《大数据技术与应用》课程设计

Steam 商店游戏评测分析

小　组

40230212　潘　昆

40230220　宋焱彪

40230221　陈　俊

课设源码：https://github.com/520badgateway/BigData40230212

一、课设介绍

本次课设是 Ubuntu 系统上 Hadoop 框架下基于 Steam 商店的玩家评测数据进行的大数据分析项目。

项目内容包括：

1. Python 爬虫获取数据集

2. Hive 处理与初步分析

3. MapReduce 评论词语共现分析

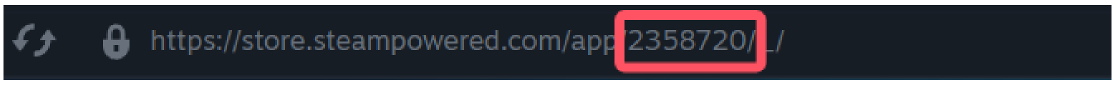
4. Spark 跑一个评论分类模型

二、实验内容

P.S. 所有用到的 python 库，如果没有都使用 pip 及时安装了

1. Python 爬虫获取数据集

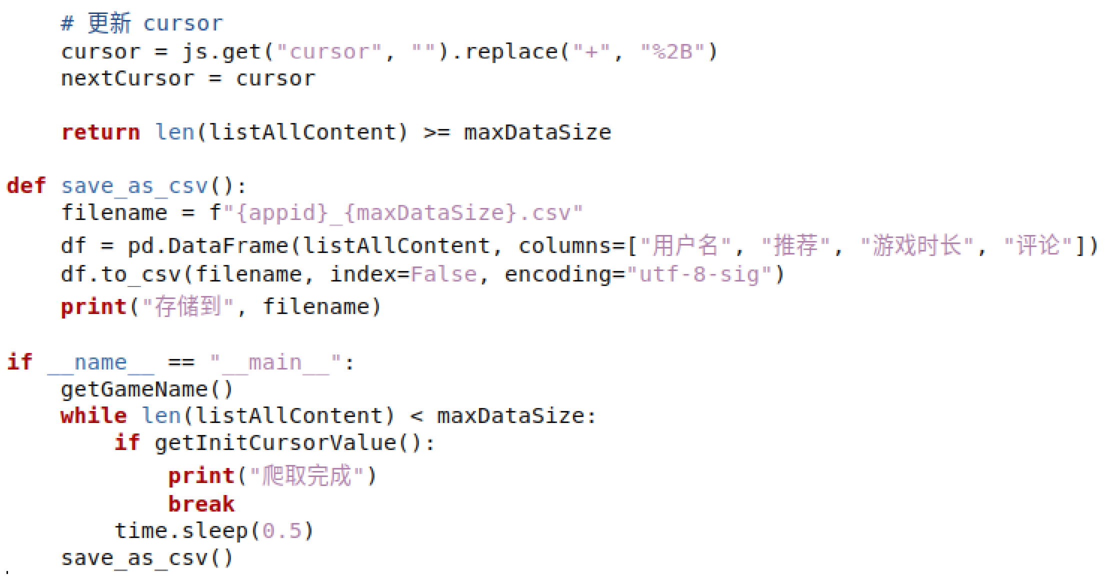
Steam 每个游戏在商店页面都有对应 ID，也可以请求 Steam 的 API 来输入游戏名映射为 ID，我们直接就在商店页面查找。


https://store.steampowered.com/app/2358720/_/

使用 Python 实现爬虫：

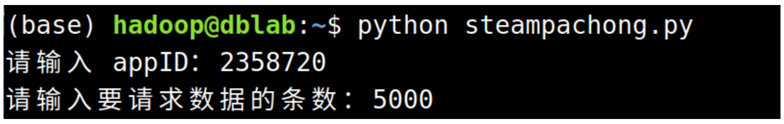
```
# steampachong.py
# -*- coding: utf-8 -*-
"""
用途:
从 Steam 商店爬取指定游戏的评论（简体中文），并保存为 csv 文件。

作者: 40230212小组
日期: 2025-06-19
"""

import requests
import time
import re
import pandas as pd
from bs4 import BeautifulSoup

# 全局配置
appid = input("请输入 appID: ").strip()
maxDataSize = int(input("请输入要请求数据的条数: ").strip())  # 10的倍数
appName = "Unknown"
listAllContent = []         # 所有评论
nextCursor = "*"            # Steam 翻页 cursor
reloadDataNum = 0           # 请求失败重试计数
reloadDataNumMax = 5        # 最大重试次数
lastedNum = 0               # 上次数据量
totalEndNum = 0             # 连续无新数据计数

# 去掉多余空白
def clean_text(text: str) -> str:
    return re.sub(r'\s+', ' ', text).strip()
```

```python
# 获取并清洗游戏名称
def getGameName():
    global appName
    try:
        resp = requests.get(f"https://store.steampowered.com/app/{appid}", timeout=10)
        resp.raise_for_status()
    except Exception as e:
        print("获取游戏名字失败: ", e)
        return

    soup = BeautifulSoup(resp.text, "html.parser")
    span = soup.find("span", itemprop="name")
    if span and span.string:
        name = span.string
        # 把文件名非法字符替换为下划线
        appName = re.sub(r'[:\\/*?"<>|]', '_', name)
        print("游戏名字:", appName)
    else:
        print("未找到游戏名字:", appName)
    time.sleep(1)

# 拉取一页评论
def getInitCursorValue() -> bool:
    global reloadDataNum, nextCursor, lastedNum, totalEndNum

    url = (
        f"https://store.steampowered.com/appreviews/{appid}"
        f"?cursor={nextCursor}&language=schinese&day_range=365"
        "&review_type=all&purchase_type=all&filter=recent"
    )

    try:
        js = requests.get(url, timeout=10).json()
    except Exception:
        reloadDataNum += 1
        print(f"请求评论失败, 尝试重新拉取...{reloadDataNum}")
        time.sleep(1)
        return reloadDataNum >= reloadDataNumMax

    reloadDataNum = 0


    # 解析 HTML
    soup = BeautifulSoup(js.get("html", ""), "html.parser")
    names = [a.string for d in soup.select("div.persona_name") for a in d.select("a")]
    recs = [clean_text(d.text) for d in soup.select("div.title.ellipsis")]
    times = [clean_text(d.text) for d in soup.select("div.hours.ellipsis")]
    comments = [clean_text(d.text) for d in soup.select("div.content")]

    # 合并
    for name, rec, tm, com in zip(names, recs, times, comments):
        if len(listAllContent) >= maxDataSize:
            break
        listAllContent.append([name, rec, tm, com])

    print(f"请求完成, 当前一共{len(listAllContent)}条")

    # 检查是否无新数据
    if len(listAllContent) == lastedNum:
        totalEndNum += 1
        print(f"请求不到更多评论...{totalEndNum}")
        if totalEndNum >= reloadDataNumMax:
            print("结束请求...")
            return True
    else:
        lastedNum = len(listAllContent)
        totalEndNum = 0
```



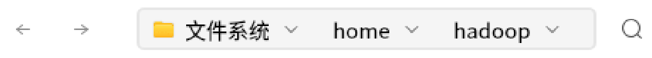
```
    # 更新 cursor
    cursor = js.get("cursor", "").replace("+", "%2B")
    nextCursor = cursor

    return len(listAllContent) >= maxDataSize

def save_as_csv():
    filename = f"{appid}_{maxDataSize}.csv"
    df = pd.DataFrame(listAllContent, columns=["用户名", "推荐", "游戏时长", "评论"])
    df.to_csv(filename, index=False, encoding="utf-8-sig")
    print("存储到", filename)

if __name__ == "__main__":
    getGameName()
    while len(listAllContent) < maxDataSize:
        if getInitCursorValue():
            print("爬取完成")
            break
        time.sleep(0.5)
    save_as_csv()
```

