CHAPTER FIVE

Information Platforms and the Rise of the Data Scientist

Jeff Hammerbacher

Libraries and Brains

At the age of 17, I was fired from my job as a cashier at Scott's grocery store in Fort Wayne, Indiana. With only two months remaining before my freshman year of college, I saw in my unemployment an opportunity. Instead of telling my parents that I had been fired, I continued to leave the house every afternoon in my cashier's outfit: black pants, black shoes, white shirt, and smock. To my parents, I looked ready for some serious coupon scanning; in reality, I was pulling 10-hour shifts reading at the public library.

All reasonably curious people wonder how their brain works. At 17, I was unreasonably curious. I used my time at the library to learn about how brains work, how they break, and how they are rebuilt. In addition to keeping us balanced, regulating our body temperature, and making sure we blink our eyelids together every now and again, our brains ingest, process, and generate massive amounts of information. We construct unconscious responses to our immediate environment, short-term plans for locution and limb placement, and long-term plans for mate selection and education. What makes brains interesting is not just their ability to generate reactions to sensory data, but their role as repository of information for both plan generation and the creation of new information. I wanted to learn how that worked.

One thing about brains, though: they remain stubbornly housed within a single body. To collect information from many brains, we build libraries. The field of library science has evolved numerous techniques for herding the information stored in libraries to enable future consumption; a fun read on the topic is Alex Wright's *Glut* (Joseph Henry Press). In addition to housing information for future retrieval, libraries play a critical role in the creation of new information. As philosopher Daniel Dennett puts it, "a scholar is just a library's way of making another library."

Libraries and brains are two examples of Information Platforms. They are the locus of their organization's efforts to ingest, process, and generate information, and they serve to accelerate the process of learning from empirical data. When I joined Facebook in 2006, I naturally started to build an Information Platform. Because of the tremendous growth in the number of users on Facebook, the system our team built ended up managing several petabytes of data. In this chapter, I'll recount the challenges faced in building out Facebook's Information Platform and the lessons learned while constructing our solution from open source software. I'll also try to outline the critical role of the Data Scientist in using that information to build data-intensive products and services and helping the organization formulate and accomplish goals. Along the way, I'll recount how some other businesses have approached the problem of building Information Platforms over the decades.

Before we get started, I should point out that my clever plan to visit the library instead of the grocery store did not work out as intended. After a few blissful days of reading, I came out of the library one evening and couldn't locate my car. It was not uncommon for me to lose my car at the time, but the lot was empty, so I knew something was up. It turns out that my mom had figured out my scheme and gotten my car towed. During the long walk home, I internalized an important lesson: regard your own solutions with skepticism. Also, don't try to outsmart your mother.

Facebook Becomes Self-Aware

In September 2005, Facebook opened to non-college students for the first time and allowed high school students to register for accounts. Loyal users were outraged, but the Facebook team felt that it was the right direction for the site. How could it produce evidence to justify its position?

In addition, Facebook had saturated the student population at nearly all of the colleges where it was available, but there were still some colleges where the product had never taken off. What distinguished these laggard networks from their more successful peers, and what could be done to stimulate their success?

When I interviewed at Facebook in February 2006, they were actively looking to answer these questions. I studied mathematics in college and had been working for a nearly a year on Wall Street, building models to forecast interest rates, price complex derivatives, and hedge pools of mortgages; I had some experience coding and a dismal GPA. Despite my potentially suboptimal background, Facebook made me an offer to join as a Research Scientist.

Around the same time, Facebook hired a Director of Reporting and Analytics. The director had far more experience in the problem domain than me; together with a third engineer, we set about building an infrastructure for data collection and storage that would allow us to answer these questions about our product.

Our first attempt at an offline repository of information involved a Python script for farming queries out to Facebook's tier of MySQL servers and a daemon process, written in C++, for processing our event logs in real time. When the scripts worked as planned, we collected about 10 gigabytes a day. I later learned that this aspect of our system is commonly termed the "ETL" process, for "Extract, Transform, and Load."

Once our Python scripts and C++ daemon had siphoned the data from Facebook's source systems, we stuffed the data into a MySQL database for offline querying. We also had some scripts and queries that ran over the data once it landed in MySQL to aggregate it into more useful representations. It turns out that this offline database for decision support is better known as a "Data Warehouse."

