

用Hadoop创建数据分析应用

敏捷数据 科学

AGILE DATA SCIENCE



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

[美] Russell Jurney 著
冯文中 朱洪波 译

内 容 简 介

本书面向大数据挖掘，以敏捷视角呈现高效构建数据模型的全程实践和思路。在一组以一个真实电子邮箱数据挖掘为例的数据 - 价值金字塔进阶模式中，你将学到：一整套实用工具及其方法论，可快速实现在 Hadoop 上构建数据分析应用；用 Python、Apache Pig 及 D3.js 等轻量级工具创建用于探索数据的敏捷环境；一种可根据数据中信息快速切换，进行不同类型数据分析的迭代式开发方法。

本书适合所有与数据工作相关的从业者，同时也适合有志成为数据科学工作者的广大读者作为入门读物。

© 2014 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and Publishing House of Electronics Industry, 2014. Authorized translation of the English edition, 2014 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书简体中文版专有版权由 O'Reilly Media, Inc. 授予电子工业出版社。未经许可，不得以任何方式复制或抄袭本书的任何部分。专有版权受法律保护。

版权贸易合同登记号 图字：01-2013-9391

图书在版编目（CIP）数据

敏捷数据科学：用 Hadoop 创建数据分析应用 / (美) 朱尔尼 (Jurney,R.) 著；冯文中，朱洪波译. —北京：电子工业出版社，2014.7

书名原文：Agile data science

ISBN 978-7-121-23619-8

I . ①敏… II . ①朱… ②冯… ③朱… III . ①数据采集 IV . ① TP274

中国版本图书馆 CIP 数据核字 (2014) 第 135213 号

责任编辑：张春雨

印 刷：北京丰源印刷厂

装 订：三河市鹏成印业有限公司

出版发行：电子工业出版社

北京市海淀区万寿路173信箱 邮编：100036

开 本：787×980 1/16 印张：11.5 字数：240千字

版 次：2014年7月第1版

印 次：2014年7月第1次印刷

定 价：49.00元

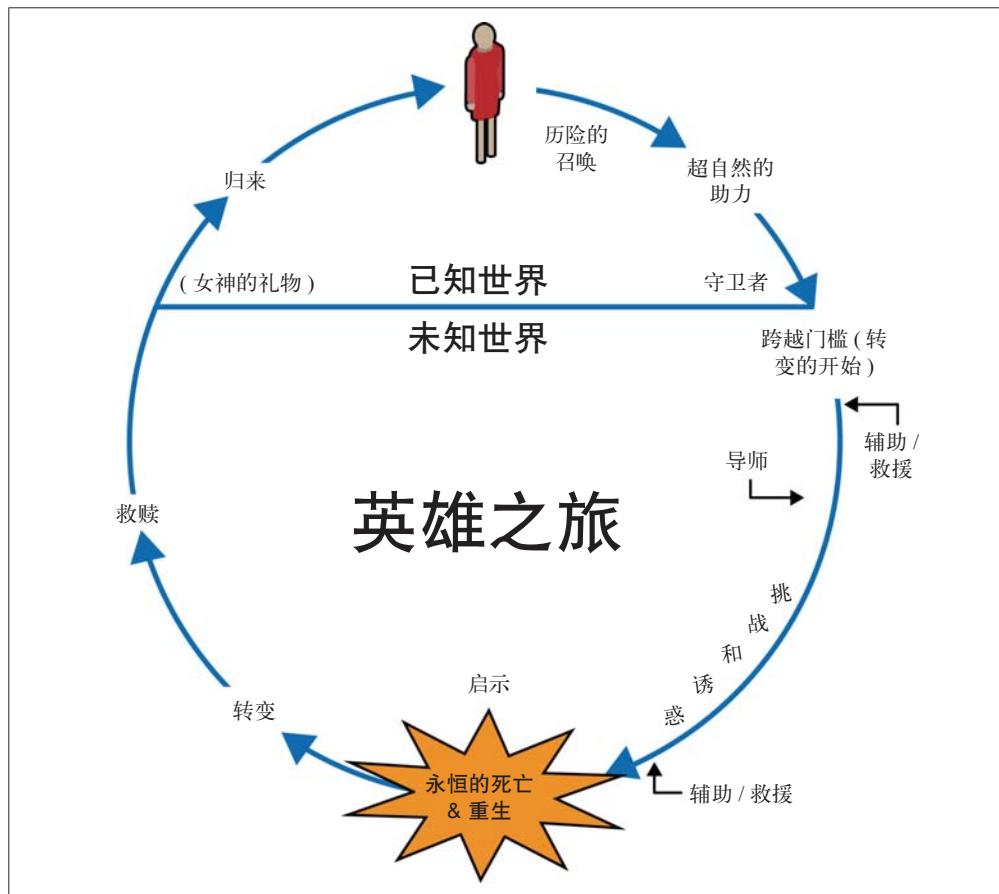
凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至zlt@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：(010) 88258888。

第 1 部分

起步



图I-1. 英雄之旅, ^{译注¹}来源: 维基百科

译注 1: 美国神话学家 Joseph Campbell 在其著作 *The Hero With a Thousand Faces* 中对各种神话故事剧情加以总结, 提出“英雄之旅 (The Henro's Journey)” 剧情发展理论。作者引用“英雄之旅”隐喻本书的敏捷数据科学开发流程。更多内容请参考 : <http://en.wikipedia.org/wiki/Monomyth>

第 1 章

理论

我们一直在实践中探寻更好的软件开发方法，身体力行的同时也帮助他人。由此我们建立了如下价值观：

- 个体和互动**高于流程和工具
- 可用的软件**高于详尽的文档
- 客户合作**高于合同谈判
- 响应变化**高于遵循计划

在每一对比较中，后者并非全无价值，但我们更看重前者。——《敏捷宣言》

敏捷大数据

敏捷大数据是一种方法论，使得在创建具有可扩展性的数据分析应用的过程中，能有效应对各种不确定性。它也是一份用 Hadoop 挖掘大数据价值的操作指南。

仓储级 (warehouse-scale) 计算赋予了我们巨大的存储和计算资源，以解决存储和处理前所未有的大规模数据。使用新的工具，解决之前看来非常困难的问题、从原始数据中得到全新的产品、从原始数据中获取有价值的洞察，并产品化为新型的分析应用，都已引起人们的极大兴趣。这些工具包括处理器、磁盘，加上可视化、统计和机器学习。这就是数据科学。

在近二十年里，万维网已经成为信息交换的主要媒介。在此期间，软件工程也在如何设计、构建、维护软件等方面被“敏捷”所革新。在多个领域里，新的敏捷流程让更多的项目与产品在预算内按时交付，也让更小的团队或者个人能开发完整的应用。这就是敏捷软件开发。

但依然有问题：与实际数据打交道、开展数据科学工作、进行严谨的研究需要很多时间——比敏捷开发的周期要长（通常以月计）。这比很多组织为一个项目分配的迭代周期要长得多，也意味着现在的应用研究人员承受着更大的时间压力。数据科学也被困在老派的软件开发流程——瀑布模型上。

问题和机遇在这两个交会的趋势里到来：数据科学作为应用研究，需要大量的工作——我们该如何按照难以预料的时间表开展数据科学工作，并和敏捷式软件开发结合起来呢？如何用比（被束之高阁的）瀑布模型更好的方式开发分析应用？又如何为未知的、不断演化的数据模型建立分析应用呢？

本书尝试综合敏捷开发和大数据科学，在研究与工程之间达成一种富有成效的关系。为了达到这个目标，本书展示了一个轻量级工具箱，以应对不确定的、不断变化的数据海洋。本书会不断向你介绍如何迭代地利用这个软件栈创造价值，重拾灵活性，并挖掘数据将其转化成财富。

敏捷大数据旨在帮助你重拾主动权，确保你的应用研究能产出满足用户真实需求的产品。

Big Words 定义

可扩展性，*NoSQL*，云计算，大数据——这些大数据领域中具有争议的术语，我们在敏捷式大数据情景下对它们做出定义。

可扩展性

可扩展性表示按需扩大 / 缩小某些操作的简单程度。在敏捷大数据里，它表示当应用的负载和复杂度线性增长时，软件工具和技术能够以亚线性的代价或复杂度增长。无论是大量还是少量数据，都用同样的工具进行处理。我们信奉一次构建，而非不断重复开发的方法论。

NoSQL

NoSQL 是“Not Only SQL”的缩写，它意味着我们远离了在完整的关系数据库里面存储结构化数据的约束。它也意味着我们超越了为在线事务处理（Online Transaction Processing）优化的工具，将它扩展为可用于在线分析处理（Online Analytic Processing）的工具集。从结构分析和算法的角度看，它们能使数据更易于访问。它还意味着摆脱了配备昂贵存储系统的单台机器，转而用可随用户和负载增长而线性扩展的并发系统。它更意味着我们不会因为数据库的问题而停下来，不断痛苦地调优、分片来缓和。

我们用到的 NoSQL 工具是 Hadoop，一个高度并行的批处理系统，以及 *MongoDB*，一个分布式的文档数据库。

云计算

云计算意味我们使用 Amazon Web Services 等类似服务商提供的基础设施服务，把数据中心抽象成计算机，在此之上构建分析应用。作为应用开发者，使用云计算可以在构建可扩展的应用时，避免陷入基础设施的细枝末节而停滞不前。

大数据

有一个信奉如下理念的市场：在现在和将来，通过一些关键的业务系统，对不断增长的事务日志进行聚合后，可以从中获取巨大的价值。这就是大数据。大数据系统可以使用本地存储，商业服务器硬件，以及免费的开源软件，以低成本处理数据，某种程度上使记录和处理海量原子记录变得切实可行。



