" : ObjectId("5689979cbcb96fc5c0d63d74"),
        "score" : "8",
        "title" : "3D终极狂飙3",
        "url" : "http://appstore.huawei.com:80/app/C10063783",
        "app_id" : "C10063783",
        "thumbnail_url" : "http://appimg.hicloud.com/hwmarket/files/application/icon144/b03f7e1fa3b04fb089fdfb3c0da88010.png",
        "intro" : "《3D终极狂飙3》第三代华丽来袭，强势登陆！ ☆全球5000万狂热玩家翘首以待！堪称移动平台赛车游戏的霸主！ ☆延续超小的体积和炫酷的3D引擎！加入360度的3D真实名车体验极度疯狂！ ☆融合了流行而火爆的改装车竞速文化，满足玩家个性需求！ ☆真实的漂移、加速特效和动态模糊，逼真的画质，让您身临其境 ☆标准、挑战、生存等赛车模式",
        "developer" : "北京中科奥科技有限公司",
        "top_5_app" : [
                "C10179074",
                "C10136197",
                "C10203802",
                "C10221865",
                "C10281290"
        ]
}
```

User App Recommendations

- How we calculate the app recommendations for an app
  - Top 5 result
  - From app to app similarity
  - From app to download history similarity
- App recommendations for a user is similar
  - Top 5 result
  - From user download history (app list) to app similarity
  - From user download history to user download history similarity

Better Performance

- Improve on Logic
  - Simply write better code
  - Only get information we need from DB
    - Only apps within user download histories have similarity
  - Only process useful information
    - Don’t calculate app similarity to itself then pop
- Multi-threading
  - thread module -> old
  - threading module -> new

thread Module

```
def calculate_and_persist_top_5_app_single_thread(self):
    # find the 5 top recommended app for each app and persist them in app_info
    print "calculate_and_persist_top_5_app_single_thread()"
    start = time.clock() # start time of processor time

    download_history = self.dict_user_download_history.values()
    for app in self.dict_app_info.keys():
        task = self.create_task(app, download_history)
        task.run()

    end = time.clock() # end time of processor time
    print "time elapsed = " + str(end - start)

def calculate_and_persist_top_5_app_multi_thread(self):
    # find the 5 top recommended app for each app and persist them in app_info
    print "calculate_and_persist_top_5_app_multi_thread()"
    start = time.clock() # start time of processor time

    download_history = self.dict_user_download_history.values()
    for app in self.dict_app_info.keys():
        task = self.create_task(app, download_history)
        thread.start_new_thread(task.run, ())

    end = time.clock() # end time of processor time
    print "time elapsed = " + str(end - start)
```

thread Module

- import thread
- Caveat: Not waiting for each thread to finish before exiting the program

threading Module

```
import threading
import time

class myThread (threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        print "Starting " + self.name
        # Get lock to synchronize threads
        threadLock.acquire()
        print_time(self.name, self.counter, 3)
        # Free lock to release next thread
        threadLock.release()

def print_time(threadName, delay, counter):
    while counter:
        time.sleep(delay)
        print "%s: %s" % (threadName, time.ctime(time.time()))
        counter -= 1

threadLock = threading.Lock()
threads = []

# Create new threads
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 2)

# Start new Threads
thread1.start()
thread2.start()

# Add threads to thread list
threads.append(thread1)
threads.append(thread2)

# Wait for all threads to complete
for t in threads:
    t.join()
print "Exiting Main Thread"
```

CPython Multi-threading

- Multiple threads will not actually run concurrently due to Python's Global Interpreter Lock (GIL)
- The GIL is an interpreter-level lock. This lock prevents execution of multiple threads at once in the Python interpreter. Each thread that wants to run must wait for the GIL to be released by the other thread, which means your multi-threaded Python application is essentially single threaded.
- It means you are using multiple threads but never at the same time.

Python Multi-threading in General

- The GIL only affects threads within a single process.
- GIL is a limitation of the implementation (CPython) and not of Python in general, it's possible to implement Python without this limitation.
- GIL-less and Automatic Mutual Exclusion (AME)
- See PyPy as an example

Reference & Read more

- https://en.wikipedia.org/wiki/Global_interpreter_lock
- https://wiki.python.org/moin/GlobalInterpreterLock
- https://en.wikipedia.org/wiki/Python_(programming_language)
- https://en.wikipedia.org/wiki/PyPy
- https://www.google.com/