Finally, we had a simple PHP script to pull data from the offline MySQL database and display summaries of the information we had collected to internal users. For the first time, we were able to answer some important questions about the impact of certain site features on user activity. Early analyses looked at maximizing growth through several channels: the layout of the default page for logged-out users, the source of invitations, and the design of the email contact importer. In addition to analyses, we started to build simple products using historical data, including an internal project to aggregate features of sponsored group members that proved popular with brand advertisers.

I didn't realize it at the time, but with our ETL framework, Data Warehouse, and internal dashboard, we had built a simple "Business Intelligence" system.

A Business Intelligence System

In a 1958 paper in the *IBM Systems Journal*, Hans Peter Luhn describes a system for "selective dissemination" of documents to "action points" based on the "interest profiles" of the individual action points. The author demonstrates shocking prescience. The title of the paper is "A Business Intelligence System," and it appears to be the first use of the term "Business Intelligence" in its modern context.

In addition to the dissemination of information in real time, the system was to allow for "information retrieval"—search—to be conducted over the entire document collection. Luhn's emphasis on action points focuses the role of information processing on goal completion. In other words, it's not enough to just collect and aggregate data; an organization must improve its capacity to complete critical tasks because of the insights gleaned from the data. He also proposes "reporters" to periodically sift the data and selectively move information to action points as needed.

The field of Business Intelligence has evolved over the five decades since Luhn's paper was published, and the term has come to be more closely associated with the management of structured data. Today, a typical business intelligence system consists of an ETL framework pulling data on a regular basis from an array of data sources into a Data Warehouse, on top of which sits a Business Intelligence tool used by business analysts to generate reports for internal consumption. How did we go from Luhn's vision to the current state of affairs?

E. F. Codd first proposed the relational model for data in 1970, and IBM had a working prototype of a relational database management system (RDBMS) by the mid-1970s. Building user-facing applications was greatly facilitated by the RDBMS, and by the early 1980s, their use was proliferating.

In 1983, Teradata sold the first relational database designed specifically for decision support to Wells Fargo. A few years later, in 1986, Ralph Kimball founded Red Brick Systems to build databases for the same market. Solutions were developed using Teradata and Red Brick's offerings, but it was not until 1991 that the first canonical text on data warehousing was published.

Bill Inmon's *Building the Data Warehouse* (Wiley) is a coherent treatise on data warehouse design and includes detailed recipes and best practices for building data warehouses. Inmon advocates constructing an enterprise data model after careful study of existing data sources and business goals.

In 1995, as Inmon's book grew in popularity and data warehouses proliferated inside enterprise data centers, The Data Warehouse Institute (TDWI) was formed. TDWI holds conferences and seminars and remains a critical force in articulating and spreading knowledge about data warehousing. That same year, data warehousing gained currency in academic circles when Stanford University launched its WHIPS research initiative.

A challenge to the Inmon orthodoxy came in 1996 when Ralph Kimball published *The Data Warehouse Toolkit* (Wiley). Kimball advocated a different route to data warehouse nirvana, beginning by throwing out the enterprise data model. Instead, Kimball argued that different business units should build their own data "marts," which could then be connected with a "bus." Further, instead of using a normalized data model, Kimball advocated the use of dimensional modeling, in which the relational data model was manhandled a bit to fit the particular workload seen by many data warehouse implementations.

As data warehouses grow over time, it is often the case that business analysts would like to manipulate a small subset of data quickly. Often this subset of data is parameterized by a few "dimensions." Building on these observations, the CUBE operator was introduced in 1997 by a group of Microsoft researchers, including Jim Gray. The new operator enabled fast querying of small, multidimensional data sets.

Both dimensional modeling and the CUBE operator were indications that, despite its success for building user-facing applications, the relational model might not be best for constructing an Information Platform. Further, the document and the action point, not the

table, were at the core of Luhn's proposal for a business intelligence system. On the other hand, an entire generation of engineers had significant expertise in building systems for relational data processing.

With a bit of history at our back, let's return to the challenges at Facebook.

The Death and Rebirth of a Data Warehouse

At Facebook, we were constantly loading more data into, and running more queries over, our MySQL data warehouse. Having only run queries over the databases that served the live site, we were all surprised at how long a query could run in our data warehouse. After some discussion with seasoned data warehousing veterans, I realized that it was normal to have queries running for hours and sometimes days, due to query complexity, massive data volumes, or both.

