



下载APP



38 | 当数据遇上AI，Twitter的数据挖掘实战（二）

2022-01-24 徐文浩

《大数据经典论文解读》

课程介绍 >



讲述：徐文浩

时长 12:22 大小 11.33M



你好，我是徐文浩。

在上节课里，我们一起了解了 Twitter 整体搭建数据系统的经验。不过，那一篇论文的主要内容还是在方法论上，一旦我们想要把这个方法论利用到我们当下就在搭建的数据系统里，就有些无从下手的感觉。



不过，好在 Twitter 还发表了很多有着具体实战经验的论文。那么，今天我就请你一起来学习下《[The Unified Logging Infrastructure for Data Analytics in Twitter](#)》这篇文章。在这篇文章里，Twitter 一点儿都没有藏私，而是给出了大量具体的实践技巧，你
可以用“抄作业”的方式，把里面的做法用到自己的系统里。事实上，在我之前搭建的大数据系统中，就从里面吸取了大量的经验。

希望在学习完这节课之后，你可以直接把所看到的具体实战方法用到实践中去。无论是对你现在已有的系统进行对照改进，还是在建设新系统的时候把 Twitter 的方法作为模版，都是一个不错的选择。

统一的用户行为日志和元数据管理

上节课我们就说过，Twitter 为了减少碎片化的日志文件和日志格式，最终在内部启动了一个项目，统一从客户端的视角来记录用户行为日志。这个日志的格式，自然是通过 Thrift 的 Schema 来定义的。而有了这个日志之后，所有的工程师和数据分析师，都可以在一个共识下工作。大部分的数据分析工作，也不再需要大量的 Join 操作，先把数据在 Hadoop 上搬运一遍。

那么，我们就先来看看这个统一的日志格式长什么样子。

首先，Twitter 把用户在 Twitter 内的所有用户行为，拆分成了一个固定的 **6 个层级的结构**，Twitter 称之为 **Click Events**。这 6 个层级，分别是来自于什么客户端（client），是客户端里的哪一个页面（page），是页面里的哪一个 Tab 页（section），是 Tab 页里的哪一个组件（component），是组件里的哪一个元素（element），以及针对这个元素用户的动作（action）是什么。

要注意，这里把这种日志叫做是 Click Events，并不是说我们只是记录用户的点击行为，而是包含了所有的用户行为、特定信息的展示，或许用户只是将鼠标 Hover 在某个 UI 元素上，同样会记录 Click Events。这个 Click Events 里面的 action，就是 Impression 或者 Hover。

我们可以来看一个具体的例子。用户使用 Web 访问 Twitter，在主页的 Mentions 这个 Tab 里的信息流里，点击了某个头像，来获取 Profile 信息，这个事件，就可以通过 web:home:mentions:stream:avatar:profile_click 来定义。

而为了确保全员有一个统一的命名习惯，Twitter 规定了**使用下划线“_”来作为单词的分割**，而不允许采用其他命名方式，比如  驼峰命名法。

组件	描述	例子
client	客户端类型	Web, iPhone, Android
page	一个页面或者功能组	Home, Profile, who_to_follow
section	一个页面里的Tab页或者信息流	Home, Mentions, Retweets, Searches, Suggestions
component	组件或者对象	search_box, Text
element	组件里的UI元素	Button, Avatar
action	实际的用户或者应用的动作	Impression, Click, Hover



Unified Logging的论文中的图1，Twitter把所有的用户行为拆成了一个6个层级的结构

在有了这样一个日志格式之后，任何一个工程师或者数据科学家，都可以很容易地定位到自己想要统计和分析的特定用户行为。

而且，因为整个用户行为的结构是层级化的，我们可以很容易地通过 `通配符` 找到特定类型的行为。比如，通过`web:home:mentions:*`；我们就可以找到用户在 Web 端，访问 Mentions 的 Timeline 里面的所有行为；或者可以通过`*:profile_click`就能找到所有客户端里，所有流程里用户点击 Profile 的行为。

事实上，Twitter 内部有一个叫做 **Oink** 的程序，会按照层级每天自动运行，统计出各个层级 Schema 下的事件数量。而这些数据，会直接展现在内部的 Dashboard 里，方便我们一眼就能看到系统的整体数据。我把对应统计的层级 Schema 放在了下面，你可以去看一下。

复制代码

```

1 (client, page, section, component, element, action)
2 (client, page, section, component, *, action)
3 (client, page, section, *, *, action)
4 (client, page, *, *, *, action)
5 (client, *, *, *, *, action)

```

Twitter 会预先自动给出整个层级结构里，各个层级的事件统计数量。

当然，日志里不光有事件类型，Twitter 的 Click Events 里，还会包含两部分类型的信息，一部分是**标准的字段**，所有的日志都有，包括前面这里的 event_name，以及 event_initiator、user_id、session_id、ip、timestamp，这些字段在任何日志里都是必不可少的。

而对于每个事件可以独有的信息，Twitter 则是通过一个 **event_details** 的字段，用 key-value 对的形式存储下来。这个 event_details 字段，就使得不同团队需要的额外信息，可以有一个统一存放的地方，但是同时不会去影响我们统一的日志的整体格式。