使用黑神话：悟空的评测：



```
(base) hadoop@dblab:~$ python steampachong.py
请输入 appID: 2358720
请输入要请求数据的条数: 5000
```

python 代码和 csv 文件位置：

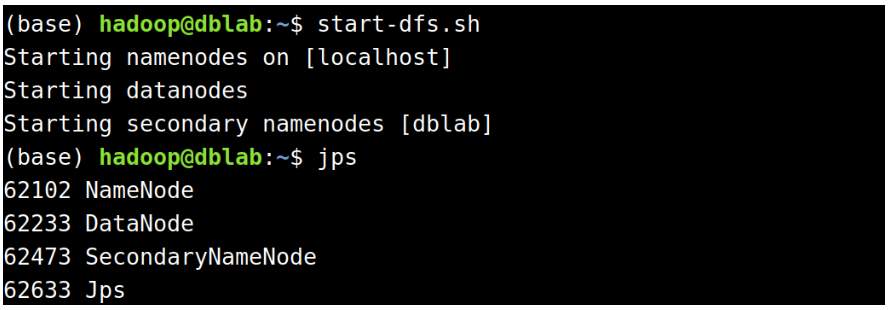


2. Hive 处理与初步分析

1) Hadoop 下启动 HDFS

① start-dfs.sh

② jps　　查看 hadoop 是否启动成功

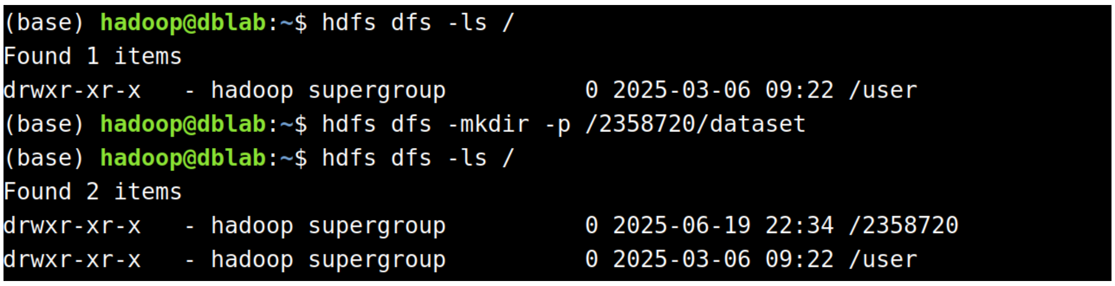


```
(base) hadoop@dblab:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [dblab]
(base) hadoop@dblab:~$ jps
62102 NameNode
62233 DataNode
62473 SecondaryNameNode
62633 Jps
```

2) 把本地的文件上传至 HDFS 中

① hdfs dfs -mkdir -p /2358720/dataset　　在 HDFS 上新建/2358720/dataset

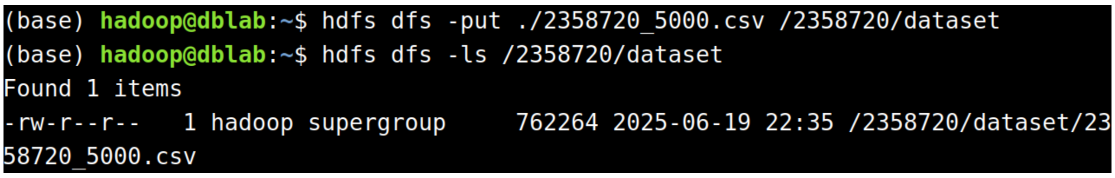
② hdfs dfs -ls /　　查看列表



```
(base) hadoop@dblab:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2025-03-06 09:22 /user
(base) hadoop@dblab:~$ hdfs dfs -mkdir -p /2358720/dataset
(base) hadoop@dblab:~$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2025-06-19 22:34 /2358720
drwxr-xr-x   - hadoop supergroup          0 2025-03-06 09:22 /user
```

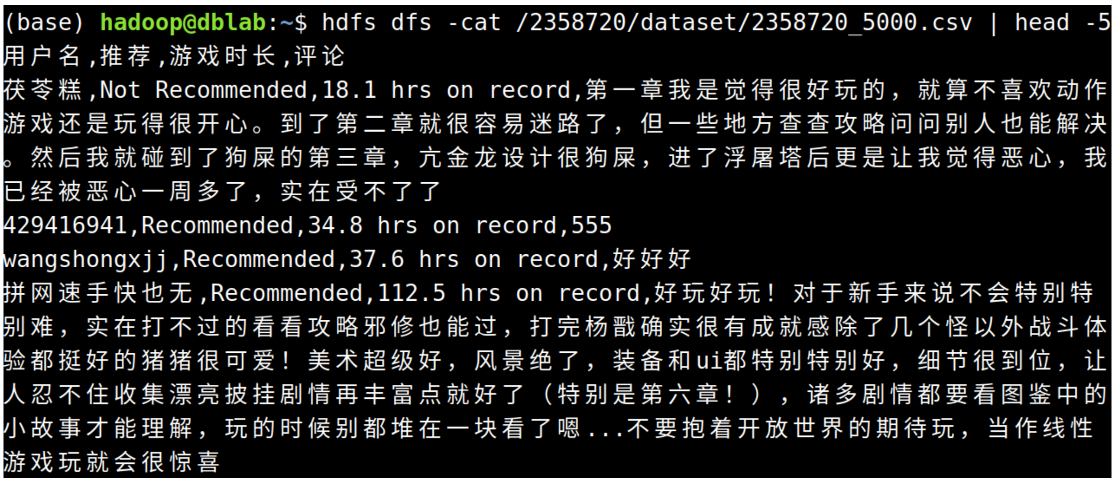
③ hdfs dfs -put ./2358720_5000.csv /2358720/dataset　　上传 2358720_5000.csv

④ hdfs dfs -ls /2358720/dataset　　查看



```
(base) hadoop@dblab:~$ hdfs dfs -put ./2358720_5000.csv /2358720/dataset
(base) hadoop@dblab:~$ hdfs dfs -ls /2358720/dataset
Found 1 items
-rw-r--r--   1 hadoop supergroup     762264 2025-06-19 22:35 /2358720/dataset/23
58720_5000.csv
```

⑤ hdfs dfs -cat /2358720/dataset/2358720_5000.csv | head -5　　查看 HDFS 中 2358720_5000.csv 的前五行

