In [ ]:

```
### 推荐服务
'''
- 离线推荐
  - 先召回对召回结果排序
  - 为每一个用户都进行召回并排序的过程并且把拍好顺序的结果放到数据库中
  - 如果需要推荐结果的时候 直接到数据库中按照user_id查询，返回推荐结果
  - 优点 结构比较简单 推荐服务只需要不断计算，把结果保存到数据库中即可
  - 缺点 实时性查 如果数据1天不更新 1天之内推荐结果一样的，不能反映用户的实时兴趣
- 实时推荐
  - 排序的模型加载好
  - 召回阶段的结果缓存
  - 所有用户的特征缓存
  - 所有物品的特征缓存
  - 把推荐的服务暴露出去（django flask）需要推荐结果的服务把 用户id 传递过来
    - 根据id 找到召回结果
    - 根据id 找到缓存的用户特征
    - 根据召回结果的物品id 找到物品的特征
    - 用户特征+物品特征-》逻辑回归模型 就可以预测点击率
    - 所有召回的物品的点记率都预测并排序 推荐topN
    - 实时通过LR模型进行排序的好处
      - 随时修改召回集
      - 随时调整用户的特征
      - 当用户需要推荐服务的时候，获取到最新的召回集和用户特征 得到最新的排序结果 更能
'''
```

离线数据缓存之离线召回集

这里主要是利用我们前面训练的ALS模型进行协同过滤召回，但是注意，我们ALS模型召回的是用户最感兴趣的类别，而我们需要的是用户可能感兴趣的广告的集合，因此我们还需要根据召回的类别匹配出对应的广告。

所以这里我们除了需要我们训练的ALS模型以外，还需要有一个广告和类别的对应关系。

过程与目的

用户—3个最感兴趣的商品类别-（在这3个类别中，）随机找到500个商品推荐给该用户

选取商品的原则：

最感兴趣的类别中如果够500个，就不需要其他类别的商品；如果不够就继续在次感兴趣的类别中补充，直到总共选出500个商品

In [1]:
```
import os
# 配置pyspark和spark driver运行时 使用的python解释器
JAVA_HOME = '/root/bigdata/jdk'
PYSPARK_PYTHON = '/miniconda2/envs/py365/bin/python'
# 当存在多个版本时，不指定很可能会导致出错
os.environ['PYSPARK_PYTHON'] = PYSPARK_PYTHON
os.environ['PYSPARK_DRIVER_PYTHON'] = PYSPARK_PYTHON
os.environ['JAVA_HOME'] = JAVA_HOME
# 配置spark信息
from pyspark import SparkConf
from pyspark.sql import SparkSession

SPARK_APP_NAME = 'recallAdSets'
SPARK_URL = 'spark://192.168.58.100:7077'

conf = SparkConf()
config = (
    ('spark.app.name',SPARK_APP_NAME),
    ('spark.executor.memory','2g'),
    ('spark.master',SPARK_URL),
    ('spark.executor.cores','2')
)
conf.setAll(config)

spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

离线推荐数据缓存

In [2]:
```
df = spark.read.csv('/data/ad_feature.csv',header=True)
df.printSchema()
df.show()
```

```
root
 |-- adgroup_id: string (nullable = true)
 |-- cate_id: string (nullable = true)
 |-- campaign_id: string (nullable = true)
 |-- customer: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- price: string (nullable = true)

+----------+-------+-----------+--------+------+-----+
|adgroup_id|cate_id|campaign_id|customer| brand|price|
+----------+-------+-----------+--------+------+-----+
|     63133|   6406|      83237|       1| 95471|170.0|
|    313401|   6406|      83237|       1| 87331|199.0|
|    248909|    392|      83237|       1| 32233| 38.0|
|    208458|    392|      83237|       1|174374|139.0|
|    110847|   7211|     135256|       2|145952|32.99|
|    607788|   6261|     387991|       6|207800|199.0|
|    375706|   4520|     387991|       6|  NULL| 99.0|
|     11115|   7213|     139747|       9|186847| 33.0|
|     24484|   7207|     139744|       9|186847| 19.0|
|     28589|   5953|     395195|      13|  NULL|428.0|
|     23236|   5953|     395195|      13|  NULL|368.0|
|    300556|   5953|     395195|      13|  NULL|639.0|
|     92560|   5953|     395195|      13|  NULL|368.0|
|    590965|   4284|      28145|      14|454237|249.0|
|    529913|   4284|      70206|      14|  NULL|249.0|
|    546930|   4284|      28145|      14|  NULL|249.0|
|    639794|   6261|      70206|      14| 37004| 89.9|
|    335413|   4284|      28145|      14|  NULL|249.0|
|    794890|   4284|      70206|      14|454237|249.0|
|    684020|   6261|      70206|      14| 37004| 99.0|
+----------+-------+-----------+--------+------+-----+
only showing top 20 rows
```

In [3]:
```
from pyspark.sql.types import IntegerType, FloatType
ad_feature_df = df.\
    withColumn("adgroup_id", df.adgroup_id.cast(IntegerType())).withColumnRenamed("adg
    withColumn("cate_id", df.cate_id.cast(IntegerType())).withColumnRenamed("cate_id",
    withColumn("campaign_id", df.campaign_id.cast(IntegerType())).withColumnRenamed("c
    withColumn("customer", df.customer.cast(IntegerType())).withColumnRenamed("custome
    withColumn("brand", df.brand.cast(IntegerType())).withColumnRenamed("brand", "bran
    withColumn("price", df.price.cast(FloatType()))
ad_feature_df.show()
```