只统一了日志格式还不够，在这个 6 个层级的日志结构下，其实这 6 个层级可能出现的“值”是很多的。

比如，我们到底有多少个不同的“UI 元素”呢？在一个系统每天都在不断迭代的团队里，其实我们是很难知道这一点的。传统的通过文档记录来传递这些信息的方式，往往也是无效的，等我们记得把 UI 元素可能出现的值记录下来的时候，某一个值可能已经废弃，并不再出现了。

而 Twitter 的做法也很简单，就是直接根据现有的数据，**反向**找出它实际会出现哪些值，提供给大家来标记每一个值是什么含义。

我们前面不是已经按照 6 个层级，一一统计了事件的统计数据吗？自然，我们也可以统计，6 个层级结构下，每一个值会有哪些不同的情况出现。Twitter 直接把这个基于数据，反向统计出来的事件层级树状图，做成了一个可展开的 Web 界面。工程师和数据科学家可以直接搜索这个事件树，也可以在具体出现的值上加上 Comments，记录这个字段值是什么意思，把信息共享出来。

以“事实数据”本身快速生成文档，而不是依赖人工去维护文档，能够大大提升我们的工作效率。

巧妙利用 Unicode 和正则表达式

不过，Click Events 只能方便我们做一些描述性的统计信息。而在实际的数据分析中，很多时候，我们重视的是**特定的用户行为模式** (pattern)。

我拿电商网站来举个例子。我们常常会去做 **漏斗分析**，看用户在浏览商品—> 加入购物车—> 进入支付—> 填写地址，到最终支付完成的整个环节里每一个步骤的统计数据，他们分别是在哪一个步骤流失的。

而且，我们也需要经常根据不同的需求，重新定义整个漏斗环节。比如，新上了一个新年促销的营销活动，我们就要去看这个营销活动的漏斗分析的数据。**根据用户的一个个 Session，去分析特定的行为模式，这个在用户行为日志的分析里，是一个非常高频的动作。**

当然，我们有原始的 Click Events，就可以写一段 Java 程序，或者一个复杂的 Pig 脚本或者 SQL 来完成分析。不过，这样我们每一次的工作显然没有被“复用”起来，而且每一个脚本，都需要根据用户的标识，去 Join 大量的数据。

而 Twitter 采取了一个非常巧妙的方式来解决了这个问题。首先，它用了一个数据处理程序，把所有的用户行为串成了一个个的用户 Session。然后，它并没有简单地把原始的数据存储下来，而是对数据进行了“**信息压缩**”。

Twitter 把每一个 web:home:mentions:stream:avatar:profile_click 这样的事件名称映射到了一个 Unicode 的字符串。然后，整个 Session 里，把用户按照时间先后顺序进行的这些事件组合在一起，就是一个长字符串，叫做 **session_sequence**。

于是，一个用户 Session 就可以简化成一条日志，里面会包含这些字段：

user_id，也就是标识哪一个用户；

session_id，也就是标识是哪一个 session；

ip，也就是用户访问时的 ip 地址；

session_sequence，也就是代表了用户顺序执行的一系列动作；

duration，也就是这个 session 的时长。

有了这个 session_sequence，我们的漏斗分析就非常简单了。我们只需要通过一个简单的**正则表达式**，就能定义出一个用户特定的行为流。

比如，我们用 `i` 代表用户浏览了单个商品，`a` 代表用户把商品加入了购物车。那么通过正则表达式 `.*i.*`，我们就能找到所有浏览过单个商品的 session，通过 `.*i.*a.*` 就代表了先浏览过商品，再把商品加入过购物车的 session。

两个数据一比较，我们的漏斗数据就计算出来了。而因为正则表达式的灵活性，我们不需要预先定义要分析什么样的漏斗，而是可以在需要的时候，写个正则表达式就完事儿了。并且在实践中，Twitter 封装好的 **Pig UDF**，让你把这一步也省下了，你只要自定义一系列的事件流，就能快速查找符合这个执行顺序的 Session，以及他对应的漏斗。

 复制代码

```
1 define Funnel ClientEventsFunnel('$EVENT1' '$EVENT2', ...);
```

你可以通过一个 UDF 定义一个你想要分析的漏斗。

 复制代码

```
1 (0, 490123)
2 (1, 297071)
3 ...
```

这个 Funnel 的计算结果就是这个事件流的每一层漏斗剩下的 session 数量。

而且，这个数据，还能够直接拿来进行机器学习的建模。我们完全可以把这样按事件顺序发生的序列，通过 [马尔可夫链](#) 或者类似的时序分析方案来进行建模，去预测在什么样的用户行为之后，会发生什么样的用户行为。

这样，我们去进行预测类的机器学习建模，也只是一个 Pig 脚本的事情了。

方便 debug 很重要

除了统一的日志格式和预生成的 `session_sequence` 之外，我们还经常会遇到的一类问题，是通过日志 **debug 在线上遇到的各种问题**。通过日志进行生产环境的 debug，是最常见的需求之一。但是，对于原始的 Click Events 日志来说，想要找到特定的日志虽然容易但是开销很大。