Metamarkets 的共同创始人与架构师 Eric Tschetter，对 NoSQL 实践有以下的看法：

“我将 NoSQL 定义为一场对存储和查询层进行组合以满足特定用例的运动。关系型数据库是非常通用的武器，它能在一定用量下满足任意的存储和查询需求。我将 NoSQL 视为一次向其他存储架构的转移：它针对某些特定的用例场合进行了优化，并能在上述场合确立操作复杂性等指标上的优势。”

敏捷大数据团队

产品是由团队构建出来的，而敏捷方法重视人本身多于过程。因此敏捷大数据从搭建团队做起。

数据科学是一个宽泛的学科，横跨分析，设计，开发，商业和研究等领域。在一个谱系上定义敏捷大数据团队成员角色，从客户到运维，大致会如图 1-1 所示：

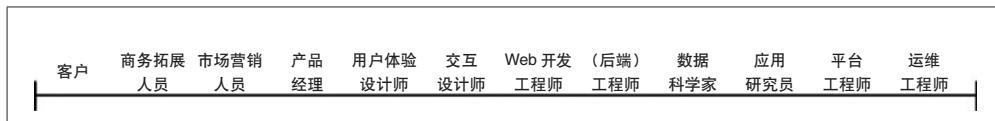


图1-1. 敏捷大数据团队中的成员角色

这些角色分别被定义为：

- **客户** 使用你的产品，点击按钮或者链接，也或者是完全无视你的产品。你的任务是为他们不断创造价值。他们对产品的兴趣决定了产品的成功与否。
- **商务拓展人员** 直接的或者通过创建登陆页和推广活动，拓展早期客户。他们将产品推向市场。
- **市场营销人员** 与客户进行交流以决定进入哪个市场。在一个敏捷大数据项目开始时，他们决定初始的定位。
- **产品经理** 吸收每个人的观点，将它们综合起来，让团队在产品的方向与愿景上达成共识。
- **用户体验设计师** 负责对数据进行设计，令它能够符合用户的视角。这个角色非常重要，因为统计模型的结果有时很难被对模型结果缺乏理解的“普通”用户接受（例如，怎么解释一件事有 75% 的可能性为真？）
- **交互设计师** 围绕数据模型进行交互式设计，让用户能够发现数据的价值。
- **Web 开发工程师** 创建将数据送达浏览器的 web 应用。
- **(后端) 工程师** 构建将数据送达应用的系统。
- **数据科学家** 用新颖的方法对数据进行探索和变换，以实现和发布一些新的特性，并将多个不同来源的数据组合到一起，创造出新的价值。数据科学家会尽早并经常与研究人员、后端工程师、web 开发工程师、设计师一起创建可视化，来展示原始数据、中间数据和最终数据。
- **应用研究员** 解决数据科学家揭露的那些阻碍创建价值的重要问题。解决这些问题需要投入巨大的精力和时间，也需要用到统计学与机器学习中的新颖方法。
- **平台工程师** 解决分布式系统的问题，让敏捷大数据项目易于扩展。平台工程师会处理工作单，解决直接影响系统的故障，并实现长期的计划和项目，为研究人员、数据科学家、后端工程师保证并提高可用性。
- **运维工程师** 确保能顺畅的设定并操作线上的数据设施。他们把部署过程自动化，并且在有问题发生的时候进行响应。

认识机遇和问题

构建数据产品所需的庞大技能集合既展示了机遇，也暴露了问题。假如这些技能都在团队里面能找到对应的专家，那么就可以分解问题，直接解决。这样数据科学就是一条高

效的流水线，如图 1-2 所示。

但是，为了满足多个分散领域的专业技能需求，增大团队规模，会迅速增加沟通的成本。距离客户有 8 个人的研究员，很可能在解决一些神秘莫测而不是客户真正相关的问题。同时，12 个人的团队会议也难以高效。也可以将这些团队分成多个部门，在部门间建立交付约定，但这又会影响敏捷和团队的内聚力。在等待研究结果的同时，我们开始写规格说明书，很快就会发现又回到瀑布模型了。

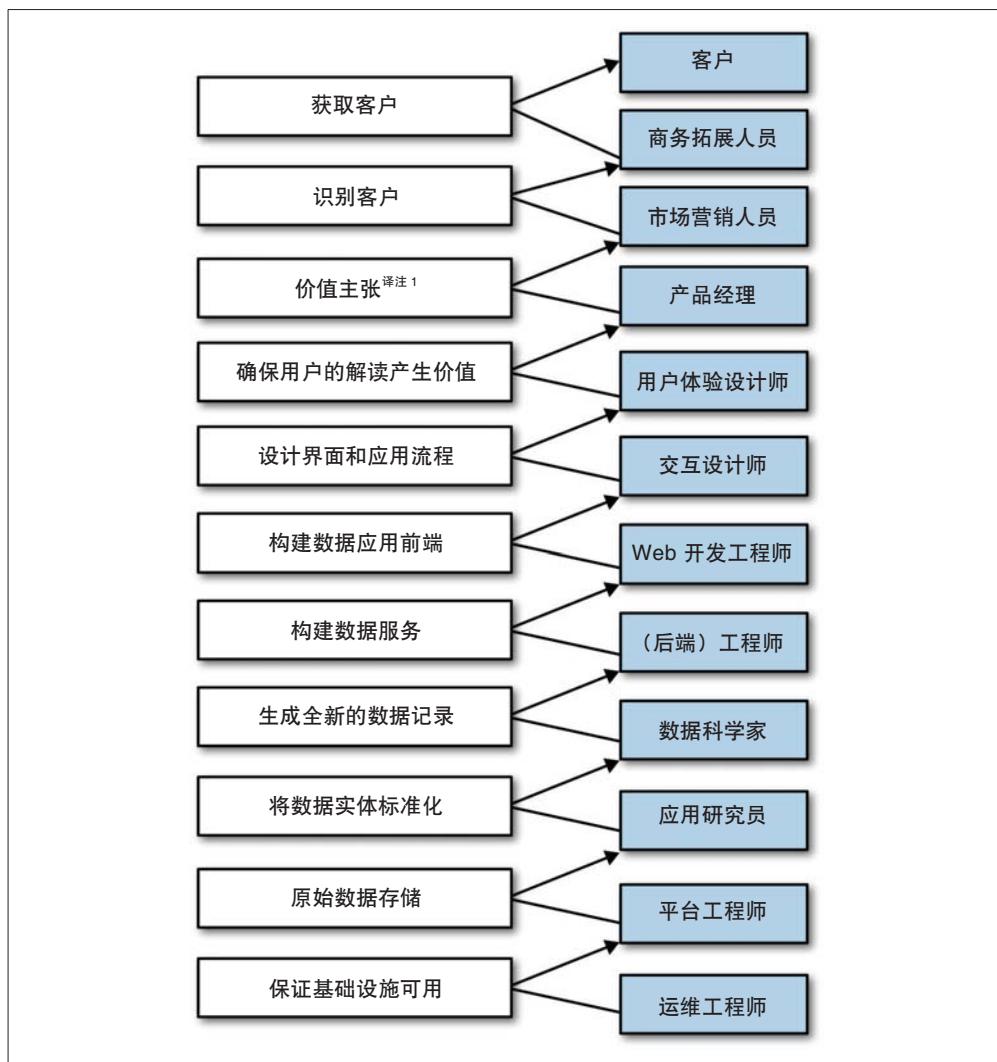


图1-2. 专家贡献者工作流

然而我们知道，保持敏捷，以及对产品的一致愿景和共识，是产品成功的关键。最糟糕的情况就是一个团队为着不同的愿景工作。要如何协调扩大了的专业技能跨度，以及应用研究、数据科学、软件开发和设计等任务之间各不相同的时间表呢？

响应变化

要保持敏捷，需要适应并拥抱这些变化。要采取措施，按照精益方法保证生产效率。

译注1：见 http://en.wikipedia.org/wiki/Value_Proposition

特别地，下面的这些变化能够让我们回归敏捷：

- 通才高于专才
- 小团队高于大团队
- 使用高阶工具和平台：云计算，分布式系统，PaaS
- 持续、迭代地分享工作成果，即使这些工作尚未完成

在敏捷大数据中，一个小型的通才团队会使用可扩展，高阶的工具与云计算来迭代地从数据中挖掘出越来越大的价值。我们会使用一个利用了云计算，分布式系统和PaaS的软件栈。然后利用这个栈去迭代发布通过深入研究得出的中间数据。我们会从简单的数据开始，一直到进行预测和指导行动，像滚雪球一样，不断创造价值，将数据转化为财富。下面逐一介绍。

利用通才的技能

在敏捷大数据中，通才比全才更有价值，如图 1-3 所示。

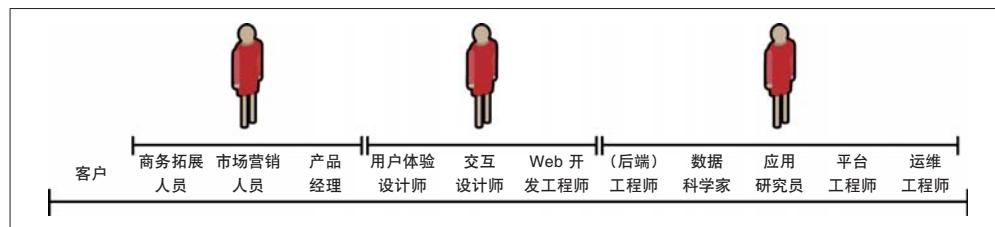


图1-3. 在敏捷大数据团队里面，人员的角色更为宽泛

换句话说，我们把成员的技能广度和他们的知识深度、其他领域的才华看得一样重要。好的敏捷大数据团队成员包括了：

- 能产出 CSS 的设计师
- 具备用户界面和用户体验方面的知识，并能构建整个 web 应用的 web 工程师
- 能胜任科研、搭建 web 服务和应用的数据科学家
- 能提交可用代码，解释数据结果并分享数据的研究员
- 能理解各个领域里细微差别的产品经理