One day, as our database was nearing a terabyte in size, the `mysqld` daemon process came to a sudden halt. After some time spent on diagnostics, we tried to restart the database. Upon initiating the restart operation, we went home for the day.

When I returned to work the next morning, the database was still recovering. To get a consistent view of data that's being modified by many clients, a database server maintains a persistent list of all edits called the "redo log" or the "write-ahead log." If the database server is unceremoniously killed and restarted, it will reread the recent edits from the redo log to get back up to speed. Given the size of our data warehouse, the MySQL database had quite a bit of recovery to catch up on. It was three days before we had a working data warehouse again.

We made the decision at that point to move our data warehouse to Oracle, whose database software had better support for managing large data sets. We also purchased some expensive high-density storage and a powerful Sun server to run the new data warehouse.

During the transfer of our processes from MySQL to Oracle, I came to appreciate the differences between supposedly standard relational database implementations. The bulk import and export facilities of each database used completely different mechanisms. Further, the dialect of SQL supported by each was different enough to force us to rewrite many of our queries. Even worse, the Python client library for Oracle was unofficial and a bit buggy, so we had to contact the developer directly.

After a few weeks of elbow grease, we had the scripts rewritten to work on the new Oracle platform. Our nightly processes were running without problems, and we were excited to try out some of the tools from the Oracle ecosystem. In particular, Oracle had an ETL tool called Oracle Warehouse Builder (OWB) that we hoped could replace our handwritten Python scripts. Unfortunately, the software did not expect the sheer number of data sources we had to support: at the time, Facebook had tens of thousands of MySQL databases from which we collected data each night. Not even Oracle could help us tackle our scaling challenges on the ETL side, but we were happy to have a running data warehouse with a few terabytes of data.

And then we turned on clickstream logging: our first full day sent 400 gigabytes of unstructured data rushing over the bow of our Oracle database. Once again, we cast a skeptical eye on our data warehouse.

Beyond the Data Warehouse

According to IDC, the digital universe will expand to 1,800 exabytes by 2011. The vast majority of that data will not be managed by relational databases. There's an urgent need for data management systems that can extract information from unstructured data in concert with structured data, but there is little consensus on the way forward.

Natural language data in particular is abundant, rich with information, and poorly managed by a data warehouse. To manage natural language and other unstructured data, often captured in document repositories and voice recordings, organizations have looked beyond the offerings of data warehouse vendors to various new fields, including one known as enterprise search.

While most search companies built tools for navigating the collection of hyperlinked documents known as the World Wide Web, a few enterprise search companies chose to focus on managing internal document collections. Autonomy Corporation, founded in 1996 by Cambridge University researchers, leveraged Bayesian inference algorithms to facilitate the location of important documents. Fast Search and Transfer (FAST) was founded in 1997 in Norway with more straightforward keyword search and ranking at the heart of its technology. Two years later, Endeca was founded with a focus on navigating document collections using structured metadata, a technique known as "faceted search." Google, seeing an opportunity to leverage its expertise in the search domain, introduced an enterprise search appliance in 2000.

In a few short years, enterprise search has grown into a multibillion-dollar market segment that is almost totally separate from the data warehouse market. Endeca has some tools for more traditional business intelligence, and some database vendors have worked to introduce text mining capabilities into their systems, but a complete, integrated solution for structured and unstructured enterprise data management remains unrealized.

Both enterprise search and data warehousing are technical solutions to the larger problem of leveraging the information resources of an organization to improve performance. As far back as 1944, MIT professor Kurt Lewin proposed "action research" as a framework that uses "a spiral of steps, each of which is composed of a circle of planning, action, and fact-finding about the result of the action." A more modern approach to the same problem can be found in Peter Senge's "Learning Organization" concept, detailed in his book *The Fifth Discipline* (Broadway Business). Both management theories rely heavily upon an organization's ability to adapt its actions after reflecting upon information collected from previous actions. From this perspective, an Information Platform is the infrastructure required by a Learning Organization to ingest, process, and generate the information necessary for implementing the action research spiral.

Having now looked at structured and unstructured data management, let's get back to the Facebook story.