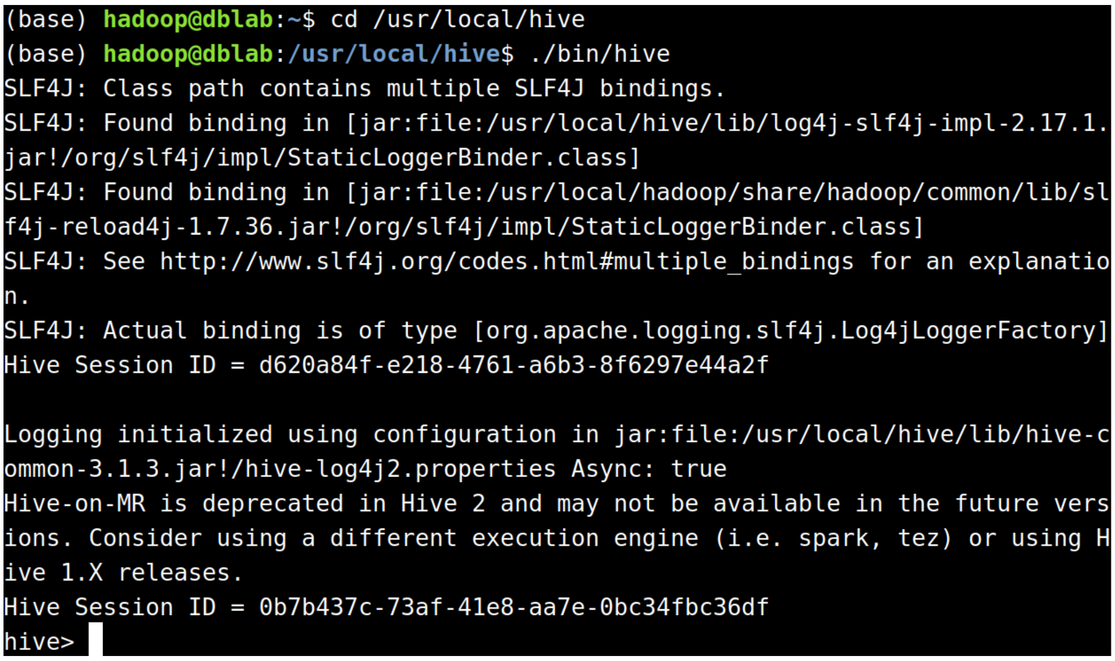


```
(base) hadoop@dblab:~$ hdfs dfs -cat /2358720/dataset/2358720_5000.csv | head -5
用户名,推荐,游戏时长,评论
茯苓糕,Not Recommended,18.1 hrs on record,第一章我是觉得很好玩的，就算不喜欢动作
游戏还是玩得很开心。到了第二章就很容易迷路了，但一些地方查查攻略问问别人也能解决
。然后我就碰到了狗屎的第三章，亢金龙设计很狗屎，进了浮屠塔后更是让我觉得恶心，我
已经被恶心一周多了，实在受不了了
429416941,Recommended,34.8 hrs on record,555
wangshongxjj,Recommended,37.6 hrs on record,好好好
拼网速手快也无,Recommended,112.5 hrs on record,好玩好玩！对于新手来说不会特别特
别难，实在打不过的看看攻略邪修也能过，打完杨戬确实很有成就感除了几个怪以外战斗体
验都挺好的猪猪很可爱！美术超级好，风景绝了，装备和ui都特别特别好，细节很到位，让
人忍不住收集漂亮披挂剧情再丰富点就好了（特别是第六章！），诸多剧情都要看图鉴中的
小故事才能理解，玩的时候别都堆在一块看了嗯...不要抱着开放世界的期待玩，当作线性
游戏玩就会很惊喜
```

3) 导入到数据库 Hive 中

① cd /usr/local/hive　　转到 hive 环境

② ./bin/hive　　启动 hive

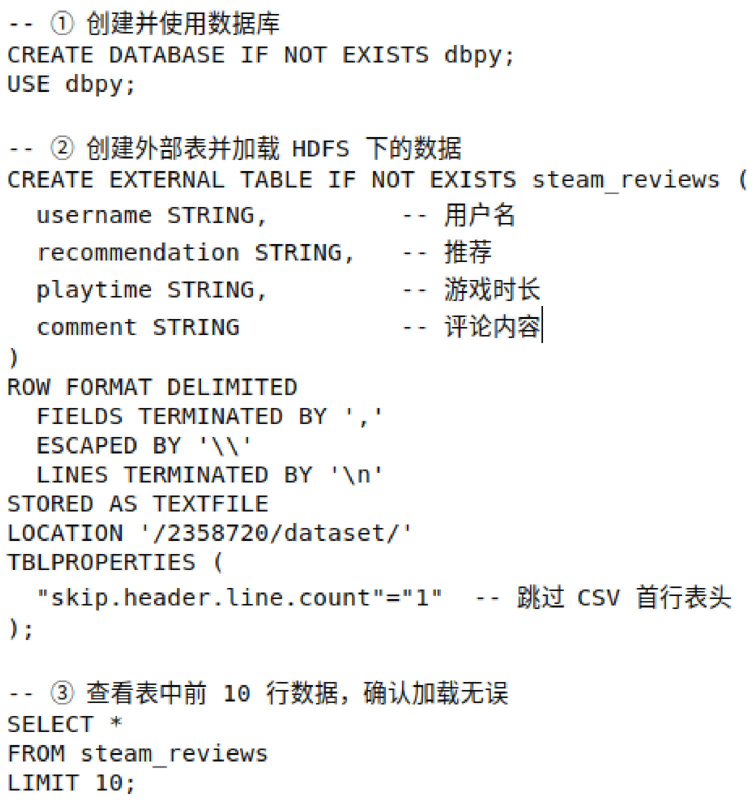


```
(base) hadoop@dblab:~$ cd /usr/local/hive
(base) hadoop@dblab:/usr/local/hive$ ./bin/hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.17.1.
jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/sl
f4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanatio
n.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = d620a84f-e218-4761-a6b3-8f6297e44a2f

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-c
ommon-3.1.3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future vers
ions. Consider using a different execution engine (i.e. spark, tez) or using H
ive 1.X releases.
Hive Session ID = 0b7b437c-73af-41e8-aa7e-0bc34fbc36df
hive>
```