在敏捷大数据团队里面，设计师是一个特别重要的角色。设计不单单是应用的外观或者体验，还会涵盖产品的各个方面，如架构、发布、用户体验、工作环境等等。



在纪录片 *The Lost Interview* 中，史蒂夫乔布斯提到了他的产品设计观：

“设计一个产品就是将 5000 样东西同时放入大脑，然后用新的方法把他们组合起来，最终得到你想要的。每天你都会发现一些新的事物，将它们用不同的方式进行组合，就可能产生新的问题或者机会。过程才是最奇妙的。”

充分利用敏捷平台

在敏捷大数据中，我们用最易于使用的分布式系统、云计算和 PaaS，以最小化构建基础设施的成本，并最大化生产力。软件栈的简洁性让我们可以回归敏捷。我们会用尽可能少的步骤在这个栈上构建可扩展的系统。这让我们快速前进并处理所有可用数据，而不会遭遇系统扩展性的问题，以致丢弃数据或者需要在系统运行时重新构建。这就是说，只需要构建一次系统。

分享中间结果

最后，为了应对研究人员、数据科学家以及团队其他成员在时间表上的差异，我们引入一种叫“数据拼贴画”的机制来弥补。换句话说，我们将丰富的视图，可视化和特性组合起来，塑造出应用的功能。

尽管研究人员和数据科学家的工作通常需要比常规的敏捷迭代周期更长的时间，但他们应当每天发布一份数据——即使这份数据还不能真正地发布。在敏捷大数据中，没有不能发布的状态。团队的其他人必须能够看到每周（每天或者更短当然更好）的数据状态更新。这种与研究人员的互动在统一团队进度以及项目管理上是非常重要的。

这意味着需要发布中间结果——不完整的数据，数据分析的草稿，等等。它们就像胶水一样能够将团队凝聚起来，有了这些数据，每个人都可以得知数据的本质、研究的进度以及如何将这些数据转化成有用特性。开发和设计工作需要这些共享的实际信息才能继续。一开始数据可以先发布给小范围的人，到后来数据质量越来越高，再发布给更多人（见图 1-4），但客户一定要迅速介入。

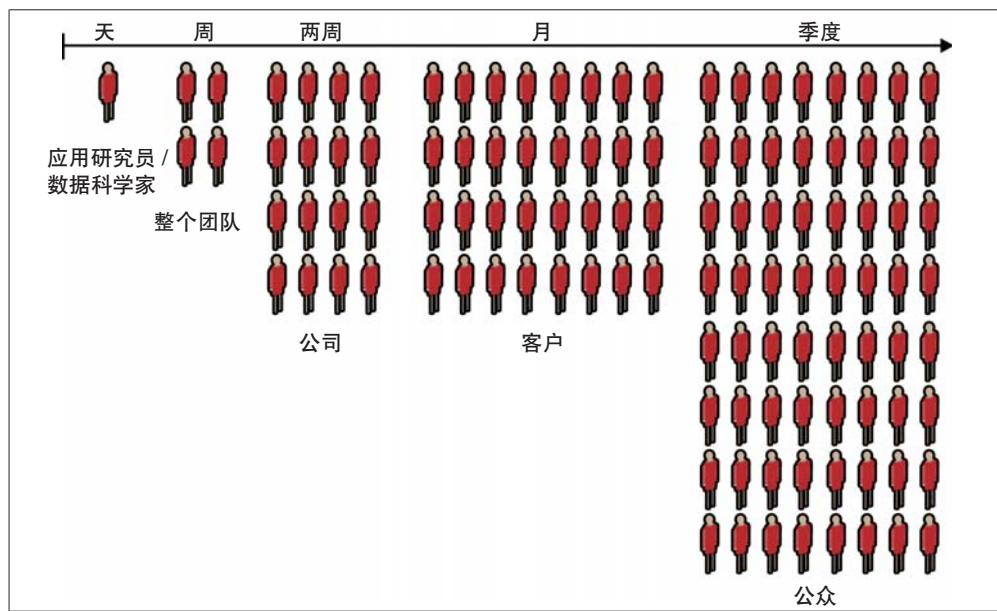


图1-4. 从概念性的数据到正式上线过程中读者群的增长

敏捷大数据流程

敏捷大数据流程利用了数据科学的迭代性本质和高效的工具，从数据中构建和抽取高阶的结构和价值。

数据产品团队技能多样，会产生多种可能性。由于团队覆盖了大量的领域，构建 web 产品也自然是一个协作的过程。团队需要方向才能协作：每个成员都应该热情饱满而又顽强地追求一个共同的目标。要明确这个方向，需要一个共识。

在协作中达成共识是开发软件过程中最难的一个环节。软件开发团队最大的风险就是根据不同的蓝图进行开发。相互抵触的愿景会让产品缺乏专注，最终失败。

有时在实际开发应用之前会做一些样品（mock）：产品经理进行市场调查，设计师根据目标用户的反馈不断改进这个样品。这些样品可以作为团队共享的蓝图。

即使数据本身是不变的，随着对用户的了解以及外界条件的改变，真实世界中的需求也会变化。所以蓝图也需要随着时间而变化。而敏捷方法就是为了更好的实现不断变化的

需求，并尽快将样品转化成真正能运行的系统而发明的。

典型的 web 产品是由表格驱动的，在后端由数据库中可预料、有约束的事务数据支撑，这和数据挖掘产品有根本上的差异。在 CRUD 应用中，数据相对一致。数据模型是可以预知的 SQL 表格或者文档，对它们进行改动是产品层面的决策。数据的“见解”则是不相关的，产品团队可根据意愿构建模型以符合应用的商业逻辑。

而对于由数据挖掘驱动的、可交互的数据产品，以上任何一条都不成立。现实数据都是脏的，要挖掘就要面对脏数据。假如数据不脏，那就不是数据挖掘了。即使是精心抽取、提炼出的信息，也可能是模糊的、不可预测的。将它们展示给消费者，还需要大量的工作和十分的细心。

对于数据产品，数据是冷酷无情的。无论希望数据能表达什么，数据对我们本身的意愿压根毫不关心，它只陈述事实。这意味着瀑布模型没有用武之地。也意味着，样品也是一个为了在软件团队中建立共识但不全面的蓝图。

数据产品的样品是应用程序的规格说明书，它没有产品最重要的特色——具有真正价值的信息。这些作为蓝图的样品会对复杂的数据模型做出毫无依据的假设。面对一个建议清单，样品经常会误导我们。一旦加上成熟的交互，样品甚至会抑制真相，放大假设。然而我们知道好的设计和用户体验就是要最小化假设。那该如何是好？

敏捷产品开发的目标是辨识出产品最根本的特性，将这个特性先实现了，然后再添加其他特性。这将敏捷带到了项目里，让项目更有可能满足产品进化过程中最真实、最根本的需求。在数据产品中，最根本的特性会给人惊喜。假如不是这样，要么是你做错了，要么是你的数据没有太大意义。信息有它的背景，如果背景易变，就无法使用洞察进行预测。

代码检查和结对编程

要避免系统性错误，数据科学家会定期将他们的代码共享给团队成员。所以，代码检查是很重要的。通过检查，可以很容易修复算法里面的系统性错误。而数据黑客们通过坐在一起进行结对编程，逐行检查代码的输出并解释它的含义，会对找到这些错误有帮助。

敏捷的场所：开发的效率

一排排工位，我紧挨着你，如蜂巢般密集
超载的会议室里，各路人马，涌入又离去

Outlook 就是现代版的打卡机
置身格子间汪洋，更让人完全无法呼吸

最后限期哪挡得住闲聊的破坏力

单薄的隔断更是不堪一击

戴上耳机？不管用，效率还是低
绞尽脑汁想提高工作效率，可惜无路可依

这么吵，只能在家上班避一避

开放式办公？我去！

——作者赋诗一首

通才比专才更需要有一个能集中精神、不被打断的安静场所，他们的工作背景会更加广泛，因此他们要更专心。他们的场所必须满足这个条件。

将每个工位的空间放大到传统空间的两到三倍吧，否则你就是在浪费你雇佣的人。在这种配置下，有些人甚至不需要桌子，这样成本也会下降。

我们在这方面能做得更好，也应该做得更好。会多花一点钱，但并不昂贵。

在敏捷大数据里面，我们将团队成员看成是富有创意的人，而不是一般的办公室职员。因此可以将工作环境布置得更像一个工作室而不是办公室。同时也认识到对数据进行高级的数学运算并构建产品需要非常安静、非常集中的思考。所以也可以引入一些图书馆的元素。

很多企业仅将员工生产力的提高局限在对技能的培养上。但是 86% 的生产力问题存在于工作环境里面。工作环境对员工的效率会有影响。员工所在环境会决定这个企业能否繁荣昌盛。

