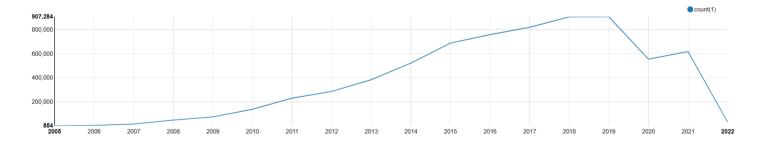
评价统计及分析模块Spark

```
1 %pyspark
2
3 #编写者: 张子杰
4 #功能: 评论词云图
5 #时间: 2024.1.3
6
 7
8 from pyspark import HiveContext
9 import matplotlib.pyplot as plt
10 from wordcloud import WordCloud
11 import pandas as pd
12
13
14 hc=HiveContext(sc)
15 df=hc.sql("SELECT rev_text from review limit 100000")
16
17
18 wordcloud = WordCloud(width=1600, height=800,
   background_color="white").generate(df.toPandas().to_string())
19
20 plt.figure(figsize=(10, 5))
21 plt.imshow(wordcloud, interpolation="bilinear")
22 plt.axis("off")
23 plt.show()
24
```



```
1 %pyspark
2
3 # 编写者:凌晨
4 # 功能:分析每年评论的数量
5 # 时间:2024.1.3
6 from pyspark.sql import HiveContext
7
8 hc = HiveContext(sc)
9 result = hc.sql("SELECT year(to_date(rev_date)) as year, COUNT(*) AS cot FROM review GROUP BY year(to_date(rev_date)) ORDER BY year DESC")
10 z.show(result)
```



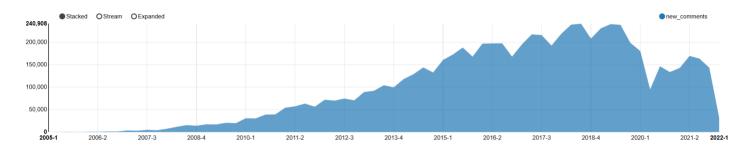
1 %pyspark

2

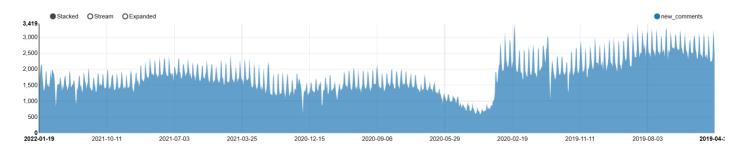
3 # 编写者:凌晨

4 # 功能:分析每季度评论的数量,每三个月视为一个季度。如2004-4代表2004的第四个季度

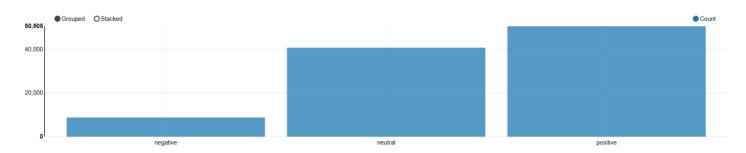
```
5 # 时间:2024.1.3
6
7 hc = HiveContext(sc)
8 result = hc.sql('''
9 SELECT
10 CONCAT(YEAR(rev_date), '-', CEIL(MONTH(rev_date) / 3)) AS quarter,
11 COUNT(*) AS new_comments
12 FROM review
13 GROUP BY CONCAT(YEAR(rev_date), '-', CEIL(MONTH(rev_date) / 3))
14 ORDER BY quarter
15 ''')
16 z.show(result)
```