4) 数据初步分析

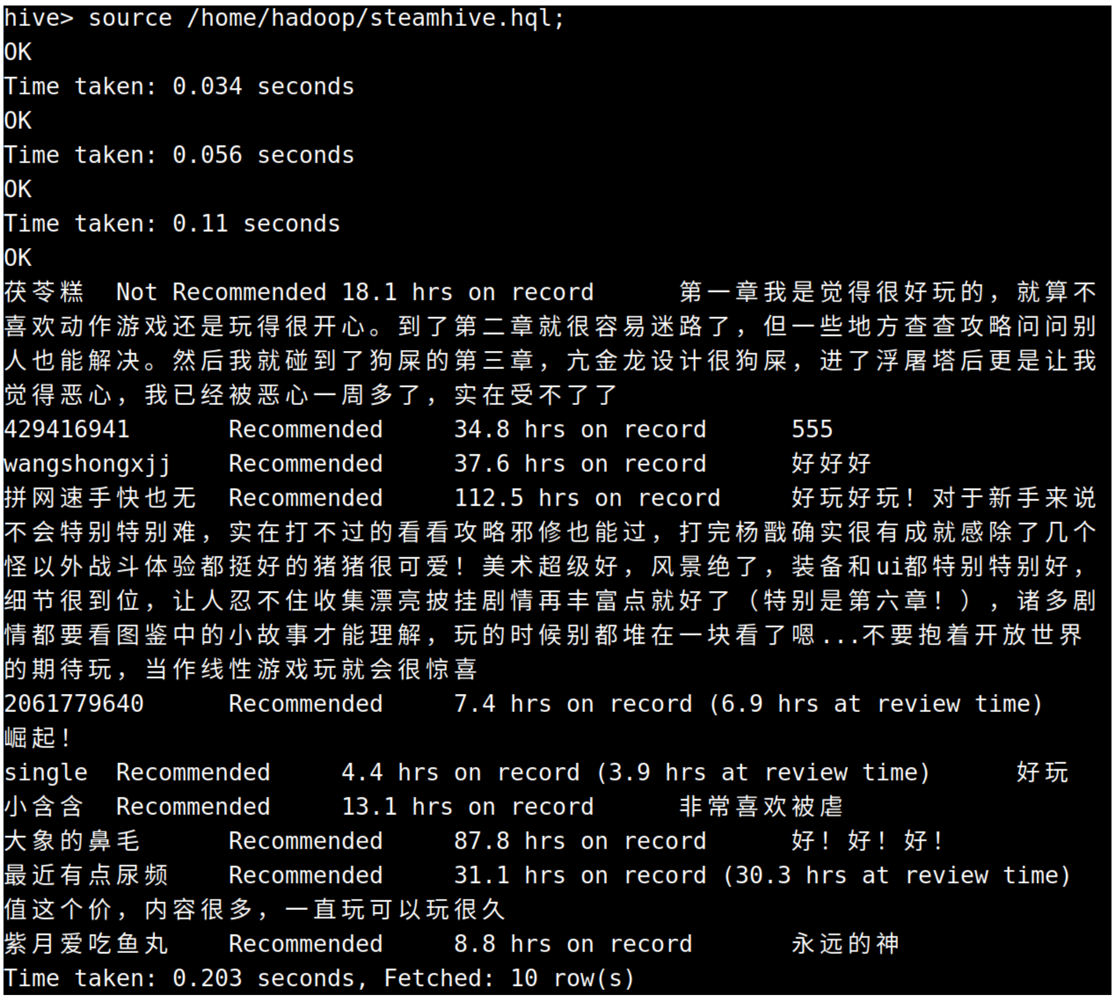
避免按行输入且便于修改，写成文件

hive>source /home/hadoop/steamhive.hql



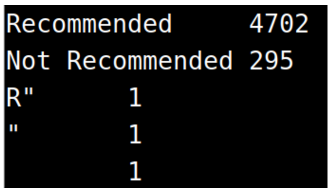
```
-- ① 创建并使用数据库
CREATE DATABASE IF NOT EXISTS dbpy;
USE dbpy;

-- ② 创建外部表并加载 HDFS 下的数据
CREATE EXTERNAL TABLE IF NOT EXISTS steam_reviews (
  username STRING,            -- 用户名
  recommendation STRING,      -- 推荐
  playtime STRING,            -- 游戏时长
  comment STRING              -- 评论内容
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  ESCAPED BY '\\'
  LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/2358720/dataset/'
TBLPROPERTIES (
  "skip.header.line.count"="1"  -- 跳过 CSV 首行表头
);

-- ③ 查看表中前 10 行数据，确认加载无误
SELECT *
FROM steam_reviews
LIMIT 10;
```

①~④ 创建数据库 dbpy，初始化表，查看一下前十行作为检查。


```
hive> source /home/hadoop/steamhive.hql;
OK
Time taken: 0.034 seconds
OK
Time taken: 0.056 seconds
OK
Time taken: 0.11 seconds
OK
茯苓糕  Not Recommended 18.1 hrs on record      第一章我是觉得很好玩的，就算不
喜欢动作游戏还是玩得很开心。到了第二章就很容易迷路了，但一些地方查查攻略问问别
人也能解决。然后我就碰到了狗屎的第三章，亢金龙设计很狗屎，进了浮屠塔后更是让我
觉得恶心，我已经被恶心一周多了，实在受不了了
429416941       Recommended     34.8 hrs on record      555
wangshongxjj    Recommended     37.6 hrs on record      好好好
拼网速手快也无  Recommended     112.5 hrs on record     好玩好玩！对于新手来说
不会特别特别难，实在打不过的看看攻略邪修也能过，打完杨戬确实很有成就感除了几个
怪以外战斗体验都挺好的猪猪很可爱！美术超级好，风景绝了，装备和ui都特别特别好，
细节很到位，让人忍不住收集漂亮披挂剧情再丰富点就好了（特别是第六章！），诸多剧
情都要看图鉴中的小故事才能理解，玩的时候别都堆在一块看了嗯...不要抱着开放世界
的期待玩，当作线性游戏玩就会很惊喜
2061779640      Recommended     7.4 hrs on record (6.9 hrs at review time)
崛起！
single  Recommended     4.4 hrs on record (3.9 hrs at review time)      好玩
小含含  Recommended     13.1 hrs on record      非常喜欢被虐
大象的鼻毛      Recommended     87.8 hrs on record      好！好！好！
最近有点尿频    Recommended     31.1 hrs on record (30.3 hrs at review time)
值这个价，内容很多，一直玩可以玩很久
紫月爱吃鱼丸    Recommended     8.8 hrs on record       永远的神
Time taken: 0.203 seconds, Fetched: 10 row(s)
```

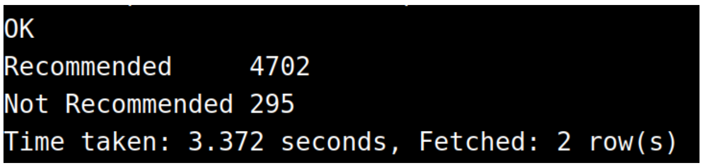
后续准备分析时发现有一些数据不对，经过查找不足 1‰，所以先清洗脏值。

```
-- ③ 数据清洗视图
CREATE OR REPLACE VIEW steam_reviews_clean AS
SELECT
  username,

  -- 1) recommendation，只保留两种值，其他全部映射为 NULL
  CASE
    WHEN recommendation RLIKE '^Recommended$' THEN 'Recommended'
    WHEN recommendation RLIKE '^Not Recommended$' THEN 'Not Recommended'
    ELSE NULL
  END  AS recommendation_clean,

  -- 2) 从 playtime 字段里提取数字部分，转成 DOUBLE,如果没提取到，会是 NULL
  CAST(
    REGEXP_EXTRACT(playtime, '([0-9]+\\.?[0-9]*)', 1)
    AS DOUBLE
  ) AS hours_played,

  comment
FROM steam_reviews;

-- ④ 查看表中前 10 行数据，确认加载无误
SELECT *
FROM steam_reviews_clean
LIMIT 10;
```

清洗掉这样的脏值：
```
Recommended      4702
Not Recommended 295
R"       1
"        1
         1
```

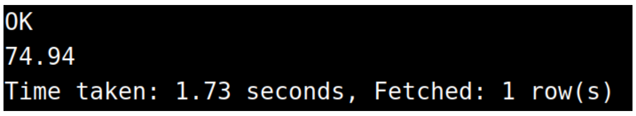
⑤ 统计好评和差评数量

```
-- ⑤ 统计好评 vs 差评 数量
SELECT
  recommendation_clean,
  COUNT(*) AS cnt
FROM steam_reviews_clean
WHERE recommendation_clean IS NOT NULL
GROUP BY recommendation_clean
ORDER BY cnt DESC;
```


```
OK
Recommended      4702
Not Recommended 295
Time taken: 3.372 seconds, Fetched: 2 row(s)
```

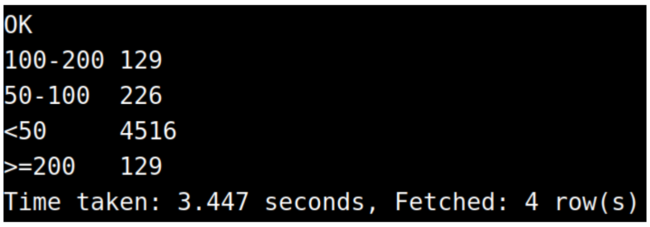
⑥计算平均游戏时长

```
-- ⑥ 计算平均游戏时长
SELECT
  ROUND(AVG(hours_played),2) AS avg_hours_played
FROM steam_reviews_clean
WHERE hours_played IS NOT NULL;
```

