第5章 网站转化率统计

学习目标

- 掌握网站转化率统计实现思路。
- 了解如何生成用户浏览网页数据。
- 掌握如何创建 Spark 连接并读取数据集。
- 掌握利用 Spark SQL 统计每个页面访问次数。
- 掌握利用 Spark SQL 获取每个用户浏览网页的顺序。
- 掌握利用 Spark SQL 合并同一用户浏览的网页。
- 掌握利用 Spark SQL 统计每个单跳的次数。
- 掌握利用 Spark SQL 计算页面单跳转化率。
- 掌握将数据持久化到 HBase 数据库。
- 熟悉通过 Spark On YARN 运行程序。

网站转化率(conversion rate)是指用户进行相应目标行动的访问次数与总访问次数的比率。这里所指的相应目标行动可以是用户登录、注册、浏览、购买等一系列用户行为,因此网站转化率是一个广义的概念。页面单跳转化率是网站转化率的一种统计形式,通过统计页面单跳转化率,来优化页面布局及营销策略,使访问网站的用户可以更深层次地浏览网站。本章对用户浏览网页数据进行分析,从而统计出页面单跳转化率。

5.1 数据集分析

本需求通过编写 Java 程序模拟生成用户浏览网页数据作为数据集,数据集中的每一行数据代表一个用户的浏览行为,所有浏览行为都与页面和用户有关,该数据集中包含了10000 条用户浏览页面的数据,虽然数据比较多,但是数据内容格式基本类似,这里选取其中一条数据进行分析,具体如下。

```
{
      "actionTime":"2020-07-22 06:34:02",
      "sessionid":"98ac879b5a0a4a4eb117dffd84da1ff4",
      "pageid":3,
      "userid":8
}
```

上述数据包含 4 个字段，每个字段都代表特定的含义，具体介绍如下。

- actionTime：用户访问页面的时间。
- sessionid：用于标识用户行为的唯一值。
- pageid：用户浏览网页的 ID。
- userid：用户 ID。

5.2 实现思路分析

当用户浏览网页时，通过当前浏览页面(A)跳转到另一个页面(B)，此用户行为被称为一次 A→B 的单跳。如计算 A→B 的页面单跳转化率，则计算公式如下：

A→B 页面单跳转化率＝A→B 的单跳总数/A 总访问次数

通过上式可以看出，计算页面单跳转化率需要两部分数据，分别是 A→B 的单跳总数和 A 总访问次数，其中 A 总访问次数可以通过聚合操作获取，A→B 的单跳总数实现思路如下。

(1) 根据用户 ID 和访问时间对数据集进行排序操作，获取每个用户浏览网页的顺序。

(2) 根据用户 ID 对排序后的数据进行分组操作，将同一用户浏览的网页进行合并。

(3) 对分组后的数据进行转换操作，将同一用户浏览的网页按照浏览顺序转换为单跳

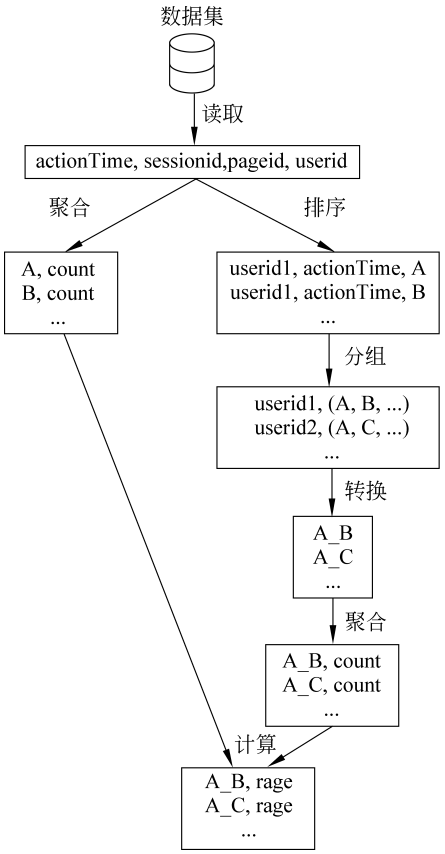

图 5-1 页面单跳转化率统计实现过程

形式。

(4) 对转换后的数据进行聚合操作统计每个单跳的总数,其中包括 A→B 的单跳总数。

此时,我们可以将 A 总访问次数和 A→B 的单跳总数这两部分数据带入计算公式中,得出 A→B 的页面单跳转化率,页面单跳转化率统计实现过程如图 5-1 所示。

5.3　实现网站转化率统计

5.3.1　生成用户浏览网页数据

实现网站转化率统计程序之前,需要模拟生成用户浏览网页数据,这里主要通过编写 Java 应用程序的方式,实现用户浏览网页数据的生成,具体实现步骤如下。

(1) 在项目 SparkProject 的 java 目录下新建 Package 包 cn.itcast.conversion,用于存放实现网站转化率统计的 Java 文件。在包 cn.itcast.conversion 中创建文件 GenerateData.java,用于模拟生成用户浏览网页数据,具体代码如文件 5-1 所示。