——Akinyele Samuel Taiwo

雇佣人员比运维一栋大厦的成本要高得多，因此将钱花在改善工作环境上性价比是最高的。因为生产力提高 0.1% 到 2% 都能对整个公司的盈利能力带来激动人心的影响。

——Derek Clements-Croome and Li Baizhan

富有创造力的员工需要 3 类空间来协作和构建产品。从开放到封闭排序，它们分别是：
协作空间、个人空间、私人空间。

协作空间

协作空间是孵化创意的地方。协作空间通常位于主干道附近、各个部门之间，开放明亮，舒适又引人入胜。它没有墙，灵活又易于配置。它总是在变，总是在被重新布置，有很多的懒人沙发、枕头以及舒适的椅子。协作空间是一个能找到公司能量的地方：笑声、重要的讨论、兴奋的声音，此起彼伏。投资并展示这个区域吧。真的绿植（不是塑料的）能够隔音，而且还能造氧呢！

私人空间

私人空间是在期限前完成任务的地方。私人空间就像图书馆一样，封闭，隔音，没有人说话。私人空间能够最小化干扰（想象下一个灯光昏暗、只有细微声响的地方）。这里有懒人沙发，睡椅和椅子，还有符合人体工程学设计的工作站。这个空间还可以引入独立的坐式 / 立式书桌，缀满珠子的帘子后面有对接好的工作站，还有 30 英寸的订制 LCD。

个人空间

个人空间是人们称之为“家”的地方。个人空间的开放性介于协作空间和私人空间之间，它应该可以根据每个人的需要作个性化的布置（例如，共享的办公室、开放式的桌子、一个或者半个工位）。应该给员工一些选项和预算来让他们去布置自己的个人空间。也应该鼓励大家在布置时选一个主题，再种些植物。这会是某些人花时间最多的地方。但另一方面，对于某些人来说，只要有充足的协作空间和私人空间、一个笔记本电脑、一台移动设备，他们就可以不需要这个个人空间了。

最重要的是，敏捷工作环境的目标是通过物理环境，例如，打印件、海报、书、白板和其他东西，创建一种沉浸在数据里的感觉。如图 1-5 所示。

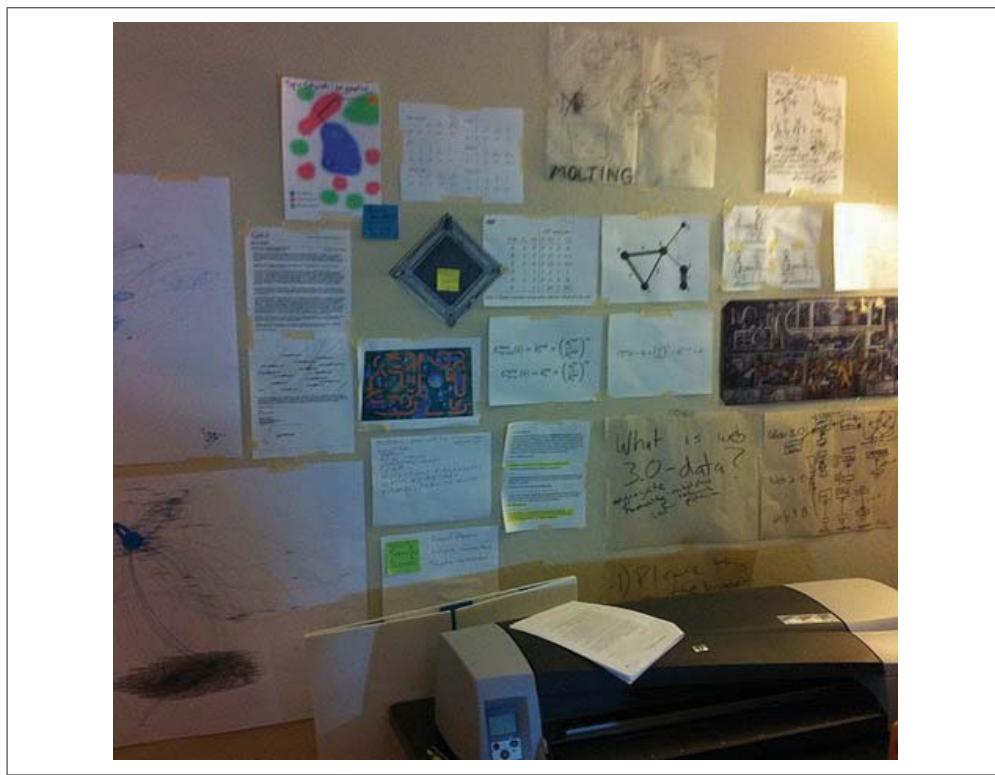


图1-5. 用拼贴画带来数据的浸入感

用大幅打印件明确表达想法

方便进行大尺寸的打印是敏捷环境里的一个需求。实际形式的可视化能鼓励大家分享，制作拼贴画，进行丰富有效的表达和产生创意。

惠普 DesignJet 111，一台 24 英寸宽的大尺寸打印机，仅需不到 1,000 美元。加上不到 100 美元的连续供墨系统后，只需不到 1 美元成本，就可以打印一张 24*36 英寸的海报。

在这种价位下，没有任何理由不给数据团队配置几台大尺寸打印机用于打印原稿稿件或者质地光滑的印刷品。当各个部门的成员看到数据科学团队的具体进度时，很容易就会兴奋起来。

第 2 章

数据

本章介绍本书余下部分所使用的数据集：你的电子邮件收件箱。本章也会介绍将要用到的工具和使用它们的理由。最后会概述多个分析数据的角度，供你进一步思考。

敏捷大数据流程会从数据开始，因此本书从数据开始介绍。



假如没有 Gmail 的账户，你需要（到 <http://mail.google.com>）创建一个，然后往邮箱里面发送一些邮件以完成本章后面的练习。

电子邮件

电子邮件是因特网的基本要素。甚至可以说，它是互联网的基石，组成了互联网和社交网络身份认证的基础。它因数量繁多为大家所熟知，除此之外，电子邮件复杂且内容丰富，能够挖掘出大量有趣的信息。

我们会用你的收件箱作为分析应用的数据集，这样例子就可以与你更相关。下载 Gmail 收件箱，并在例子中使用它，马上就会面临一个“大型”或者实际上是“中型”规模的数据问题——数据基本上无法在单机上有效处理^{译注1}。处理难以完全放入内存的数据需要可扩展的工具，这也促使我们去学习这些工具。使用你的收件箱，将能够对你的小小世界有更清楚的认识，并让你看到哪种技术更有效。这能培养我们对数据的感觉，同时也是敏捷大数据的一个重要主题。

在本书中，使用的工具与处理 PB 量级数据时用到的工具相同，只是在本地机器上运行。这个方法不仅能高效处理数据，而且我们所选择的工具确保只需要构建一次，就能得到

译注 1：作者在这里假定收件箱中内容很多，数据量相对较大。

可扩展性的应用。这样所有工作都会变得简洁，并让我们敏捷起来。

处理原始数据

原始的电子邮件

电子邮件的格式在 IETF RFC-5322 (Request For Comments by the Internet Engineering Taskforce) 有严谨的定义。要在 Gmail 中查看原始的邮件信息，选择一封邮件，然后在右上角的下拉菜单中选择“查看原始信息”选项。

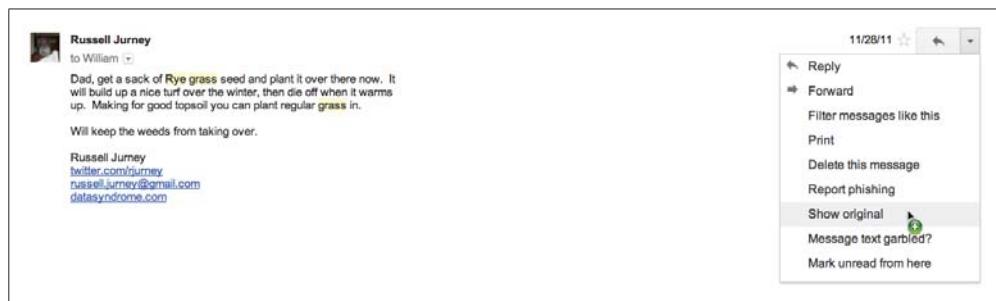


图 2-1. Gmail 的“查看原始信息”选项

一封原始格式的 email 看起来是这样的：

```
From: Russell Jurney <russell.jurney@gmail.com>
Mime-Version: 1.0 (1.0)
Date: Mon, 28 Nov 2011 14:57:38 -0800
Delivered-To: russell.jurney@gmail.com
Message-ID: <4484555894252760987@unknownmsgid>
Subject: Re: Lawn
To: William Jurney <*****@hotmail.com>
Content-Type: text/plain; charset=ISO-8859-1

Dad, get a sack of Rye grass seed and plant it over there now. It
will build up a nice turf over the winter, then die off when it warms
up. Making for good topsoil you can plant regular grass in.

Will keep the weeds from taking over.

Russell Jurney datasyndrome.com
```

这被称为半结构化数据。

结构化与半结构化数据

维基百科将半结构化数据定义为：

一种结构化数据形式，它并不符合正式的表格结构或关系数据模型，但具有标签或者其他标记，从中可分离出语义成分，并能从数据中形成记录或者字段的层次结构。

这与关系型、结构化的数据有区别。结构化的数据，意味着数据按照严谨明确的模式（schema）^{译注2}来组织，从而在后续工作中能更加有效地被查询。Andrew Fiore 和 Jeff Heer 展示了伯克利安然公司邮件数据集格式中邮件信息的结构化视图，如图 2-2。

