### 数据清洗经验

平时习惯了在某些特定的[数据](http://www.techxue.com/dutu/dianshangshuju/)集合上做实验，简单的tokenization、预处理等步骤就足够了。但是在数据越来越大的年代，数据清洗越来越重要，也越来越复杂。看到Philip J.Guo 的这篇英文文章《Parsing Raw Data》觉得不错，学习并译成中文，难免谬误，仅供参考。

**前言**

科研工作者、工程师、业务分析者这些和数据打交道的职业，[数据分析](http://techxue.com/fenxi/)在他们工作中是一项核心任务。这么不仅仅针对“[大数据](http://www.techxue.com/search.php?mod=portal&kw=%B4%F3%CA%FD%BE%DD)”的从业者，即使你笔记本硬盘上的数据也值得分析。数据分析的第一步是洗数据，原始数据可能有各种不同的来源，包括：

1、Web服务器的日志

2、某种科学仪器的输出结果

3、在线调查问卷的导出结果

4、1970s的政府数据

5、企业顾问准备的报告

这些来源的共同点是：你绝对料想不到他们的各种怪异的格式。数据给你了，那就要处理，但这些数据可能经常是：

1、不完整的（某些记录的某些字段缺失）

2、前后不一致（字段名和结构前后不一）

3、数据损坏（有些记录可能会因为种种原因被破坏）

因此，你必须经常维护你的清洗程序来清洗这些原始数据，把他们转化成易于分析的格式，通常称为data wrangling。接下来会介绍一些关于如何有效清洗数据，所有介绍的内容都可以由任意编程语言实现。

**使用断言**

这是最重要的一点经验：使用断言(Assertions)揪出代码中的bug。用断言的形式写下你对代码格式的假设，如果一旦发现有数据跟你的断言相悖，就修改这些断言。

记录是有序的？如果是，断言之！每一条记录都是有7个字段么？如果是，断言之。每一个字段都是0-26之间的奇数么？如果是，断言之！总之，能断言的都断言！

在理想世界中，所有记录都应该是整整齐齐的格式，并且遵循某种简洁的内在结构。但是实际当中可不是这样。写断言写到你眼出血，即便是出血还得再写。

洗数据的程序肯定会经常崩溃。这很好，因为每一次崩溃都意味着你这些糟糕的数据又跟你最初的假设相悖了。反复的改进你的断言直到能成功的走通。但一定要尽可能让他们保持严格，不要太宽松，要不然可能达不到你要的效果。最坏的情况不是程序走不通，而是走出来不是你要的结果。

**不要默默的跳过记录**

原始数据中有些记录是不完整或者损坏的，所以洗数据的程序只能跳过。默默的跳过这些记录不是最好的办法，因为你不知道什么数据遗漏了。因此，这样做更好:

1、打印出warning提示信息，这样你就能够过后再去寻找什么地方出错了

2、记录总共跳过了多少记录，成功清洗了多少记录。这样做能够让你对原始数据的质量有个大致的感觉，比如，如果只跳过了0.5%，这还说的过去。但是如果跳过了35%，那就该看看这些数据或者代码存在什么问题了。

使用Set或者Counter把变量的类别以及类别出现的频次存储起来

数据中经常有些字段是枚举类型的。例如，血型只能是A、B、AB或者O。用断言来限定血型只能是这4种之一虽然挺好，但是如果某个类别包含多种可能的值，尤其是当有的值你可能始料未及的话，就不能用断言了。这时候，采用counter这种数据结构来存储就会比较好用。这样做你就可以：

1、对于某个类别，假如碰到了始料未及的新取值时，就能够打印一条消息提醒你一下。

2、洗完数据之后供你反过头来检查。例如，假如有人把血型误填成C，那回过头来就能轻松发现了。

**断点清洗**

如果你有大量的原始数据需要清洗，要一次清洗完可能需要很久，有可能是5分钟，10分钟，一小时，甚至是几天。实际当中，经常在洗到一半的时候突然崩溃了。

假设你有100万条记录，你的清洗程序在第325392条因为某些异常崩溃了，你修改了这个bug，然后重新清洗，这样的话，程序就得重新从1清洗到325391，这是在做无用功。其实可以这么做：1. 让你的清洗程序打印出来当前在清洗第几条，这样，如果崩溃了，你就能知道处理到哪条时崩溃了。2. 让你的程序支持在断点处开始清洗，这样当重新清洗时，你就能从325392直接开始。重洗的代码有可能会再次崩溃，你只要再次修正bug然后从再次崩溃的记录开始就行了。

当所有记录都清洗结束之后，再重新清洗一遍，因为后来修改bug后的代码可能会对之前的记录的清洗带来一些变化，两次清洗保证万无一失。但总的来说，设置断点能够节省很多时间，尤其是当你在debug的时候。

**在一部分数据上进行测试**

不要尝试一次性清洗所有数据。当你刚开始写清洗代码和debug的时候，在一个规模较小的子集上进行测试，然后扩大测试的这个子集再测试。这样做的目的是能够让你的清洗程序很快的完成测试集上的清洗，例如几秒，这样会节省你反复测试的时间。

但是要注意，这样做的话，用于测试的子集往往不能涵盖到一些奇葩记录，因为奇葩总是比较少见的嘛。

**把清洗日志打印到文件中**

当运行清洗程序时，把清洗日志和错误提示都打印到文件当中，这样就能轻松的使用文本编辑器来查看他们了。

可选：把原始数据一并存储下来

当你不用担心存储空间的时候这一条经验还是很有用的。这样做能够让原始数据作为一个字段保存在清洗后的数据当中，在清洗完之后，如果你发现哪条记录不对劲了，就能够直接看到原始数据长什么样子，方便你debug。

不过，这样做的坏处就是需要消耗双倍的存储空间，并且让某些清洗操作变得更慢。所以这一条只适用于效率允许的情况下。

最后一点，验证清洗后的数据

记得写一个验证程序来验证你清洗后得到的干净数据是否跟你预期的格式一致。你不能控制原始数据的格式，但是你能够控制干净数据的格式。所以，一定要确保干净数据的格式是符合你预期的格式的。

这一点其实是非常重要的，因为你完成了数据清洗之后，接下来就会直接在这些干净数据上进行下一步工作了。如非万不得已，你甚至再也不会碰那些原始数据了。因此，在你开始数据分析之前要确保数据是足够干净的。要不然的话，你可能会得到错误的分析结果，到那时候，就很难再发现很久之前的数据清洗过程中犯的错了。