The Cheetah and the Elephant

On the first day of logging the Facebook clickstream, more than 400 gigabytes of data was collected. The load, index, and aggregation processes for this data set really taxed the Oracle data warehouse. Even after significant tuning, we were unable to aggregate a day of clickstream data in less than 24 hours. It was clear we'd need to aggregate our logfiles outside of the database and store only the summary information for later querying.

Luckily, a top engineer from a large web property had recently joined our team and had experience processing clickstream data at web scale. In just a few weeks, he built a parallelized log processing system called Cheetah that was able to process a day of clickstream data in two hours. There was much rejoicing.

Despite our success, Cheetah had some drawbacks: first, after processing the clickstream data, the raw data was stored in archival storage and could not be queried again. In addition, Cheetah pulled the clickstream data from a shared NetApp filer with limited read bandwidth. The "schema" for each logfile was embedded in the processing scripts rather than stored in a format that could be queried. We did not collect progress information and we scheduled Cheetah jobs using a basic Unix utility called `cron`, so no sophisticated load-sharing logic could be applied. Most importantly, however, Cheetah was not open source. We had a small team and could not afford the resources required to develop, maintain, and train new users to use our proprietary system.

The Apache Hadoop project, started in late 2005 by Doug Cutting and Mike Cafarella, was a top candidate to replace Cheetah. Named after the stuffed elephant of Doug's son, the Hadoop project aimed to implement Google's distributed filesystem and MapReduce technologies under the Apache 2.0 license. Yahoo! hired Doug Cutting in January 2006 and devoted significant engineering resources to developing Hadoop. In April 2006, the software was able to sort 1.9 terabytes in 47 hours using 188 servers. Although Hadoop's design improved on Cheetah's in several areas, the software was too slow for our needs at that time. By April 2008, however, Hadoop was able to sort 1 terabyte in 209 seconds using 910 servers. With the improved performance numbers in hand, I was able to convince our operations team to stick three 500-gigabyte SATA drives in the back of 60 unused web servers, and we went forward with our first Hadoop cluster at Facebook.

Initially, we started streaming a subset of our logs into both Hadoop and Cheetah. The enhanced programmability of Hadoop coupled with the ability to query the historical data led to some interesting projects. One application involved scoring all directed pairs of interacting users on Facebook to determine their affinity; this score could then be used for search and News Feed ranking. After some time, we migrated all Cheetah workflows to Hadoop and retired the old system. Later, the transactional database collection processes were moved to Hadoop as well.

With Hadoop, our infrastructure was able to accommodate unstructured and structured data analysis at a massive scale. As the platform grew to hundreds of terabytes and thousands of jobs per day, we learned that new applications could be built and new questions could be answered simply because of the scale at which we were now able to store and retrieve data.

When Facebook opened registration to all users, the user population grew at disproportionately rapid rates in some countries. At the time, however, we were not able to perform granular analyses of clickstream data broken out by country. Once our Hadoop cluster was up, we were able to reconstruct how Facebook had grown rapidly in places such as Canada and Norway by loading all of our historical access logs into Hadoop and writing a few simple MapReduce jobs.

Every day, millions of semi-public conversations occur on the walls of Facebook users. One internal estimate put the size of the wall post corpus at 10 times the size of the blogosphere! Before Hadoop, however, the contents of those conversations remained inaccessible for data analysis. In 2007, a summer intern with a strong interest in linguistics and statistics, Roddy Lindsay, joined the Data team. Using Hadoop, Roddy was able to single-handedly construct a powerful trend analysis system called Lexicon that continues to process terabytes of wall post data every night; you can see the results for yourself at *http://facebook.com/lexicon*.

Having the data from disparate systems stored in a single repository proved critical for the construction of a reputation scoring system for Facebook applications. Soon after the launch of the Facebook Platform in May of 2007, our users were inundated with requests to add applications. We quickly realized that we would need a tool to separate the useful applications from those the users perceived as spam. Using data collected from the API servers, user profiles, and activity data from the site itself, we were able to construct a model for scoring applications that allowed us to allocate invitations to the applications deemed most useful to users.