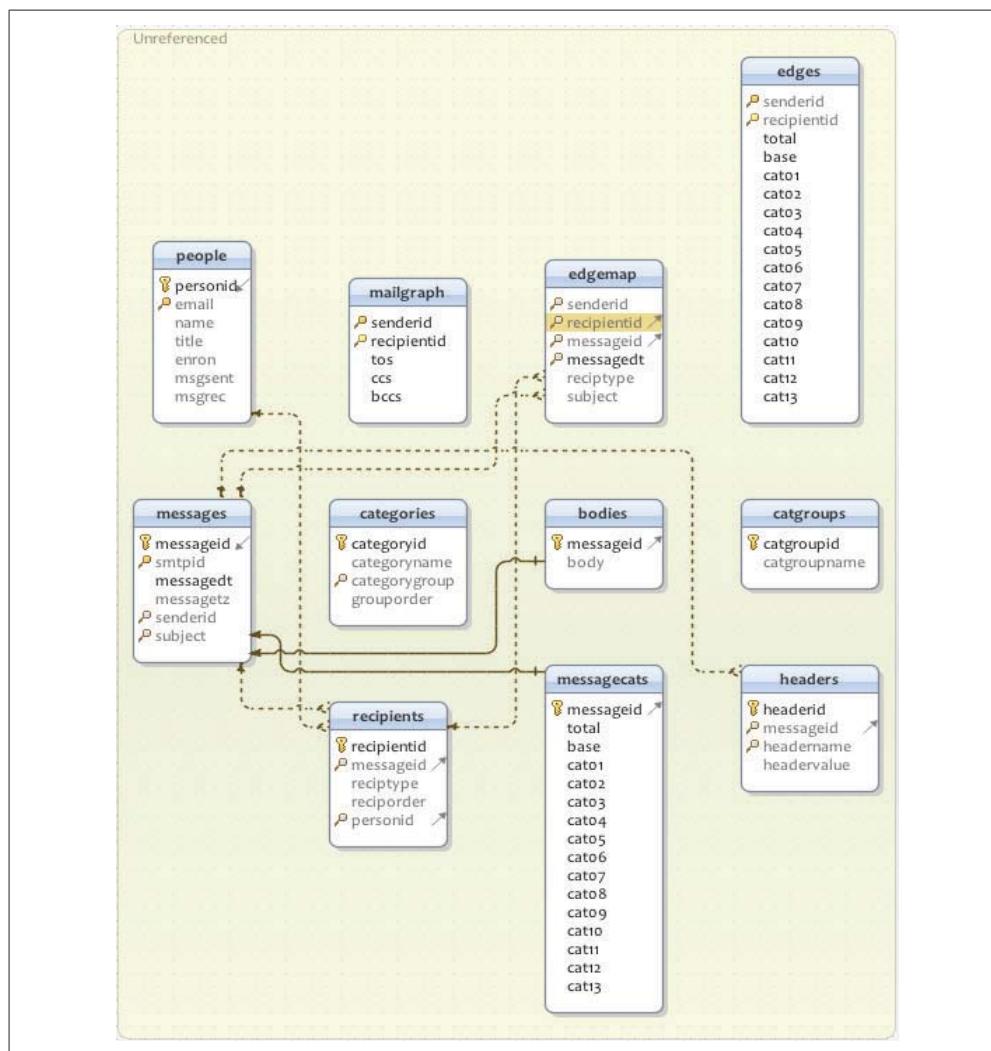


图2-2. 安然公司邮件格式

译注2：本书原文较多的使用 schema 一词，为保证连贯、易懂，在翻译过程中，译者在不同的上下文中将其翻译为模式、格式，请读者注意区分。

SQL

要查询关系型、结构化的数据模式，我们通常会用类似 SQL 这样的声明式编程语言。在 SQL 中，只需要描述想要什么，而不是去做什么。这和命令式编程有所不同。在 SQL 中，需要指定的是想要的输出，而非对数据的一系列操作。在安然关系型邮件数据库中，获取某封邮件全文的 SQL 查询如下所示：

```
select m.smtpid as id,
       m.messagedt as date,
       s.email as sender,
       (select GROUP_CONCAT(CONCAT(r.receiptype, ':', p.email) SEPARATOR ' ')
        from recipients r
        join people p
        on r.personid=p.personid
        where r.messageid = 511) as to_cc_bcc,
       m.subject as subject,
       SUBSTR(b.body, 1, 200) as body
      from messages m
     join people s
       on m.senderid=s.personid
    join bodies b
       on m.messageid=b.messageid
  where m.messageid=511;

| <25772535.1075839951307.JavaMail.evans@thyme> | 2002-02-02 12:56:33
| pete.davis@enron.com |
to:pete.davis@enron.com cc:albert.meyers@enron.com cc:bill.williams@enron.com
cc:craig.dean@enron.com
cc:geir.solberg@enron.com cc:john.anderson@enron.com cc:mark.guzman@enron.com
cc:michael.mier@enron.com
cc:pete.davis@enron.com cc:ryan.slinger@enron.com bcc:albert.meyers@enron.com
bcc:bill.williams@enron.com
bcc:craig.dean@enron.com bcc:geir.solberg@enron.com bcc:john.anderson@enron.com
bcc:mark.guzman@enron.com
bcc:michael.mier@enron.com bcc:pete.davis@enron.com bcc:ryan.slinger@enron.com
| Schedule Crawler:
HourAhead Failure | Start Date: 2/2/02; HourAhead hour: 11;
HourAhead schedule download failed. Manual intervention required. |
```

可以看到这个查询非常的复杂，需要对 3 个表进行连接（JOIN）操作，外加一个子查询，还使用了 MySQL 函数 GROUP_CONCAT, CONCAT 和 SUBSTR。关系型数据不鼓励根据数据本身的形式去展示数据，而是要求我们根据关系的模式去思考。这种复杂性将影响整个数据分析过程，让我们面对一个充满 SQL 而非真实数据的世界。

还要注意，定义上述数据表也相当复杂：

```
CREATE TABLE bodies (
  messageid int(10) unsigned NOT NULL default '0',
  body text,
```

```
    PRIMARY KEY  (messageid)
) TYPE=MyISAM;

CREATE TABLE categories (
    categoryid int(10) unsigned NOT NULL auto_increment,
    categoryname varchar(255) default NULL,
    categorygroup int(10) unsigned default NULL,
    grouporder int(10) unsigned default NULL,
    PRIMARY KEY  (categoryid),
    KEY categories_categorygroup (categorygroup)
) TYPE=MyISAM;

CREATE TABLE catgroups (
    catgroupid int(10) unsigned NOT NULL default '0',
    catgroupname varchar(255) default NULL,
    PRIMARY KEY  (catgroupid)
) TYPE=MyISAM;

CREATE TABLE edgemap (
    senderid int(10) unsigned default NULL,
    recipientid int(10) unsigned default NULL,
    messageid int(10) unsigned default NULL,
    messagedt timestamp(14) NOT NULL,
    reciptype enum('bcc','cc','to') default NULL,
    subject varchar(255) default NULL,
    KEY senderid (senderid,recipientid),
    KEY messageid (messageid),
    KEY messagedt (messagedt),
    KEY senderid_2 (senderid),
    KEY recipientid (recipientid)
) TYPE=MyISAM;

CREATE TABLE edges (
    senderid int(10) unsigned default NULL,
    recipientid int(10) unsigned default NULL,
    total int(10) unsigned NOT NULL default '0',
    base int(10) unsigned NOT NULL default '0',
    cat01 int(10) unsigned NOT NULL default '0',
    cat02 int(10) unsigned NOT NULL default '0',
    cat03 int(10) unsigned NOT NULL default '0',
    cat04 int(10) unsigned NOT NULL default '0',
    cat05 int(10) unsigned NOT NULL default '0',
    cat06 int(10) unsigned NOT NULL default '0',
    cat07 int(10) unsigned NOT NULL default '0',
    cat08 int(10) unsigned NOT NULL default '0',
    cat09 int(10) unsigned NOT NULL default '0',
    cat10 int(10) unsigned NOT NULL default '0',
    cat11 int(10) unsigned NOT NULL default '0',
    cat12 int(10) unsigned NOT NULL default '0',
    cat13 int(10) unsigned NOT NULL default '0',
```

```
        UNIQUE KEY senderid (senderid,recipientid)
    ) TYPE=MyISAM;

CREATE TABLE headers (
    headerid int(10) unsigned NOT NULL auto_increment,
    messageid int(10) unsigned default NULL,
    headername varchar(255) default NULL,
    headervalue text,
    PRIMARY KEY (headerid),
    KEY headers_headername (headername),
    KEY headers_messageid (messageid)
) TYPE=MyISAM;

CREATE TABLE messages (
    messageid int(10) unsigned NOT NULL auto_increment,
    smtplib varchar(255) default NULL,
    messagedt timestamp(14) NOT NULL,
    messagezt varchar(20) default NULL,
    senderid int(10) unsigned default NULL,
    subject varchar(255) default NULL,
    PRIMARY KEY (messageid),
    UNIQUE KEY smtplib (smtplib),
    KEY messages_senderid (senderid),
    KEY messages_subject (subject)
) TYPE=MyISAM;

CREATE TABLE people (
    personid int(10) unsigned NOT NULL auto_increment,
    email varchar(255) default NULL,
    name varchar(255) default NULL,
    title varchar(255) default NULL,
    enron tinyint(3) unsigned default NULL,
    msgsent int(10) unsigned default NULL,
    msgrec int(10) unsigned default NULL,
    PRIMARY KEY (personid),
    UNIQUE KEY email (email)
) TYPE=MyISAM;
-- 
--

CREATE TABLE recipients (
    recipientid int(10) unsigned NOT NULL auto_increment,
    messageid int(10) unsigned default NULL,
    recipctype enum('bcc','cc','to') default NULL,
    reciporder int(10) unsigned default NULL,
    personid int(10) unsigned default NULL,
    PRIMARY KEY (recipientid),
    KEY messageid (messageid)
) TYPE=MyISAM;
```