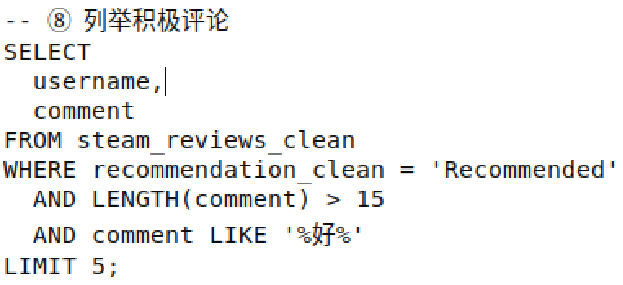

```
OK
74.94
Time taken: 1.73 seconds, Fetched: 1 row(s)
```

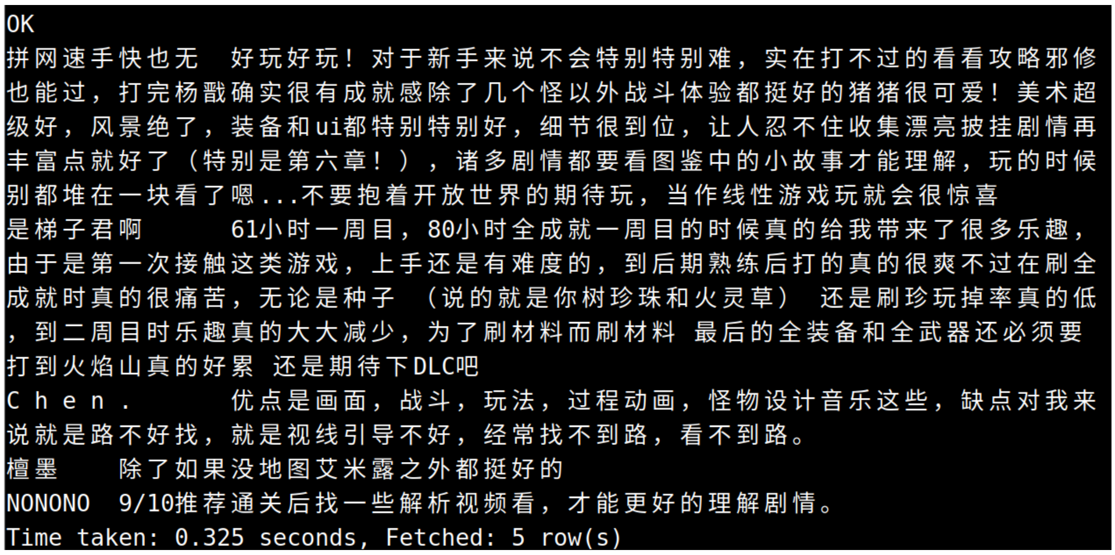
⑦统计评论长度

```
-- ⑦ 评论长度分布
SELECT
  CASE
    WHEN LENGTH(comment)<50   THEN '<50'
    WHEN LENGTH(comment)<100  THEN '50-100'
    WHEN LENGTH(comment)<200  THEN '100-200'
    ELSE '>=200'
  END AS length_bucket,
  COUNT(*) AS cnt
FROM steam_reviews_clean
GROUP BY
  CASE
    WHEN LENGTH(comment)<50   THEN '<50'
    WHEN LENGTH(comment)<100  THEN '50-100'
    WHEN LENGTH(comment)<200  THEN '100-200'
    ELSE '>=200'
  END
ORDER BY length_bucket;
```


```
OK
100-200 129
50-100  226
<50     4516
>=200   129
Time taken: 3.447 seconds, Fetched: 4 row(s)
```

⑧列举积极评论

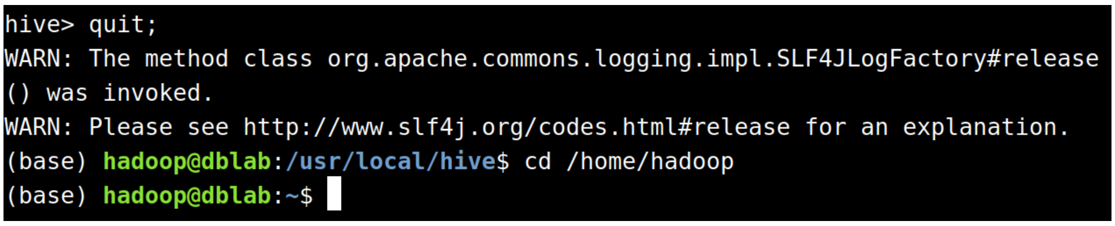

```
-- ⑧ 列举积极评论
SELECT
  username,
  comment
FROM steam_reviews_clean
WHERE recommendation_clean = 'Recommended'
  AND LENGTH(comment) > 15
  AND comment LIKE '%好%'
LIMIT 5;
```


```
OK
拼网速手快也无　好玩好玩！对于新手来说不会特别特别难，实在打不过的看看攻略邪修
也能过，打完杨戬确实很有成就感除了几个怪以外战斗体验都挺好的猪猪很可爱！美术超
级好，风景绝了，装备和ui都特别特别好，细节很到位，让人忍不住收集漂亮披挂剧情再
丰富点就好了（特别是第六章！），诸多剧情都要看图鉴中的小故事才能理解，玩的时候
别都堆在一块看了嗯...不要抱着开放世界的期待玩，当作线性游戏玩就会很惊喜
是梯子君啊　　61小时一周目，80小时全成就一周目的时候真的给我带来了很多乐趣，
由于是第一次接触这类游戏，上手还是有难度的，到后期熟练后打的真的很爽不过在刷全
成就时真的很痛苦，无论是种子 （说的就是你树珍珠和火灵草） 还是刷珍玩掉率真的低
，到二周目时乐趣真的大大减少，为了刷材料而刷材料 最后的全装备和全武器还必须要
打到火焰山真的好累 还是期待下DLC吧
Ｃｈｅｎ．　　优点是画面，战斗，玩法，过程动画，怪物设计音乐这些，缺点对我来
说就是路不好找，就是视线引导不好，经常找不到路，看不到路。
檀墨　　除了如果没地图艾米露之外都挺好的
NONONO　9/10推荐通关后找一些解析视频看，才能更好的理解剧情。
Time taken: 0.325 seconds, Fetched: 5 row(s)
```

最后，退出 hive

hive>quit;

cd /home/hadoop/　　回到 base 环境


```
hive> quit;
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release
() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
(base) hadoop@dblab:/usr/local/hive$ cd /home/hadoop
(base) hadoop@dblab:~$
```