文件 5-1　GenerateData.java

```
import com.alibaba.fastjson.JSONObject;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
import java.util.UUID;
public class GenerateData {
    public static void main(String[] arg) throws IOException {
        //创建 Random 对象,用于生成随机数
        Random random =new Random();
        //创建 JSONObject 对象,用于存储用户浏览网页数据
        JSONObject jsonObject =new JSONObject();
        //格式化日期的格式为 yyyy-MM-dd
        SimpleDateFormat DATE_FORMAT =new SimpleDateFormat("yyyy-MM-dd");
        //获取当前日期,进行格式化处理获取基础日期 date
        String date =DATE_FORMAT.format(new Date());
        //指定用户浏览网页数据输出路径及文件名称
        String outputFile ="D:\sparkdata\user_conversion.json";
        //为基础日期 date 随机添加小时
        String baseActionTime =
                date +" " +fulfuill(String.valueOf(random.nextInt(24)));
        //创建 FileWriter 输出流对象
        FileWriter fw =new FileWriter(outputFile);
        //通过 for 循环生成 10000 条用户浏览网页数据
        for (int i =0;i<10000;i++){
            //通过 UUID 类的方法 randomUUID()随机生成 sessionid
            String sessionid =
                    UUID.randomUUID().toString().replace("-", "");
```

```
30                //通过 Random 对象的方法 nextInt()随机生成 pageid
31                long pageid =random.nextInt(10)+1;
32                //通过 Random 对象的方法 nextInt()随机生成 userid
33                long userid =random.nextInt(100)+1;
34                //为添加小时的基础日期随机添加分和秒,随机生成 actionTime
35                String actionTime =baseActionTime +":"
36                        +fulfuill(String.valueOf(random.nextInt(60)))
37                        +":" +fulfuill(String.valueOf(random.nextInt(60)));
38                //将 sessionid、pageid、userid 和 actionTime 添加到 JSONObject 对象
39                jsonObject.put("sessionid",sessionid);
40                jsonObject.put("pageid",pageid);
41                jsonObject.put("userid",userid);
42                jsonObject.put("actionTime",actionTime);
43                //将 JSONObject 对象转换成 String 字符串类型并追加放入输出流对象 fw
44                fw.append(jsonObject.toString()+"\n");
45            }
46            //将输出流对象中的字符串数据写入指定文件
47            fw.flush();
48            //关闭输出流
49            fw.close();
50        }
51        //fulfuill()方法用于填充字符串,若字符串长度为 1,则在字符串首部添加 0
52        public static String fulfuill(String str) {
53            if(str.length() ==2) {
54                return str;
55            } else {
56                return "0" +str;
57            }
58        }
59    }
```

在文件 5-1 中,首先模拟生成用户浏览网页数据,用户浏览网页数据包含 sessionid(用户 Session)、pageid(页面 ID)、userid(用户 ID)和 actionTime(时间)。然后,将数据存储到 JSONObject 对象中。最后,通过 FileWriter 对象将 JSONObject 对象转换成 String 字符串类型,并追加写入目录 D:\\sparkdata 下的文件 user_conversion.json 中。

(2) 右击文件 GenerateData.java,在弹出的快捷菜单中选择 Run.GenerateData.main()运行程序,生成用户浏览网页数据。

(3) 程序运行完成后在 D:\\sparkdata 目录中会生成 JSON 文件 user_conversion.json,该文件包含用户浏览网页数据。

5.3.2 修改 pom.xml 文件

由于实现网站转化率统计是通过 Spark SQL 程序实现的,所以需要在项目 SparkProject 中添加 Spark SQL 依赖。在配置文件 pom.xml 的<dependency>标签中添加如下内容。

```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.11</artifactId>
    <version>2.3.2</version>
</dependency>
```