相比之下，在敏捷大数据中，我们使用数据流语言在代码中定义数据的格式，甚至无需指定数据模式就可以将数据直接发布到文档仓库中。这为流程进行了优化：从事数据科

学就是要从已有的数据中挖掘出新的信息。由此来看，从数据外部指定数据模式非但没有好处，反而是额外的负担。归根结底，我们无法预知最终将会得到什么信息，因为数据科学总是充满惊喜的。

当然，关系型的结构化数据也有好处。例如，只需要一个简单的 `select/group by/order` 查询就可以看到用户在哪个时间发邮件。

```
select senderid as id,
       hour(messageadt) as sent_hour,
       count(*)
  from messages
 where senderid=511
 group by
       senderid,
       m_hour
 order by
       senderid,
       m_hour;
```

查询会返回如下结果：

senderid	m_hour	count(*)
1	0	4
1	1	3
1	3	2
1	5	1
1	8	3
1	9	1
1	10	5
1	11	2
1	12	2
1	14	1
1	15	5
1	16	4
1	17	1
1	19	1
1	20	1
1	21	1
1	22	1
1	23	1

关系型数据库会根据数据的结构将数据分拆到不同的表中，并为表之间的操作预建索引。索引可以让系统在单机下拥有更好的响应速度。声明式编程则用于查询这些结构。

声明式编程非常适宜于处理和查询结构化数据，进行聚合，然后生成简单的图表。如果明确知道我们想要的，就可以直截了当的告诉 SQL 引擎，它会替我们算好。并不需要考虑查询执行的细节。

NoSQL

与 SQL 相反，创建分析应用时，我们并不知道需要查询什么。对于任意问题，都可能需要多次实验和迭代才能得到解决。数据经常难以以关系型格式存在。原始的数据没有标准化，含混不清且充满噪音。为了不同的产品特性处理数据，从数据中抽取结构，是一个长期的迭代过程。

由于这些原因，在敏捷大数据里，我们主要使用命令式编程语言在分布式系统中工作。类似 Pig Latin 的命令式语言指明了数据流水线（pipeline）中每一步的数据操作。用多个处理器核心，并行地读入独立的数据，而不是在未知结构的数据上预先计算索引。Hadoop 及其工作队列让这种做法变得可能。

除了能更好的和 Hadoop 这样具有扩展性的分布式技术对接，命令式语言让我们能够将注意力放在创建分析应用的若干关键步骤上去，做一些巧妙的事情，最大化分析应用的价值。

由于引入了统计学、机器学习、社交网络分析等工具，因此与命令式编程相比起来，编写 SQL 来完成这些复杂操作，耗时长久且容易让人精疲力尽。还是命令式编程语言适合完成这类任务。

总结一下，当数据模式定义严谨，且 SQL 是唯一的工具时，我们就不会从挖掘数据的角度看待数据，因为视角会被优化过的数据处理工具所支配。严谨的数据格式抑制了我们在数据与直觉之间建立起联系的能力。另一方面，半结构化的数据可以让我们专注于数据本身，通过迭代地操作数据来抽取价值，并转化为产品。在敏捷大数据中选择 NoSQL，是因为它让我们具备了上述能力。

序列化

虽然可以用纯文本方式处理半结构化数据，但在原始记录中添加某些模式仍有帮助。序列化系统给予了这样的能力。有下面这些序列化系统可供选择：

Thrift: <http://thrift.apache.org/>
Protobuf: <http://code.google.com/p/protobuf/>
Avro: <http://avro.apache.org/>

尽管在以上选择中，Avro 是最不成熟的一个，但我们仍然会使用它。Avro 可定义复杂的数据结构，在每个文件中包含模式，且支持 Apache Pig。安装 Avro 很简单，不依赖任何外部服务就能运行。

我们会为电子邮件按照 RFC-5322 标准定义一个简单的 Avro 模式。预定义模式是个不错的做法，但实际上，抽取出模式中所有的实体需要相当多的工作量。因此可以从简单做起，如下所示：

```
{  
    "type": "record",  
    "name": "RawEmail",  
    "fields":  
    [  
        {  
            "name": "thread_id",  
            "type": ["string", "null"],  
            "doc": ""  
        },  
        {  
            "name": "raw_email",  
            "type": ["string", "null"]  
        }  
    ]  
}
```

我们抽取 `thread_id` 作为邮件唯一的标识符，并将整封原始邮件放在 `raw_email` 字段上。假如难以从原始数据中抽取出邮件的唯一标识符，可以生成一个 UUID (universally unique identifier, 全局唯一标识符) 并将其添加为一个字段。

在处理数据时，我们的任务就是提取出字段并添加到模式中，并尽可能保留数据的原始面貌。这样就总能有一份原始数据可供参考。

从演变的模式中抽取和展示特征

Pete Warden 在他的演讲“拥抱数据混沌”中提到，(<http://bit.ly/17u/27>) 绝大部分能免费获取的数据都是粗糙和非结构化的。正是从这些大规模的脏数据、没有被仔细清理并标准化的表格里，诞生了“大数据”。这里存在着机会：从粗糙数据中挖掘出有价值的信息，并驱动各种新的行动。

从非结构化数据中抽取特征，是一个非常苛刻的任务；用户会不断的使用、抱怨这些特征。抽取出的特征不能被应用起来是一件非常糟糕的事情。构建数据产品最难的地方在于将实体和特征抽取与一个比最终愿景要小的产品绑在一起。这也是数据模式必须从 blob 状态^{译注3} 的非结构化文本开始、仅当特征被抽取出来时才转为结构化的原因。

特征一旦被创建，必须以某种产品形式展示，否则它们永远不会处于准产品状态。没有开放的衍生数据不太可能会成型。最好从一开始就能够创建实体页面，将实体放到一个

译注 3：blob, binary large objects 的缩写。

用户层面的表单中，逐步地改进这些实体，渐进地组合它们，以在一个更广的视角下展示更多样的衍生数据。

在对数据进行挖掘并转换成结构化信息的过程中，用这些信息去展示新的事实，做出能指导行动的预测，会产生巨大的潜在价值。数据冷酷无情，也不会原谅我们的错误。不能意识到它真正的含义，会打碎绝大部分怀着雄心壮志的产品经理的梦想。

在书中我们将看到，数据模式会不断演化和改进，用于展示数据的特征也一样。当它们能够同时演化时，就真的实现敏捷了。

数据流水线

我们会在数据流水线里使用半结构化数据去抽取并展示数据的不同特征。这样做的好处是，不需要浪费时间去抽取数据结构，除非我们对结构感兴趣并觉得它有用。所以，根据 KISS (Keep it simple, stupid！) 原则和 YAGNI (You Ain't Gonna Need It) 原则，如非必要，就先放到一边。在第三章将看到，我们的工具集能让流水线更高效。