3. MapReduce 评论词语共现分析

我们想使用 MapReduce 来处理高复杂度的任务，数据挖掘课上我们了解到根据词语共同出现的频次来做推荐的方式，所以做了词语共现分析。

mapper.py

```python
#!/home/hadoop/anaconda3/bin/python
# -*- coding: utf-8 -*-
import sys
import csv
import jieba

# 停用词
stopwords = set()
with open('stopwords.txt', encoding='utf-8') as f:
    for line in f:
        w = line.strip()
        if w:
            stopwords.add(w)

def bucket_pairs(words):
    """对排序后的词列表两两组合，保证字典序 w[i]<w[j]"""
    n = len(words)
    for i in range(n):
        for j in range(i+1, n):
            yield words[i], words[j]

reader = csv.reader(sys.stdin)
# 跳过 header
next(reader, None)

for cols in reader:
    comment = cols[3].strip()
    if not comment:
        continue

    # 分词
    tokens = [tk for tk in jieba.lcut(comment)
              if tk.strip() and tk not in stopwords and len(tk) > 1]

    # 去重并排序
    uniq = sorted(set(tokens))

    for w1, w2 in bucket_pairs(uniq):
        print(f"{w1},{w2}\t1")
```

reducer.py

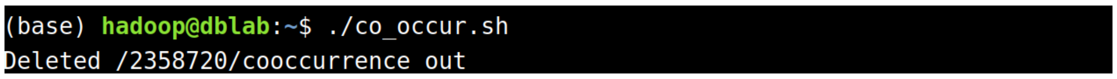
```python
#!/home/hadoop/anaconda3/bin/python
# -*- coding: utf-8 -*-
import sys

current_key = None
current_count = 0

for line in sys.stdin:
    key, value = line.strip().split('\t', 1)
    try:
        count = int(value)
    except:
        continue

    if key == current_key:
        current_count += count
    else:
        if current_key:
            print(f"{current_key}\t{current_count}")
        current_key = key
        current_count = count

if current_key:
    print(f"{current_key}\t{current_count}")
```

采用的停用词表链接：

https://github.com/goto456/stopwords/blob/master/baidu_stopwords.txt

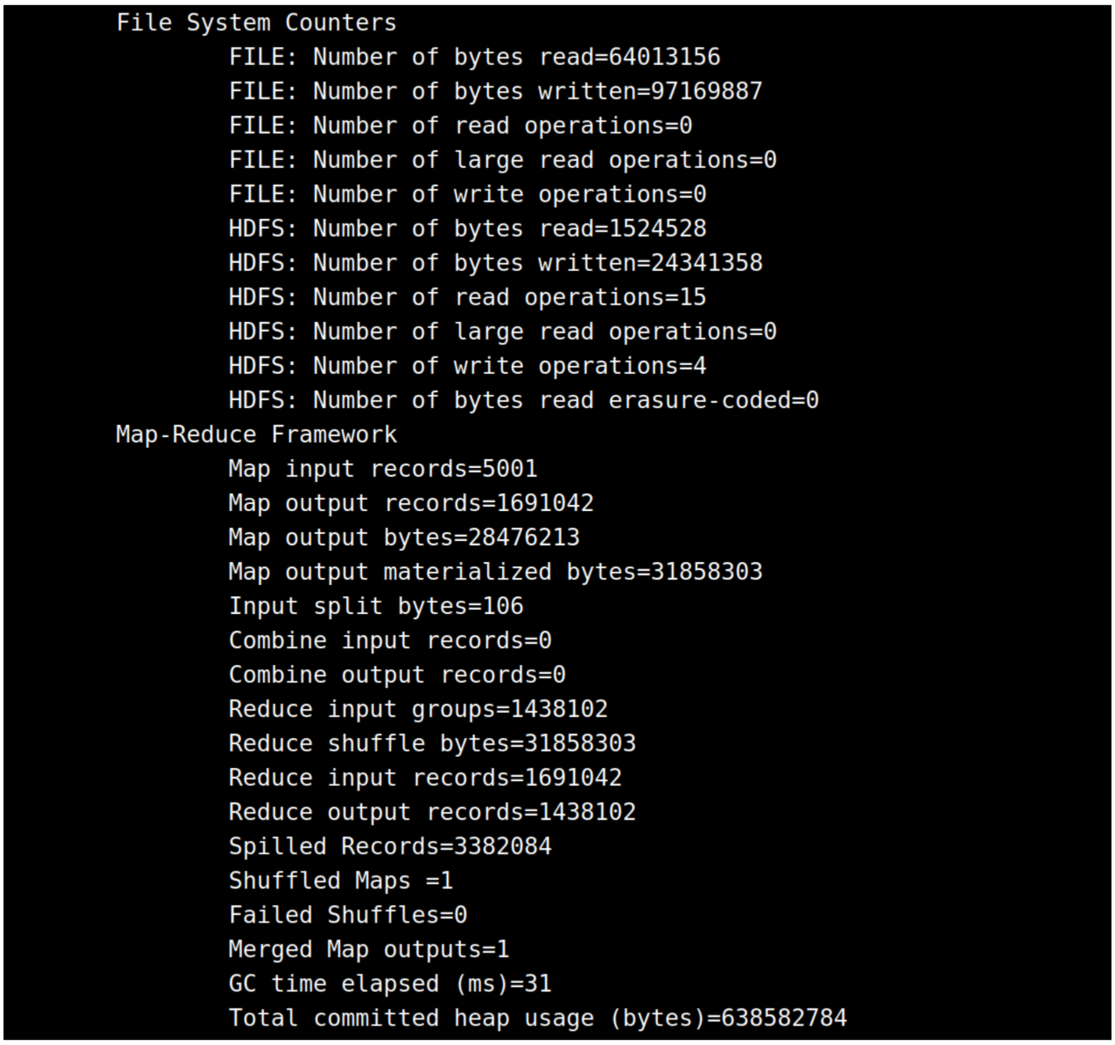
这里把 hadoop 启动 MapReduce 的命令也编写成一个文件 co_occur.sh

```bash
#!/bin/bash
INPUT=/2358720/dataset/2358720_5000.csv
OUTPUT=/2358720/cooccurrence_out

hdfs dfs -rm -r $OUTPUT

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
  -files mapper.py,reducer.py,stopwords.txt \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py" \
  -input $INPUT \
  -output $OUTPUT
```



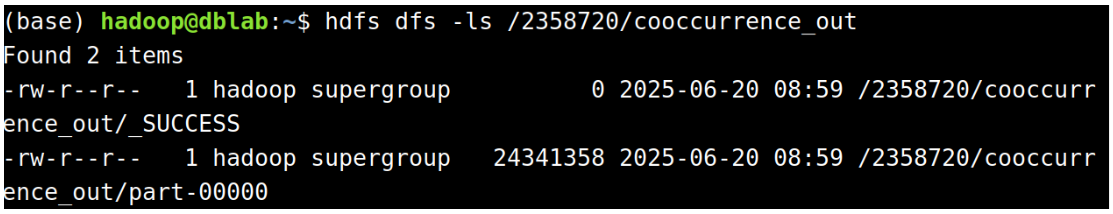
```
(base) hadoop@dblab:~$ ./co_occur.sh
Deleted /2358720/cooccurrence_out
```