在文件 pom.xml 中添加依赖后，通过按 Ctrl＋S 组合键保存文件 pom.xml，此时 IntelliJ IDEA 会自动下载文件 pom.xml 中添加的 Spark SQL 依赖。

5.3.3　创建 Spark 连接并读取数据集

在项目 SparkProject 的包 cn.itcast.conversion 中创建文件 PageConversion.java，用于实现网站转化率统计，具体代码如文件 5-2 所示。

文件 5-2　PageConversion.java

```
1    public class PageConversion {
2        public static void main(String[] arg){
3        //实现 Spark SQL 程序
4        }
5    }
```

在文件 5-2 中，第 2～4 行代码创建 main()方法，该方法是 Java 程序执行的入口，在 main()方法中实现 Spark SQL 程序。

在文件 5-2 的 main()方法中创建 SparkSession 对象，用于实现 Spark SQL 程序，具体代码如下。

```
1    SparkSession spark =SparkSession
2                    .builder()
3                    //设置 Application 名称为 page_conversion
4                    .appName("page_conversion")
5                    .getOrCreate();
```

在文件 5-2 的 main()方法中，调用 SparkSession 对象的 read().json()方法读取外部 JSON 文件，将 JSON 文件中的数据加载到 userConversionDS，具体代码如下。

```
Dataset<Row> userConversionDS =spark.read().json(arg[0]);
```

上述代码中，通过变量 arg[0]指定 JSON 文件路径，目的是执行提交 Spark SQL 程序到 YARN 集群运行的命令中，通过参数指定 JSON 文件路径。

在文件 5-2 的 main()方法中，调用 DataSet 的 createOrReplaceTempView()方法，将 userConversionDS 创建为全局临时视图 conversion_table，具体代码如下。

```
userConversionDS.createOrReplaceTempView("conversion_table");
```

上述代码中，创建全局临时视图的目的是之后可以通过 SQL 语句操作

userConversionDS 中的数据。

5.3.4 统计每个页面访问次数

在文件 5-2 的 main()方法中,调用 SparkSession 的 sql()方法统计每个页面访问次数,将统计结果加载到 pageIdPvDS,具体代码如下。

```
Dataset<Row> pageIdPvDS = spark
    .sql("select pageid,count(*) as pageid_count " +
           "from conversion_table " +
           "group by pageid");
```

上述代码中的 SQL 语句,使用 group by 子句和 count()函数相结合的方式,对全局临时视图 conversion_table 的字段 pageid 进行分组聚合操作,从而统计每个页面访问次数。

5.3.5 获取每个用户浏览网页的顺序

在文件 5-2 的 main()方法中,调用 SparkSession 的 sql()方法对每个用户浏览网页的顺序进行排序,将排序结果加载到 useridGroupSortDS,具体代码如下。

```
Dataset<Row> useridGroupSortDS = spark
    .sql("select userid,actionTime,pageid " +
           "from conversion_table " +
           "order by userid,actionTime");
```

上述代码中的 SQL 语句,使用 order by 子句对全局临时视图 conversion_table 的字段 userid 和 actionTime 进行升序排序,从而根据用户访问时间对每个用户浏览网页的顺序进行排序。

在文件 5-2 的 main()方法中,调用 DataSet 的 createOrReplaceTempView()方法,将 useridGroupSortDS 创建为全局临时视图 conversion_group_sort_table,具体代码如下。

```
useridGroupSortDS.createOrReplaceTempView("conversion_group_sort_table");
```

5.3.6 合并同一用户浏览的网页

在文件 5-2 的 main()方法中,调用 SparkSession 的 sql()方法对同一用户浏览的网页进行合并,将合并结果加载到 pageConversionRDD,具体代码如下。

```
JavaRDD<Row>  pageConversionRDD = spark.sql("select userid," +
           "concat_ws(',',collect_list(pageid)) as column2s " +
           "from conversion_group_sort_table " +
           "group by userid").toJavaRDD();
```

上述代码中的 SQL 语句,首先使用 group by 子句根据字段 userid 进行分组,然后使用函数 collect_list()将同一组内的 pageid 合并到一个集合中,最后使用函数 concat_ws()将集合中的每个 pageid 通过分隔符“,”拼接成字符串存储在字段 column2s 中。

为了后续使用 flatMap()算子对合并后的数据进行扁平化处理，这里通过 SparkSession 的 toJavaRDD()方法将 DataSet 转换为 JavaRDD。