```
+---------+------+----------+----------+-------+-----+
|adgroupId|cateId|campaignId|customerId|brandId|price|
+---------+------+----------+----------+-------+-----+
|    63133|  6406|     83237|         1|  95471|170.0|
|   313401|  6406|     83237|         1|  87331|199.0|
|   248909|   392|     83237|         1|  32233| 38.0|
|   208458|   392|     83237|         1| 174374|139.0|
|   110847|  7211|    135256|         2| 145952|32.99|
|   607788|  6261|    387991|         6| 207800|199.0|
|   375706|  4520|    387991|         6|   null| 99.0|
|    11115|  7213|    139747|         9| 186847| 33.0|
|    24484|  7207|    139744|         9| 186847| 19.0|
|    28589|  5953|    395195|        13|   null|428.0|
|    23236|  5953|    395195|        13|   null|368.0|
|   300556|  5953|    395195|        13|   null|639.0|
|    92560|  5953|    395195|        13|   null|368.0|
|   590965|  4284|     28145|        14| 454237|249.0|
|   529913|  4284|     70206|        14|   null|249.0|
|   546930|  4284|     28145|        14|   null|249.0|
|   639794|  6261|     70206|        14|  37004| 89.9|
|   335413|  4284|     28145|        14|   null|249.0|
|   794890|  4284|     70206|        14| 454237|249.0|
|   684020|  6261|     70206|        14|  37004| 99.0|
+---------+------+----------+----------+-------+-----+
only showing top 20 rows
```

In [4]:
```
_ = ad_feature_df.select('adgroupId','cateId')
# 由于这里数据集其实很少，所以我们再直接转成Pandas dataframe来处理，把数据载入内存
pdf = _.toPandas()
# 手动释放一些内存
del df
del ad_feature_df
del _
import gc
gc.collect()
```

Out[4]: 38

In [5]:
```
1 pdf
```

Out[5]:

	adgroupId	cateId
0	63133	6406
1	313401	6406
2	248909	392
3	208458	392
4	110847	7211
...	...	...
846806	824255	4526
846807	790170	4280
846808	845286	6261
846809	824732	4520
846810	845337	11156

846811 rows × 2 columns

In [6]:
```
# 根据指定的类别找到对应的广告
import numpy as np
import  pandas as  pd
pdf.where(pdf.cateId==11156).dropna().adgroupId
np.random.choice(pdf.where(pdf.cateId==11156).dropna().adgroupId.astype(np.int64),200
```

Out[6]:
```
array([350371,  46698, 456726, 463475, 610540, 553547, 298413, 305413,
       547627, 214219, 334026,  58220, 394104, 417290, 777147, 255812,
       247523, 481589, 269520, 652149, 343454, 241178, 570494, 427537,
       841627, 318839, 502409, 813551, 735150, 404787, 210049, 202896,
       324996, 235106, 293771, 548632,  99717, 318769, 238093,  51054,
       262506, 318839, 747213, 456353, 212226, 691180, 766358, 196508,
       373198, 151999, 133970, 231001, 675002, 573206, 773529, 352273,
       115117, 244105, 621927, 363155, 670031, 422430, 447492,  85423,
        61255,  71053, 610549, 691915, 121742, 290666,  47556,   8927,
       293771,   9933, 674882, 456726, 752187, 279847, 494730, 207350,
       110509, 534658, 118936, 579680, 205668, 471508, 339134, 231234,
       664607, 552853, 766865, 331045, 845337,   8128, 634766, 580399,
       688335, 403185, 479329, 364290, 691915, 706581, 644401, 749196,
       683715, 151999, 562147, 262504, 404733, 806260, 196359,  84482,
       299053, 467495, 403406, 377407,  66710, 506028, 514192, 115117,
       479517, 123884, 484455, 600931, 733900, 277013, 384104, 634766,
       234129, 691249, 466721, 413610, 280681, 384104, 361748, 234129,
       577891,  85647,  10617,  69529, 618012, 551452, 280630, 173950,
       364325,  13858, 316322,  55278, 610614, 547627, 561338, 430104,
       110881, 634766, 683560, 121517, 173414, 519218, 592961, 261067,
       126708, 418447, 494193, 706818, 297172, 291832, 661774, 260764,
       173950, 593768,  87118,  79165, 456726, 160709, 813551, 202738,
       436001, 523136, 660174, 765440, 207498, 664445, 547627, 295645,
       662714,  97235,  69529, 707612, 401092, 230645, 668118, 348378,
       668118, 639007, 618101, 382873, 698267, 280630, 607661,  62630])
```

In [7]:

```python
# 利用ALS模型进行类别的召回

# 加载als模型，注意必须先有spark上下文管理器，即sparkContext，但这里sparkSession创建后

from pyspark.ml.recommendation import ALSModel
als_model = ALSModel.load('/models/userCateRatingALSModel.obj')
# 返回模型中关于用户的所有属性   df:   id   features
# features是隐因子
als_model.userFactors.show(truncate=False)
```

```
+---+-----------------------------------------------------------------------------
--------------------------------------------+
|id |features
|
+---+-----------------------------------------------------------------------------
--------------------------------------------+
|8  |[0.34110302, 0.5864484, 0.5938051, -0.11306913, 1.5407808, -0.96665496, -0.20389
616, -0.35854334, 0.26578107, 0.7115058]          |
|18 |[0.26234755, 0.05093716, 0.6261596, -1.2136128, -0.098772116, -0.48658535, 0.189
82896, -0.19280958, -0.002499504, 0.48769426] |
|28 |[-0.42565244, 0.5686206, 0.009474004, -0.5029053, -0.094820365, 0.057066303, 0.2
1090633, -0.17339933, -0.344597, 0.13138233]   |
|38 |[-0.17222986, 0.10863494, 0.021417063, -0.624083, 0.048439596, 0.43609896, 0.568
92, -0.057229422, 0.20223776, -0.12358887]        |
|48 |[-0.7299992, 0.038008712, 0.73940223, -0.6590781, 1.2926366, -0.71040106, -0.235
03205, -0.29304594, 0.47312748, 0.52401793]     |
|58 |[0.20949, 0.7887514, -0.6033555, -0.5711672, 0.23008735, 0.29748958, 0.5820638,
0.5619163, 0.40513405, 0.17732376]                |
|68 |[-2.1288443, 0.13392396, -0.3612919, 0.67474073, -0.7158458, 1.5407596, 0.612886
85, -0.35098934, -0.15001762, -0.064970486]     |
|78 |[0.20121495, 0.6876173, 0.3622489, -0.84896266, -0.2793664, -0.31254628, -0.2689
6414, -0.47881302, -0.044694267, 0.639279]       |
|88 |[0.57730436, 0.16511923, -0.86218643, -0.47414196, -0.17603633, 0.6621098, 1.272
7635, -0.49922916, 1.0252689, 1.3391578]          |
|98 |[0.22689435, 0.8057461, 0.0498942, -0.26441857, 0.5764083, -0.025070967, 0.66579
807, -0.064371265, 0.33619398, -0.24704869]     |
|108|[-0.32026184, 0.123412564, -0.4155425, -0.5243413, 0.69900405, 1.2177451, 0.2137
7827, 0.30828297, -0.10080125, 0.53502196]       |
|118|[-0.06577069, 0.6053421, 0.31097224, -0.060283836, 0.12539195, 0.3317593, 1.0821
071, 0.40615815, 0.09566138, 0.022687059]        |
|128|[0.3708652, 0.523741, -0.0067040836, -0.47812364, -0.9798019, 0.2644588, 0.03226
3972, -0.36943913, -0.3558998, 0.50211006]       |
|138|[-0.31295472, -0.29118866, 0.047263842, -0.60274184, 0.46988657, 0.78914577, 0.1
8607041, 0.037370276, 0.03699361, 0.45439047] |
|148|[-0.032690633, 0.6432083, 0.35597494, -0.12200055, -1.2113955, -0.86280704, 0.50
80956, -0.13657402, 0.5111354, -0.04310837]     |
|158|[-0.05601896, 0.039884217, 0.15044844, -0.3582624, 0.026798097, 0.08856124, 0.35
76647, 0.023552353, -0.13243553, -0.043265775]|
|168|[0.58979905, 0.16813694, 0.9276362, -0.34517348, 0.30036208, 0.28011554, 0.03956
1532, -0.37168157, 0.089810126, 0.630418]        |
|178|[0.14070545, 0.8080799, 1.2514955, -1.3344088, -0.10965286, 0.95266587, 1.479752
5, -0.47243184, 0.6205894, -0.23914866]           |
|188|[0.17815629, 0.5967054, -0.40372986, -0.3374032, -0.33554748, 0.36597526, -0.049
79073, 0.15350537, -0.38125184, 0.3769183]       |
|198|[-0.34739748, 1.6911665, 1.2147195, -1.3682649, 0.28932884, -0.12586524, 0.96803
933, -0.22250393, -0.15037307, -0.75448304]     |
+---+-----------------------------------------------------------------------------
```