部分运行日志

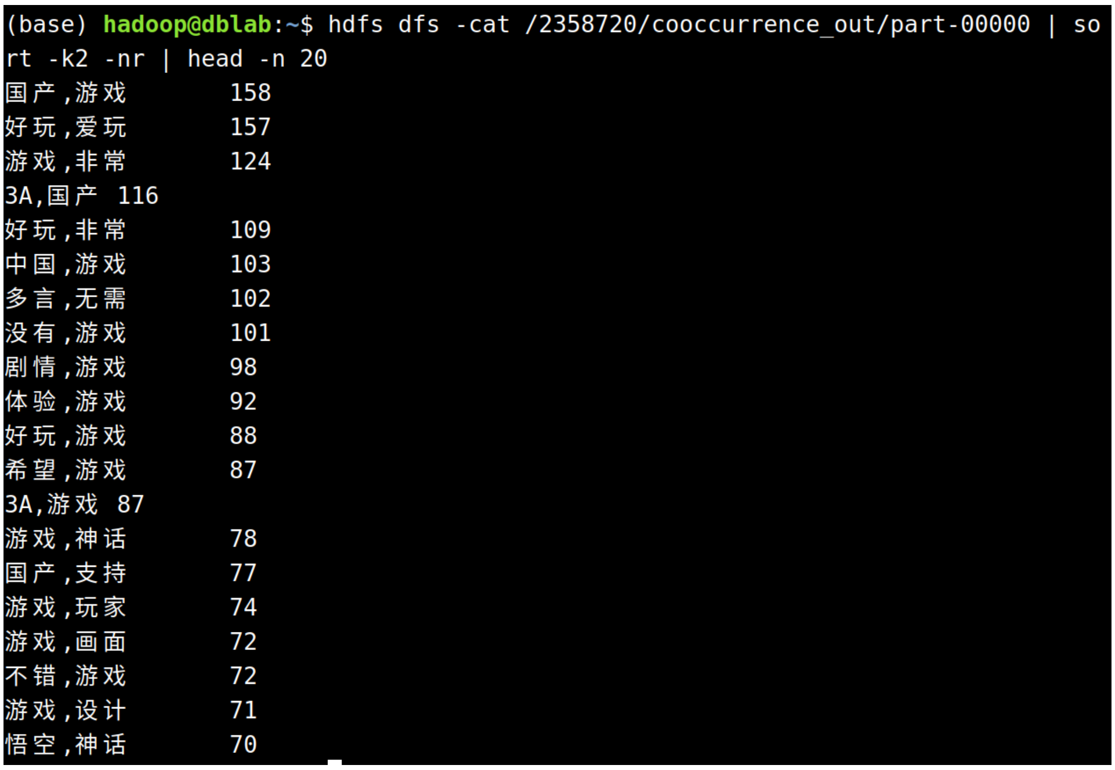


```
        File System Counters
                FILE: Number of bytes read=64013156
                FILE: Number of bytes written=97169887
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1524528
                HDFS: Number of bytes written=24341358
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=5001
                Map output records=1691042
                Map output bytes=28476213
                Map output materialized bytes=31858303
                Input split bytes=106
                Combine input records=0
                Combine output records=0
                Reduce input groups=1438102
                Reduce shuffle bytes=31858303
                Reduce input records=1691042
                Reduce output records=1438102
                Spilled Records=3382084
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=31
                Total committed heap usage (bytes)=638582784
```

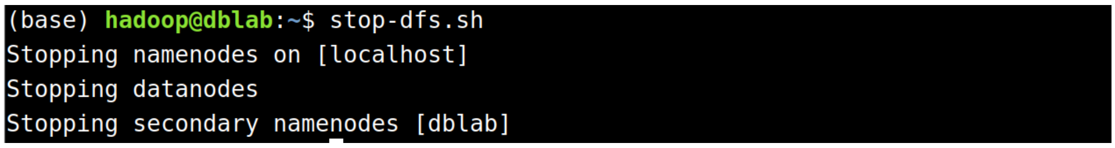
检查结果输出



```
(base) hadoop@dblab:~$ hdfs dfs -ls /2358720/cooccurrence_out
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2025-06-20 08:59 /2358720/cooccurr
ence_out/_SUCCESS
-rw-r--r--   1 hadoop supergroup   24341358 2025-06-20 08:59 /2358720/cooccurr
ence_out/part-00000
```


```
(base) hadoop@dblab:~$ hdfs dfs -cat /2358720/cooccurrence_out/part-00000 | so
rt -k2 -nr | head -n 20
国产,游戏       158
好玩,爱玩       157
游戏,非常       124
3A,国产  116
好玩,非常       109
中国,游戏       103
多言,无需       102
没有,游戏       101
剧情,游戏       98
体验,游戏       92
好玩,游戏       88
希望,游戏       87
3A,游戏  87
游戏,神话       78
国产,支持       77
游戏,玩家       74
游戏,画面       72
不错,游戏       72
游戏,设计       71
悟空,神话       70
```

用完要关闭 HDFS

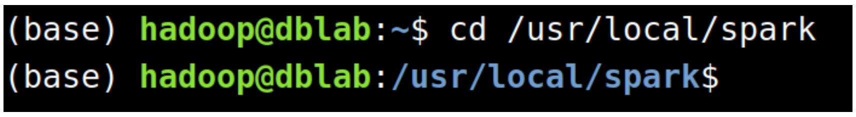

```
(base) hadoop@dblab:~$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [dblab]
```

4. Spark 跑一个评论分类模型

我们希望把这个大数据项目与机器学习结合，考虑到我们学习的分类任务比较多，设计一个根据评论内容分类为正面或反面评论的逻辑回归模型，也可以沿用分词操作，并尝试大数据讲座提到的 Word2Vec 特征训练。

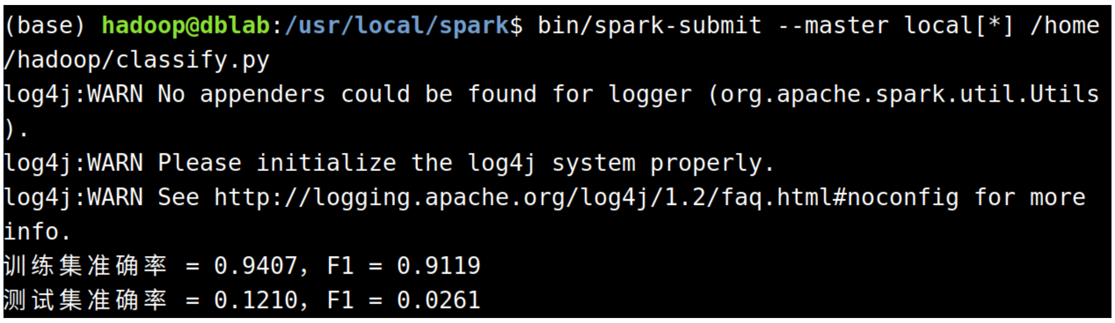
先 cd /usr/local/spark 转入 spark 环境

然后用 bin/spark-submit –master local[*] /home/hadoop/classify.py 来上传运行


```
(base) hadoop@dblab:~$ cd /usr/local/spark
(base) hadoop@dblab:/usr/local/spark$
```

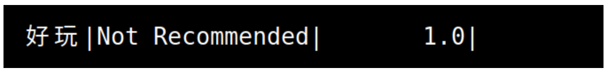
起初用黑神话：悟空的评测作为训练，然后爬取了三国杀：一将成名的评测用来做测试集。

但最后效果非常感人，训练集好评很多，测试集都是差评

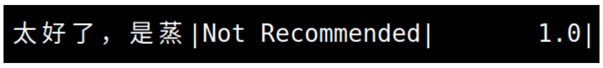


```
(base) hadoop@dblab:/usr/local/spark$ bin/spark-submit --master local[*] /home
/hadoop/classify.py
log4j:WARN No appenders could be found for logger (org.apache.spark.util.Utils
).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more
info.
训练集准确率 = 0.9407, F1 = 0.9119
测试集准确率 = 0.1210, F1 = 0.0261
```

而且有很多无参考意义的测评



```
 好玩|Not Recommended|        1.0|
```

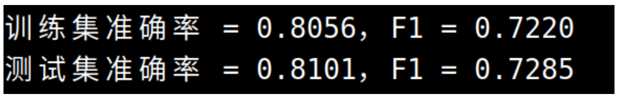


```
太好了，是蒸|Not Recommended|        1.0|
```