5.3.7　统计每个单跳的次数

在文件 5-2 的 main()方法中，使用 flatMap()算子对 pageConversionRDD 进行扁平化处理，根据用户浏览网页的顺序将相邻网页拼接为单跳，将处理结果加载到 rowRDD，具体代码如下。

```
JavaRDD<Row> rowRDD =pageConversionRDD
            .flatMap(new FlatMapFunction<Row, Row>() {
        @Override
        public Iterator<Row> call(Row row) throws Exception {
            List<Row> list =new ArrayList<>();
            //将用户浏览网页数据拆分为数组
            String[] page =row.get(1).toString().split(",");
            String pageConversionStr ="";
            //遍历数组
            for (int i =0;i<page.length-1;i++){
            //判断相邻网页是否相同
            if (!page[i].equals(page[i+1])){
                //将相邻网页通过字符“_”进行拼接
                pageConversionStr =page[i]+"_"+page[i+1];
                //将字符串转换为 Row 对象添加到集合 list
                list.add(RowFactory.create(pageConversionStr));
            }
        }
            return list.iterator();
        }
    });
```

上述代码中，第 7 行通过 row.get(1)获取 Row 对象中的用户浏览网页数据，这是因为 pageConversionRDD 中每一行数据包含 userid 和 column2s，其中 column2s 表示用户浏览网页数据，所以 row.get(1)表示获取字段 column2s，也就是获取用户浏览网页数据。

在文件 5-2 的 main()方法中，调用 SparkSession 的 createDataFrame()方法和 registerTempTable()方法，将存储单跳数据的 rowRDD 注册为临时表 page_conversion_table，具体代码如下。

```
//指定临时表的字段信息
StructType schema =DataTypes
            .createStructType(
                    //创建临时表字段
                    new StructField[]{
                            DataTypes.createStructField(
                                    //指定字段名为 page_conversion
                                    "page_conversion",
                                    //指定字段类型为 String
```

```
                                    DataTypes.StringType,
                                    //指定字段值可以为空
                                    true)});
        spark.createDataFrame(rowRDD, schema)
                .registerTempTable("page_conversion_table");
```

上述代码中，第 13～14 行首先通过 SparkSession 的 createDataFrame()方法，根据单跳数据和临时表的字段信息创建 DataFrame，然后通过 SparkSession 的 registerTempTable()方法将 DataFrame 注册为临时表 page_conversion_table。

在文件 5-2 的 main()方法中，使用 SparkSession 的 sql()方法统计每个单跳的次数，根据统计结果创建全局临时视图 page_conversion_count_table，具体代码如下。

```
spark.sql(
        "select page_conversion," +
        "count(*) as page_conversion_count " +
        "from page_conversion_table " +
        "group by page_conversion")
        .createOrReplaceTempView("page_conversion_count_table");
```

上述代码中的 SQL 语句，使用 group by 子句和 count()函数相结合的方式，对临时表 page_conversion 的字段 page_conversion 进行分组聚合操作，从而统计每个单跳的次数。

5.3.8 计算页面单跳转化率

通过页面单跳转化率的计算公式得知，若要计算单跳 A→B 的页面单跳转化率，首先需要获取页面 A 的访问次数，然后需要获取单跳 A→B 的次数，最终将这两部分数据代入页面单跳转化率的计算公式中计算单跳 A→B 的页面单跳转化率。

在计算每个单跳的页面单跳转化率之前，需要将每个页面与每个单跳一一对应，也就是说如果计算单跳 A→B 的页面单跳转化率，那么代入页面单跳转化率计算公式中的一定是页面 A 的访问次数，而不能是页面 B 或 C 的访问次数。

在文件 5-2 的 main()方法中，使用 SparkSession 的 sql()方法拆分单跳为起始页面和结束页面，将拆分后的数据加载到 pageConversionCountDS，具体代码如下。

```
Dataset<Row> pageConversionCountDS = spark
        .sql("select page_conversion_count," +
                "split(page_conversion,'_')[0] as start_page," +
                "split(page_conversion,'_')[1] as last_page " +
                "from page_conversion_count_table");
```

上述代码中的 SQL 语句，使用 split()函数将单跳字段(page_conversion)中的数据按照分隔符“_”拆分为起始页面字段(start_page)和结束页面字段(last_page)。

在文件 5-2 的 main()方法中，使用 join()算子对 pageIdPvDS(存储每个页面访问的次数)和 pageConversionCountDS(存储每个单跳拆分后的数据)进行连接，根据连接结果创建全局临时视图 page_conversion_join，具体代码如下。

```
pageConversionCountDS
    .join(
            pageIdPvDS,
            new Column("start_page").equalTo(new Column("pageid")),
            "left")
    .createOrReplaceTempView("page_conversion_join");
```