图 2-3 展示了一条计算两个地址间发送邮件次数的数据流水线。

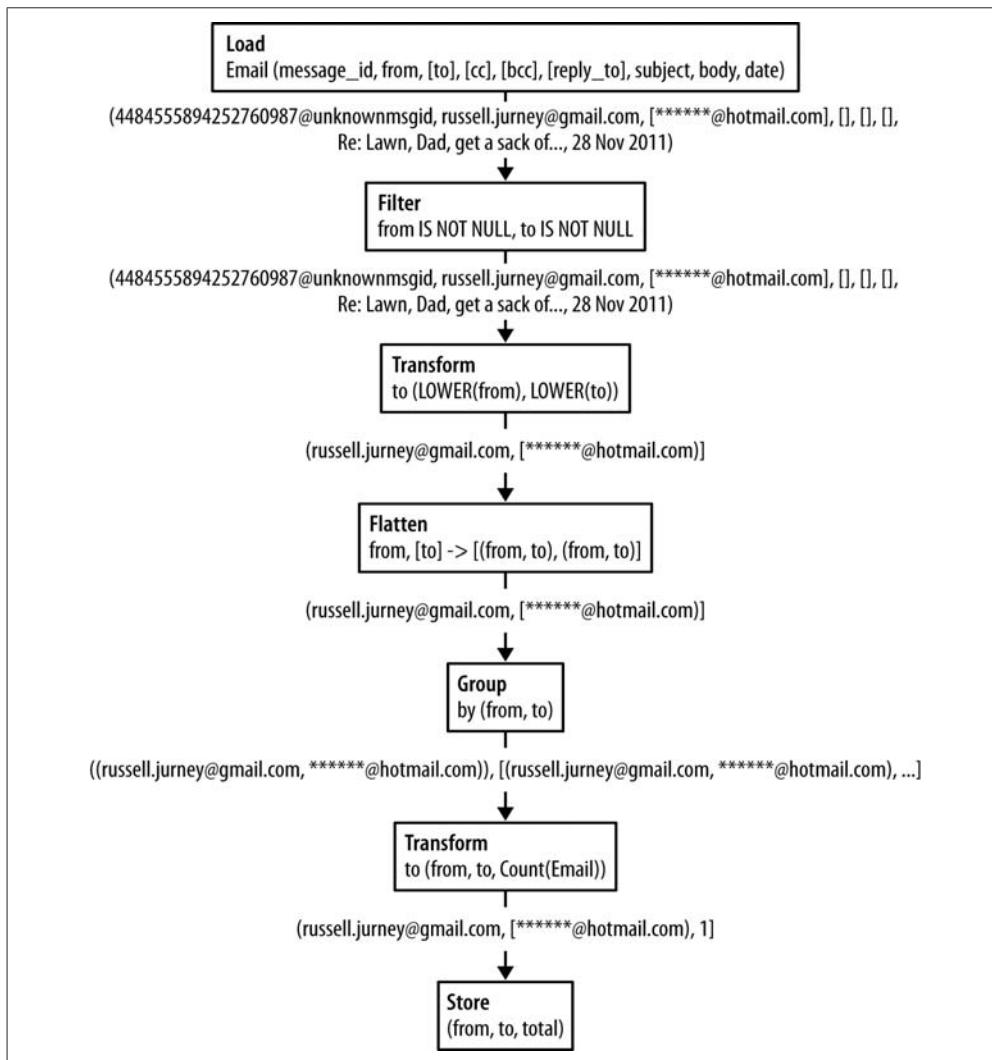


图2-3. 计算两个地址间发送邮件次数的数据流

假如已经习惯了 SQL，这条数据流看起来可能很复杂，但很快你就会习惯这种工作方式，而且这种简单的流程也会变成一种习惯。

数据透视

介绍这部分之前，着重介绍考察邮件数据的不同方法会有所帮助。在敏捷大数据中，我们会使用多种不同的角度和方法去考察并挖掘数据，因为只用某一两种你觉得有效

的方法，很容易遇到困难。接下来，会讨论几种考察邮件数据的不同视角，它们会贯穿全书。

社交网络

社交网络指一组人员以及他们之间的联系和链接。这些联系可能是有方向的，例如“鲍勃知道萨拉是谁”。也可能是无向的，例如“鲍勃和萨拉是朋友”。联系也可以有强度或权重，例如“鲍勃很了解萨拉”（分值从 0 到 1），或者“鲍勃和萨拉是夫妻”（分值是 0 或者 1）。

从邮件的发件人、收件人、抄送、密送字段能提取出邮件的发送者和接收者，这些信息就可以用于创建一个社交网络。例如，下面这封邮件定义了两个实体，`russell.jurney@gmail.com` 和 `*****@hotmail.com`。

```
From: Russell Jurney <russell.jurney@gmail.com>
To: ***** Jurney <*****@hotmail.com>
```

这封邮件本身就暗示了他们之间的关系。我们将它表示成一个简单的社交网络，如图 2-4 所示。

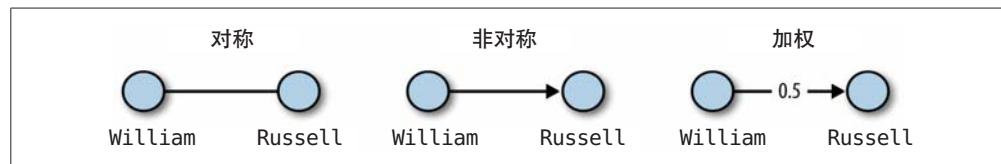


图2-4. 两个结点的社交网络

图 2-5 描绘了一个复杂一些的社交网络。

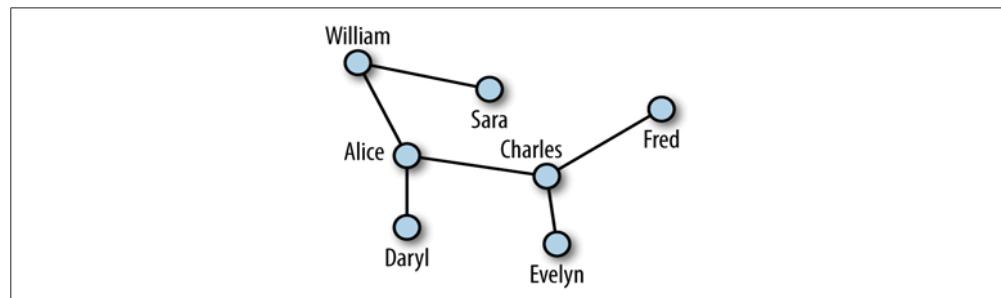


图2-5. 社交网络

图 2-6 展示了由安然公司 200MB 的邮件数据构成的社交网络。

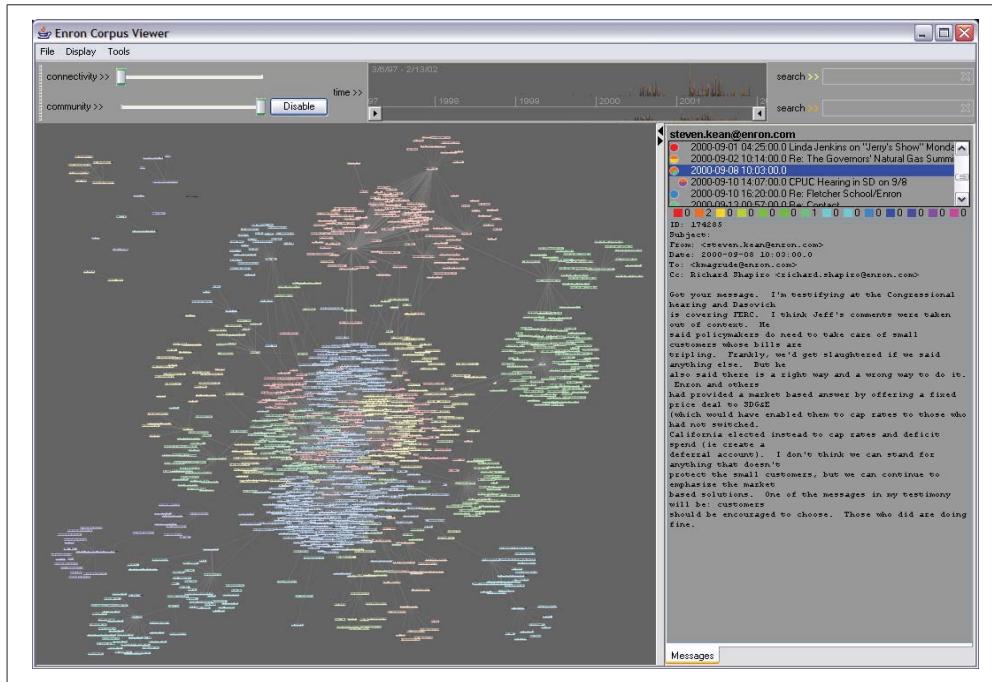


图2-6. 安然公司语料库查看器，由Jeffrey Heer 和 Andrew Fiore 创建

社交网络分析（Social network analysis, 简称 SNA），是针对社交网络的科学研究与分析。通过对我们的电子邮件收件箱进行社交网络建模，就可以利用社交网络分析方法（比如 PageRank），对数据、我们的个人网络得到更深入的了解和认识。图 2-7 展示了应用于安然公司社交网络数据的一种分析。

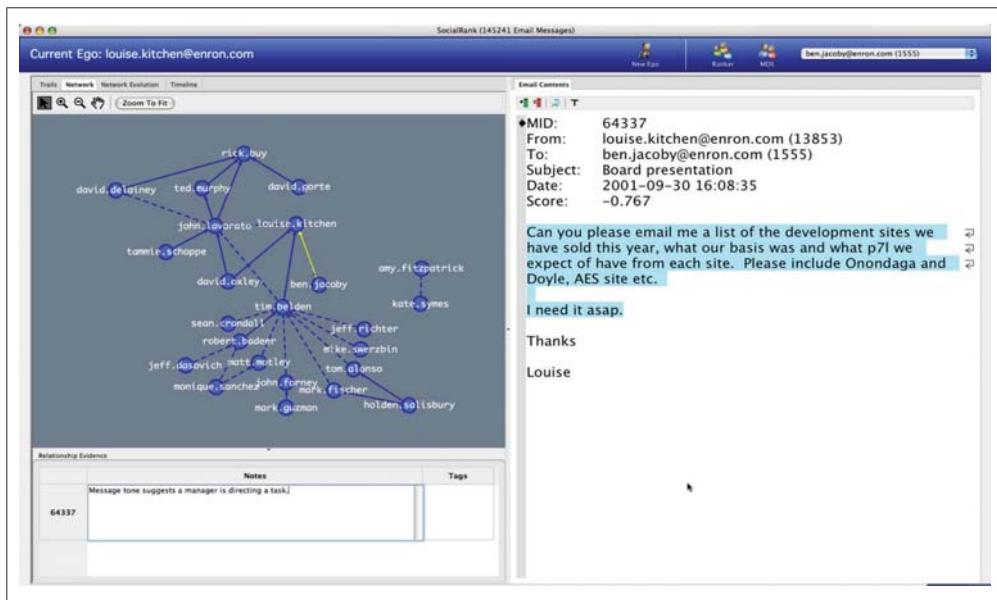


图2-7. 安然公司 SocialRank, 由Jaime Montemayor, Chris Diehl, Mike Pekala, 和David Patrone创建

时间序列

时间序列是根据时间戳进行排序的一系列数据点。时间序列能够让我们看到随着时间变化，数据的变化和趋势。所有的邮件都有时间戳，所以可以将一系列的邮件表示成时间序列，如图 2-8 所示。