我们想要找到特定的日志，其实并不麻烦，通过 Pig 写个脚本，加上我们需要的筛选条件跑一下，就能找到这样的日志。但是，因为每一个 Pig 任务都是在完整地扫描一遍日志，这个开销可不小。

而且，在 debug 的过程中，这个是一个**交互式**的过程。我们可能先写了一个筛选条件，然后查询到了一部分日志，但是这些日志可能并不符合我们的需要。于是我们又变化了一下筛选条件，这样就要再完整地扫描一遍日志。可能为了调查一个小小的线上问题，我们把数十 TB 的原始日志扫了又扫，如果算算实际的硬件开销，那可绝对小不了。

Twitter 的解决办法也很简单，那就是**为日志文件生成索引**。不过，这个索引和关系数据库不一样，它不是到每一条的日志这个级别的，而是到每一个 Block 级别。Twitter 的日志是用  LZO 的压缩算法压缩的，它的索引，也只是确认特定字段的值，是否在整个 Block 中出现过。

这样，索引的尺寸可以大大减少，毕竟如果把每一条日志所在的位置都建立索引，那么索引的尺寸可能就和日志一样大了，并起不到减少开销的作用。

而有了这个索引文件以后，我们就可以在查询特定日志的时候，通过索引大幅度减少需要扫描的数据。事实上，Twitter 专门开源了一个叫做  elephant-twin 的项目。你只要在 Pig 脚本中，加载索引、对应的 MapReduce 任务，它就会自动只读取索引找到的对应 Block，从而就减少了需要扫描的数据。

在实践过程中，我还这样做过。我们干脆直接基于特定字段的索引，比如 user_id，提供了一个在线查询特定用户行为的日志的**内部站点**。通过这个站点，数据科学家们能够直接观察用户的行为模式来寻找灵感。虽然这样的服务，需要用户对于 HDFS 进行随机访问，其实并不符合 HDFS 设计的数据访问模式。但是，因为**并发量很小**，对于 HDFS 并不会造成什么压力。

其实，这样的技巧、方法和工具，在大数据领域非常实用，能够大大提升我们解决问题的效率。

小结

通过这节课的学习，我们其实可以看到，在优秀的基础设施、杰出的团队理念之后，一个基于用户行为日志的大数据系统，是怎么落到实处的。

在数据处理层面，Twitter 通过统一格式的、层级化的日志，使得日常的描述性数据分析变得非常容易，这可以大大缩减新人想要使用系统的上手成本。

通过对数据进行预处理变成 `session_sequence`，并将一个个事件映射成 Unicode，Twitter 使得基于用户时序行为的分析也变得非常容易。无论是进行漏斗分析，还是用户建模，都变成了简单 Pig 脚本可以完成的事情。

进一步地，为了方便系统开发人员排查问题，Twitter 还为处理完的日志建立了索引。这就让我们在海量日志中进行一些 debug 工作和特定案例的分析，也变得特别容易了。

希望通过这节课，你能够一窥 Twitter 这样公司的内部数据分析系统，具体是怎么样的。这也是大量工程师在实践过程中，找到的一系列最佳实践。如果你刚开始为你的团队和产品搭建大数据体系，完全可以依样画葫芦地实践一遍。

推荐阅读

作为一个解读大数据论文的专栏，这节课最后要给你推荐的资料，仍然是我们今天所讲到的三篇论文。虽然和 Bigtable、Spanner 这样石破天惊的新系统比起来，这些论文只是一些实践中的应用系统和策略。不过，这样的论文对于大部分实际是在应用大数据系统，完成业务目标的我们来说，也有着更强的借鉴意义。

也许，把某一个你想到的解决方案抽象总结出来，能发表出来，就可以让更多周围的人受益。我把论文的链接放在了下面，你可以去阅读参考一下。

[🔗 The Unified Logging Infrastructure for Data Analytics in Twitter](#)

[🔗 Full-text indexing for optimizing selection operations in large-scale data analytics.](#)

[🔗 Discovering similar users on twitter](#)

思考题

Twitter 为了让团队可以快速检索日志，在日志文件的 block 层面建立索引，也开源了对应的 elephant-twin 项目。不过，这个项目今天其实早就不再使用了。你能想一想这是为什么吗？

欢迎在留言区说说你的观点和见解，也欢迎你把今天的内容分享给更多的朋友。

分享给需要的人，Ta购买本课程，你将得 **20** 元

 生成海报并分享

 赞 0

 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 37 | 当数据遇上AI, Twitter的数据挖掘实战（一）

下一篇 39 | 十年一梦，一起来看Facebook的数据仓库变迁（一）

更多学习推荐

190 道大数据高频面试真题

涵盖 11 个核心技术栈 + 4 套大厂真题

免费领取 



精选留言 (2)

 写留言



CRT

2022-02-05

是因为这样的索引不方便数据重新分配，代价太大？



**piboye**

2022-01-24

clickhouse 好像也是这样在block级别建立索引