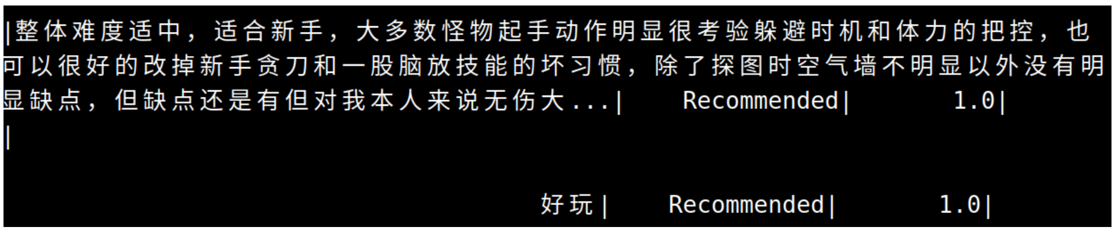
classify.py

```python
# classify.py
# -*- coding: utf-8 -*-

from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import (
    RegexTokenizer,
    StopWordsRemover,
    Word2Vec,
    StringIndexer
)
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import col

def main():
    spark = SparkSession.builder \
        .appName("SteamSentimentWord2Vec") \
        .master("local[*]") \
        .getOrCreate()

    train_df = spark.read \
        .option("header", "true") \
        .csv("file:///home/hadoop/2358720_5000.csv") \
        .withColumnRenamed("推荐", "recommended") \
        .withColumnRenamed("评论", "comment") \
        .select("recommended", "comment") \
        .filter(col("comment").isNotNull() & (col("comment") != ""))

    test_df = spark.read \
        .option("header", "true") \
        .csv("file:///home/hadoop/1180320_1000.csv") \
        .withColumnRenamed("推荐", "recommended") \
        .withColumnRenamed("评论", "comment") \
        .select("recommended", "comment") \
        .filter(col("comment").isNotNull() & (col("comment") != ""))

    label_indexer = StringIndexer(
        inputCol="recommended",
        outputCol="label",
        stringOrderType="alphabetAsc"
    )

    tokenizer    = RegexTokenizer(inputCol="comment", outputCol="words", pattern="\\W+")
    stop_remover = StopWordsRemover(inputCol="words", outputCol="filtered")
    # Word2Vec: 100 维，忽略出现少于 5 次的词
    word2Vec = Word2Vec(
        inputCol="filtered",
        outputCol="features",
        vectorSize=100,
        minCount=5
    )

    lr = LogisticRegression(maxIter=20, regParam=0.01, elasticNetParam=0.0)

    pipeline = Pipeline(stages=[
        label_indexer,
        tokenizer,
        stop_remover,
        word2Vec,
        lr
    ])

    model = pipeline.fit(train_df)

    pred_train = model.transform(train_df)
    evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
    acc_train = evaluator.evaluate(pred_train)
    f1_train  = MulticlassClassificationEvaluator(metricName="f1").evaluate(pred_train)
    print(f"训练集准确率 = {acc_train:.4f}, F1 = {f1_train:.4f}")

    pred_test = model.transform(test_df)
    acc_test = evaluator.evaluate(pred_test)
    f1_test  = MulticlassClassificationEvaluator(metricName="f1").evaluate(pred_test)
    print(f"测试集准确率 = {acc_test:.4f}, F1 = {f1_test:.4f}")

    print("\n--- 预测错误示例 (前 5 条) ---")
    pred_test.select("comment", "recommended", "prediction") \
        .where(col("label") != col("prediction")) \
        .show(5, truncate=100)

    spark.stop()

if __name__ == "__main__":
    main()
```

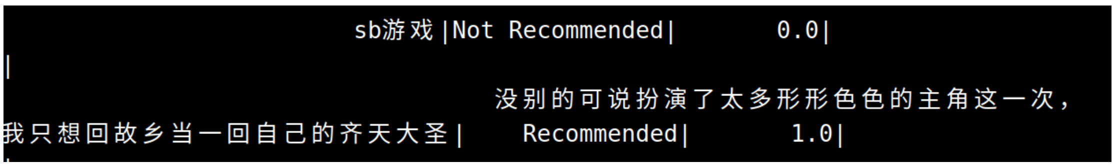
为了让模型好坏能够学习的均衡一些，把两个数据集合并，再进行随机划分测试集和训练集，有一定的效果



```
训练集准确率 = 0.8056, F1 = 0.7220
测试集准确率 = 0.8101, F1 = 0.7285
```



```
|整体难度适中，适合新手，大多数怪物起手动作明显很考验躲避时机和体力的把控，也可以很好的改掉新手贪刀和一股脑放技能的坏习惯，除了探图时空气墙不明显以外没有明显缺点，但缺点还是有但对我本人来说无伤大...|     Recommended|        1.0|
|
                                                     好玩|     Recommended|        1.0|
```



```
                                   sb游戏|Not Recommended|        0.0|
|
                                            没别的可说扮演了太多形形色色的主角这一次，我只想回故乡当一回自己的齐天大圣|     Recommended|        1.0|
```

classify2.py

```python
# classify2.py
# -*- coding: utf-8 -*-

from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import (
    RegexTokenizer,
    StopWordsRemover,
    Word2Vec,
    StringIndexer
)
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import col, rand

def main():
    spark = SparkSession.builder \
        .appName("SteamSentimentCombined") \
        .master("local[*]") \
        .getOrCreate()

    def load(path):
        return spark.read \
            .option("header", "true") \
            .option("inferSchema", "false") \
            .csv(f"file://{path}") \
            .withColumnRenamed("推荐", "recommended") \
            .withColumnRenamed("评论", "comment") \
            .select("recommended", "comment") \
            .filter(col("comment").isNotNull() & (col("comment") != ""))

    df1 = load("/home/hadoop/2358720_5000.csv")
    df2 = load("/home/hadoop/1180320_1000.csv")

    all_df = df1.union(df2)
    train_df, test_df = all_df.randomSplit([0.8, 0.2], seed=42)

    label_indexer = StringIndexer(
        inputCol="recommended",
        outputCol="label",
        stringOrderType="alphabetAsc"
    )

    tokenizer     = RegexTokenizer(inputCol="comment", outputCol="words", pattern="\\W+")
    stop_remover = StopWordsRemover(inputCol="words", outputCol="filtered")
    word2Vec      = Word2Vec(
        inputCol="filtered",
        outputCol="features",
        vectorSize=100,
        minCount=5
    )
    lr = LogisticRegression(maxIter=20, regParam=0.01, elasticNetParam=0.0)

    pipeline = Pipeline(stages=[
        label_indexer,
        tokenizer,
        stop_remover,
        word2Vec,
        lr
    ])

    model = pipeline.fit(train_df)

    pred_train = model.transform(train_df)
    evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
    acc_train = evaluator.evaluate(pred_train)
    f1_train  = MulticlassClassificationEvaluator(metricName="f1").evaluate(pred_train)
    print(f"训练集准确率 = {acc_train:.4f}, F1 = {f1_train:.4f}")

    # 9. 测试集评估
    pred_test = model.transform(test_df)
    acc_test = evaluator.evaluate(pred_test)
    f1_test  = MulticlassClassificationEvaluator(metricName="f1").evaluate(pred_test)
    print(f"测试集准确率 = {acc_test:.4f}, F1 = {f1_test:.4f}")

    print("\n--- 随机 10 条预测示例 ---")
    pred_test.select("comment", "recommended", "prediction") \
        .orderBy(rand())  \
        .limit(10) \
        .show(truncate=100)

    spark.stop()

if __name__ == "__main__":
    main()
```

但对许多测评还有学习不全面的问题，未来可以作更复杂的语义序列提取，考虑采用深度学习的模型来做这个项目。