上述代码中，通过 pageConversionCountDS 的起始字段（start_page）和 pageIdPvDS 的页面 ID（pageid）进行左外连接（left）。

在文件 5-2 的 main()方法中，使用 SparkSession 的 sql()计算页面单跳转化率，将计算结果加载到 resultDS，具体代码如下。

```
Dataset<Row> resultDS = spark
.sql("select " +
    "concat(pageid,'_',last_page) as conversion," +
    "round(" +
    "CAST(page_conversion_count AS DOUBLE)/CAST(pageid_count AS DOUBLE)" +
    ",3) as rage " +
   "from page_conversion_join");
```

上述代码中的 SQL 语句，首先使用函数 concat()将每一行数据中的页面 ID（pageid）和结束页面（last_page）通过字符“_”合并为单跳，存储在字段 conversion 中；接着使用函数 CAST()将字段 page_conversion_count（单跳次数）和 pageid_count（页面访问次数）的数据类型转为 DOUBLE 类型；然后通过运算符“/”计算页面单跳转化率；最后使用函数 round()保留计算结果的 3 位小数，并存储在字段 rage 中。

5.3.9　数据持久化

为了保证各网站转化率统计结果的持久性，便于查看以及应用到数据可视化中，这里需要进行数据持久化操作，将网站转化率统计结果存储到 HBase 数据库中，具体实现步骤如下。

在文件 5-2 的 PageConversion 类中添加方法 conversionToHBase()，用于将页面单跳转化率统计结果持久化到 HBase 数据库中，该方法包含参数 dataset，表示需要向方法中传递页面单跳转化率统计结果数据，具体代码如下。

```
    public static void conversionToHBase(Dataset<Row> dataset)
            throws IOException {
        //创建数据表 conversion 和列族 page_conversion
        HbaseUtils.createTable("conversion","page_conversion");
        //创建数组 column,用于指定数据表的列名 convert_page 和 convert_rage
        String[] column = {"convert_page","convert_rage"};
        //遍历页面单跳转化率统计结果数据 dataset
        dataset.foreach(new ForeachFunction<Row>() {
            @Override
            public void call(Row row) throws Exception {
                //获取单跳数据
```

```
12                String conversion = row.get(0).toString();
13                //获取页面单跳转化率数据
14                String rage = row.get(1).toString();
15                //创建数组 value,用于指定数据表的值 conversion 和 rage
16                String[] value = {conversion,rage};
17                HbaseUtils.putsToHBase("conversion",
18                        conversion+rage,
19                        "page_conversion",
20                        column,
21                        value);
22            }
23        });
24    }
```

上述代码中,第 17～21 行调用 HBase 数据库操作工具类的 putsToHBase()方法,用于持久化页面单跳转化率统计结果数据。putsToHBase()方法包含 5 个参数:其中第 1 个参数为字符串 conversion,表示数据表名称;第 2 个参数为字符串对象 conversion+rage,表示数据表的行键;第 3 个参数为字符串 page_conversion,表示数据表的列族;第 4 个参数为数组 column,数组中的每一个元素表示数据表的列名;第 5 个参数为数组 value,数组中的每一个元素表示数据表的值。

在文件 5-2 的 main()方法中,调用 conversionToHBase()方法并传入参数 resultDS,用于在 Spark SQL 程序中实现 conversionToHBase()方法,将页面单跳转化率统计结果数据持久化到 HBase 数据库中的数据表 conversion,具体代码如下。

```
1    //通过 try…catch 抛出异常
2    try {
3          conversionToHBase(resultDS);
4    } catch (IOException e) {
5          e.printStackTrace();
6    }
7    //关闭 HBase 连接
8    HbaseConnect.closeConnection();
9    //关闭 SparkSession 连接
10   spark.close();
```

5.4 运行程序

页面单跳转化率统计程序编写完成后,需要在 IntelliJ IDEA 中将程序封装成 jar 包,并上传到集群环境中,通过 spark-submit 将程序提交到 YARN 中运行,具体步骤如下。

1. 封装 jar 包

由于在封装各区域热门商品 Top3 分析程序 jar 包时,将程序主类指向了 cn.itcast.top3.AreaProductTop3,因此这里需要将程序主类修改为 cn.itcast.conversion.PageConversion。对于封装 jar 包的操作可参照 3.4 节,这里不再赘述。将封装完成的 jar