```
+-----------------------------------------------+
only showing top 20 rows
```

In [8]:
```
cateId_df = pd.DataFrame(sorted(pdf.cateId.unique()),columns=['cateId'])
# 上行可改为 cateId_df = pd.DataFrame(np.array(list(set(pdf.cateId))).reshape(6769,1),
cateId_df
# list(set([9,6,7,1,10])) -> [1, 6, 7, 9, 10], set之后变成有序了
```

Out[8]:

	cateId
0	1
1	2
2	3
3	4
4	5
...	...
6764	12919
6765	12947
6766	12948
6767	12955
6768	12960

6769 rows × 1 columns

In [9]:
```
# 插入一列
# insert(loc) loc=0 userid在左边；loc=1 userid在右边
cateId_df.insert(0,'userId',np.array([8 for i in range(6769)]))
cateId_df
```

Out[9]:

	userId	cateId
0	8	1
1	8	2
2	8	3
3	8	4
4	8	5
...	...	...
6764	8	12919
6765	8	12947
6766	8	12948
6767	8	12955
6768	8	12960

6769 rows × 2 columns

In [10]:
```
1 spark.createDataFrame(cateId_df).show()
```

```
+------+------+
|userId|cateId|
+------+------+
|     8|     1|
|     8|     2|
|     8|     3|
|     8|     4|
|     8|     5|
|     8|     6|
|     8|     7|
|     8|     8|
|     8|     9|
|     8|    10|
|     8|    11|
|     8|    12|
|     8|    13|
|     8|    15|
|     8|    16|
|     8|    17|
|     8|    18|
|     8|    19|
|     8|    21|
|     8|    22|
+------+------+
only showing top 20 rows
```

In [11]:
```
# 传入 userid、cataId的df，对应预测值进行排序
als_model.transform(spark.createDataFrame(cateId_df)).sort('prediction',ascending=Fal
```

```
+------+------+----------+
|userId|cateId|prediction|
+------+------+----------+
|     8|  8882| 7.0304847|
|     8|  5993|  5.751598|
|     8| 12282|  5.327876|
|     8|  6734|    5.2168|
|     8|  6421| 5.1464972|
|     8|  1155|  4.995791|
|     8|   856|  4.618875|
|     8|  6261| 4.5706525|
|     8|  2767| 4.5490074|
|     8|  1101| 4.5368094|
|     8|  7752| 4.2273746|
|     8| 11752| 4.1914773|
|     8|  2468| 4.1329217|
|     8|  6617| 3.9452367|
|     8|  7214| 3.8621702|
|     8|  8340| 3.7554228|
|     8|  8348| 3.6372938|
|     8|  1157| 3.6295743|
|     8|   841|  3.499203|
|     8|   184| 3.3779442|
+------+------+----------+
only showing top 20 rows
```

In [12]:

```
#============================往redis里面写数据：113万用户*500，耗时长估计3-4h=========
import numpy as np
import pandas as pd

import redis
# 为每个用户（共6769个用户）选出500个物品(adgroupId)
# 选物品的原则：先从该用户（预测）评分最高的物品种类(cateId)选500个物品（adgroupId），
# 不够就在下一个物品种类（adgroupId）中选物品（cateId），直到选出500个物品（cateId），
# 存储用户召回，使用redis第9号数据库，类型：sets类型
client = redis.StrictRedis(host="192.168.58.100", port=6379, db=9)

for r in als_model.userFactors.select("id").collect():

    userId = r.id

    cateId_df = pd.DataFrame(pdf.cateId.unique(),columns=["cateId"])
#     cateId_df = pd.DataFrame(np.array(list(set(pdf.cateId))).reshape(6769,1), columns=["cateId"])
    cateId_df.insert(0, "userId", np.array([userId for i in range(6769)]))
    ret = set()

    # 利用模型，传入datasets(userId, cateId)，这里控制了userId一样，所以相当于是在求某
    # 按照 对物品种类感兴趣的程度 为物品排序
    cateId_list = als_model.transform(spark.createDataFrame(cateId_df)).sort("predict
    # 从前20个感兴趣的种类中选出500个进行召回
    for i in cateId_list.head(20):
        need = 500 - len(ret)     # 如果不足500个，那么随机选出need个广告
        ret = ret.union(np.random.choice(pdf.where(pdf.cateId==i.cateId).adgroupId.dr
        if len(ret) >= 500:     # 如果达到500个则退出
            break
    client.sadd(userId, *ret)

# 如果redis所在机器，内存不足，会抛出异常
```

...