Date: Mon, 28 Nov 2011 14:57:38 -0800

再查看几封其他邮件，就可以将数据以时间序列形式绘制出来。

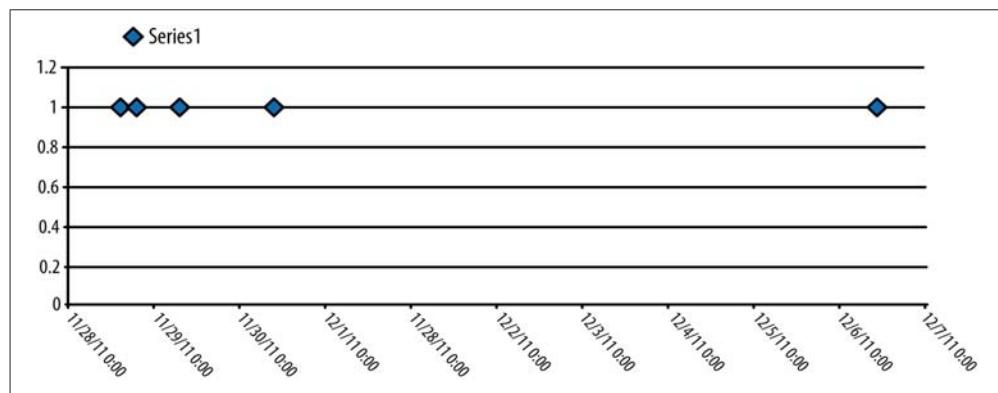


图2-8. 原始的时间序列图

既然并不关注时间序列上的某个特定数值，那可以将数据以天为单位聚合起来，以更清晰地展示数据（见图 2-9）。这会告诉我们这两个地址之间每天发送了多少封邮件。

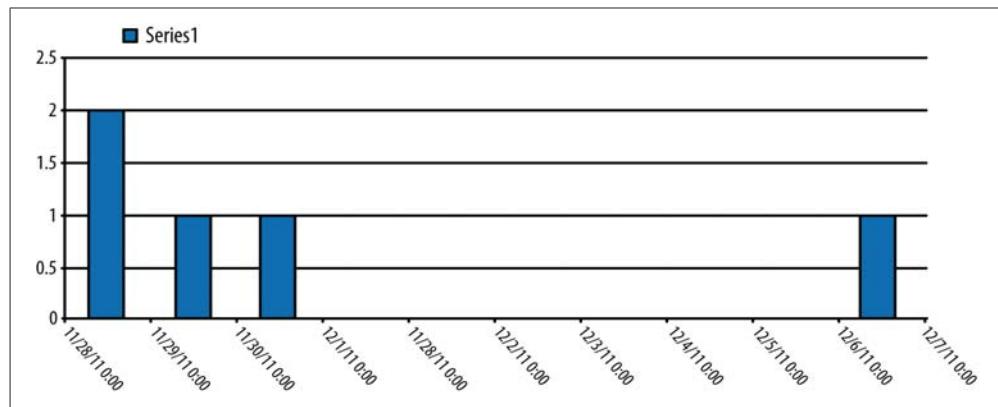


图2-9. 按天聚合后的时间序列

基于时间序列的分析可以告诉我们主要在什么时候收到某人的来信，甚至发件人的工作日程表。

自然语言

邮件的精华部分在于文本内容。尽管邮件中能附带 MIME 多媒体附件，但它主要还是由文本组成。

```
Subject: Re: Lawn
Content-Type: text/plain; charset=ISO-8859-1
```

Dad, get a sack of Rye grass seed and plant it over there now. It will build up a nice turf over the winter, then die off when it warms up. Making for good topsoil you can plant regular grass in.

Will keep the weeds from taking over.

Russell Jurney
twitter.com/rjurney
russell.jurney@gmail.com
datasyndrome.com

可以通过计算正文的词频来分析这封邮件。移除停用词（如 *of* 和 *it*）以后，词频如图 2-10 所示。

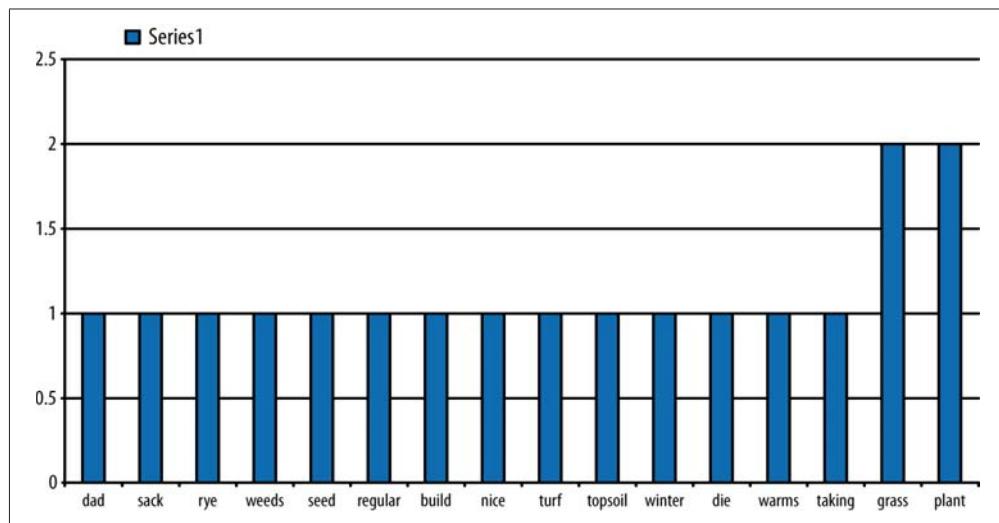


图2-10. 邮件正文词频

可以用这个词频数据推测这封邮件的主题是 “*plant*” 和 “*grass*”，因为它们出现次数最多。用这个方法处理自然语言能够帮助我们从半结构化数据中提取数据的各种属性，使数据更加结构化。这也让我们可以用这些结构化后的属性进行分析。

Wordle 是一种展示词频的有趣方法，如图 2-11。



图2-11. 邮件正文的Wordle

概率

在概率论中，可以统计数据中不同属性的出现次数、共现次数，创建概率分布，以对看似随机的过程建模。然后根据这些概率分布可以得出建议，也可以将实体分到不同的类别中。

还可以用这些概率分布进行预测。例如可以为邮件的发件人、收件人、抄送字段建立概率分布。已知我的邮件地址 `russell.jurney@gmail.com` 作为发件人，某一个邮件地址作为收件人，另一个邮件地址出现在抄送字段中的概率是多少呢？

在这例子里，原始数据是每封邮件的收件人，发件人和抄送字段。

```
From: Russell Jurney <russell.jurney@gmail.com>
To: ***** Jurney <*****@hotmail.com>
Cc: Ruth Jurney <****@hotmail.com>
```

首先，对所有的发件人和收件人组合进行计数。这也被称为两个属性的共现数。让我们重点关注特定的组合吧，看看 O'Reilly 编辑 Mike Loukides 和我之间的邮件。（表 2-1）

表2-1. 发件人/收件人组合的总发信数

From	To	Count
russell.jurney@gmail.com	*****.jurney@gmail.com	10
russell.jurney@gmail.com	toolsreq@oreilly.com	10
russell.jurney@gmail.com	yoga*****@gmail.com	11
russell.jurney@gmail.com	user@pig.apache.org	14
russell.jurney@gmail.com	*****@hotmail.com	15
russell.jurney@gmail.com	mikel@oreilly.com	28
russell.jurney@gmail.com	russell.jurney@gmail.com	44

将上面的数值除以邮件的总数，就能得出给定发件人，某个邮件地址会成为“收件人”的概率分布（表 2-2）。

 表2-2. $P(to|from)$: 给定发件人时收件人的概率

From	To	Probability
russell.jurney@gmail.com	*****.jurney@gmail.com	0.0359
russell.jurney@gmail.com	toolsreq@oreilly.com	0.0359
russell.jurney@gmail.com	yoga*****@gmail.com	0.0395
russell.jurney@gmail.com	user@pig.apache.org	0.0503
russell.jurney@gmail.com	*****@hotmail.com	0.0539
russell.jurney@gmail.com	mikel@oreilly.com	0.1007
russell.jurney@gmail.com	russell.jurney@gmail.com	0.1582

最后可以列出给定一个收件人，两个邮件接收者共现的概率。（表 2-3）

 表2-3. $P(cc|from \cap to)$: 给定发件人和收件人，抄送另外一个人的概率

From	To	Cc	Probability
russell.jurney@gmail.com	mikel@oreilly.com	toolsreq@oreilly.com	0.0357
russell.jurney@gmail.com	mikel@oreilly.com	meghan@oreilly.com	0.25
russell.jurney@gmail.com	mikel@oreilly.com	mstallone@oreilly.com	0.25
russell.jurney@gmail.com	toolsreq@oreilly.com	meghan@oreilly.com	0.1
russell.jurney@gmail.com	toolsreq@oreilly.com	mikel@oreilly.com	0.2

这样，就可以用这份数据去展示，给定一个收件人，谁更有可能同时出现在一封邮件中。这份数据可以用于驱动类似 Gmail 中的“建议接收者”功能，如图 2-12 所示。



图2-12. Gmail的建议接收者特性

在后面将看到，在表 2-3 数据不完整的情况下，如何利用贝叶斯推断来对邮件接收者给出合理的建议。

小结

就像所看到的那样，根据不同的算法、结构或视角去考察半结构化数据，比之将数据标准化后展示在结构化的表格里，需要开发更多的功能。本章中定义的这些视角会贯穿全书，在攀登数据 – 价值金字塔^{译注4} 的过程中，我们会用这些视角来开发各种功能特性。在下一章，你会学到如何直接使用 Apache Pig 指定存储系统中的数据模式。

译注 4： 作者在书中多次将敏捷数据开发过程比喻成攀登金字塔，塔底是数据本身，塔的顶部则是从数据中挖掘出的价值。