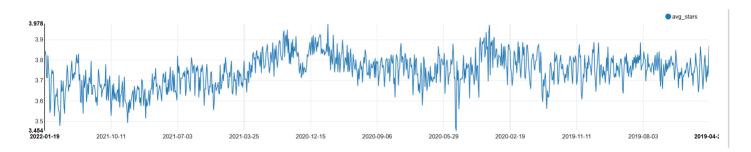
```
1 %pyspark
2
3 # 编写者:凌晨
4 # 功能:统计按天的评论数目
5 # 时间:2024.1.3
6
7 from pyspark.sql import HiveContext
8
9 hc = HiveContext(sc)
10 result = hc.sql("SELECT DATE(rev_date) AS date, COUNT(*) AS new_comments FROM review GROUP BY DATE(rev_date) ORDER BY date DESC LIMIT 1000")
11 z.show(result)
```



```
1 %pyspark
 2
3 # 编写者:张子杰
 4 # 功能: 评论情感分析统计
 5 # 时间:2024.1.4
 6
7 from pyspark import HiveContext
 8 from pyspark.sql.functions import col, udf
9 from pyspark.sql.types import StringType
10 from textblob import TextBlob
11
12 hc = HiveContext(sc)
13
14 df = hc.sql("SELECT rev_text FROM review LIMIT 100000")
15
16 def analyze_sentiment(text):
       analysis = TextBlob(str(text))
17
       # 判断情感极性
18
       if analysis.sentiment.polarity > 0.15:
19
           return 'positive'
20
       elif analysis.sentiment.polarity < -0.15:
21
           return 'negative'
22
       else:
23
           return 'neutral'
24
25
26 df = df.toPandas()
27
28 df['sentiment'] = df.apply(analyze_sentiment, axis=1)
29
30
31 result = df.groupby('sentiment').size().reset_index(name='Count')
32
33
34 z.show(result)
```



```
1 %pyspark
2
3 # 编写者: 钟鹏
4 # 功能: 统计每天评论分数
5 # 时间: 2024.1.4
6
7 from pyspark.sql import HiveContext
  from pyspark.sql.functions import *
9
10 hcx = HiveContext(sc)
11
   df_review = hcx.table('review')
12
13
   # 将日期列转换为日期格式,并按日期分组计算平均stars
14
  result = df_review.select('rev_date', 'rev_stars') \
15
       .withColumn('date', to_date(col('rev_date'))) \
16
       .groupBy('date') \
17
18
       .agg(avg('rev_stars').alias('avg_stars')) \
       .orderBy('date',ascending=False)
19
20
21 z.show(result)
```



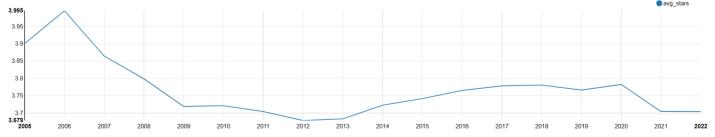
```
1 %pyspark
2 # 編写者: 钟鹏
3 # 功能: 统计每月评论分数
4 # 时间: 2024.1.4
5 from pyspark.sql import HiveContext
6 from pyspark.sql.functions import *
7
8 hcx = HiveContext(sc)
9
10 df_review = hcx.table('review')
11 result_monthly = df_review.select('rev_date', 'rev_stars') \
12 .withColumn('year_month', date_format(to_date(col('rev_date')), 'yyyy-MM')) \
13 .groupBy('year_month') \
```

```
.agg(avg('rev_stars').alias('avg_stars')) \
.orderBy('year_month',ascending=False)

z.show(result_monthly)
```

```
4.333
4.2
4.1
4.1
4.3
3.9
3.8
3.7
3.616
2002-01 2020-05 2018-09 2017-01 2015-05 2013-09 2012-01 2010-05 2008-09 2007-01 2005-02
```

```
1 %pyspark
 2 # 编写者: 钟鹏
 3 # 功能: 统计每年评论分数
4 # 时间: 2024.1.4
 5 from pyspark.sql import HiveContext
 6 from pyspark.sql.functions import *
 7
8 hcx = HiveContext(sc)
9
10 df_review = hcx.table('review')
11 result_yearly = df_review.select('rev_date', 'rev_stars') \
       .withColumn('year', year(to_date(col('rev_date')))) \
12
       .groupBy('year') \
13
       .agg(avg('rev_stars').alias('avg_stars')) \
14
       .orderBy('year',ascending=False)
15
16
17 z.show(result_yearly)
```



4