The Unreasonable Effectiveness of Data

In a recent paper, a trio of Google researchers distilled what they have learned from trying to solve some of machine learning's most difficult challenges. When discussing the problems of speech recognition and machine translation, they state that, "invariably, simple models and a lot of data trump more elaborate models based on less data." I don't intend to debate their findings; certainly there are domains where elaborate models are successful. Yet based on their experiences, there does exist a wide class of problems for which more data and simple models are better.

At Facebook, Hadoop was our tool for exploiting the unreasonable effectiveness of data. For example, when we were translating the site into other languages, we tried to target users who spoke a specific language to enlist their help in the translation task. One of our Data Scientists, Cameron Marlow, crawled all of Wikipedia and built character trigram frequency counts per language. Using these frequency counts, he built a simple classifier

that could look at a set of wall posts authored by a user and determine his spoken language. Using this classifier, we were able to actively recruit users into our translation program in a targeted fashion. Both Facebook and Google use natural language data in many applications; see Chapter 14 of this book for Peter Norvig's exploration of the topic.

The observations from Google point to a third line of evolution for modern business intelligence systems: in addition to managing structured and unstructured data in a single system, they must scale to store enough data to enable the "simple models, lots of data" approach to machine learning.

New Tools and Applied Research

Most of the early users of the Hadoop cluster at Facebook were engineers with a taste for new technologies. To make the information accessible to a larger fraction of the organization, we built a framework for data warehousing on top of Hadoop called Hive.

Hive includes a SQL-like query language with facilities for embedding MapReduce logic, as well as table partitioning, sampling, and the ability to handle arbitrarily serialized data. The last feature was critical, as the data collected into Hadoop was constantly evolving in structure; allowing users to specify their own serialization format allowed us to pass the problem of specifying structure for the data to those responsible for loading the data into Hive. In addition, a simple UI for constructing Hive queries, called HiPal, was built. Using the new tools, non-engineers from marketing, product management, sales, and customer service were able to author queries over terabytes of data. After several months of internal use, Hive was contributed back to Hadoop as an official subproject under the Apache 2.0 license and continues to be actively developed.

In addition to Hive, we built a portal for sharing charts and graphs called Argus (inspired by IBM's work on the Many Eyes project), a workflow management system called Databee, a framework for writing MapReduce scripts in Python called PyHive, and a storage system for serving structured data to end users called Cassandra (now available as open source in the Apache Incubator).

As the new systems stabilized, we ended up with multiple tiers of data managed by a single Hadoop cluster. All data from the enterprise, including application logs, transactional databases, and web crawls, was regularly collected in raw form into the Hadoop distributed filesystem (HDFS). Thousands of nightly Databee processes would then transform some of this data into a structured form and place it into the directory of HDFS managed by Hive. Further aggregations were performed in Hive to generate reports served by Argus. Additionally, within HDFS, individual engineers maintained "sandboxes" under their home directories against which prototype jobs could be run.

At its current capacity, the cluster holds nearly 2.5 petabytes of data, and new data is added at a rate of 15 terabytes per day. Over 3,000 MapReduce jobs are run every day, processing 55 terabytes of data. To accommodate the different priorities of jobs that are run on the cluster, we built a job scheduler to perform fair sharing of resources over multiple queues.

In addition to powering internal and external reports, a/b testing pipelines, and many different data-intensive products and services, Facebook's Hadoop cluster enabled some interesting applied research projects.

One longitudinal study conducted by Data Scientists Itamar Rosenn and Cameron Marlow set out to determine what factors were most critical in predicting long-term user engagement. We used our platform to select a sample of users, trim outliers, and generate a large number of features for use in several least-angle regressions against different measures of engagement. Some features we were able to generate using Hadoop included various measures of friend network density and user categories based on profile features.

Another internal study to understand what motivates content contribution from new users was written up in the paper "Feed Me: Motivating Newcomer Contribution in Social Network Sites," published at the 2009 CHI conference. A more recent study from the Facebook Data team looks at how information flows through the Facebook social graph; the study is titled "Gesundheit! Modeling Contagion through Facebook News Feed," and has been accepted for the 2009 ICWSM conference.

Every day, evidence is collected, hypotheses are tested, applications are built, and new insights are generated using the shared Information Platform at Facebook. Outside of Facebook, similar systems were being constructed in parallel.

MAD Skills and Cosmos

In "MAD Skills: New Analysis Practices for Big Data," a paper from the 2009 VLDB conference, the analysis environment at Fox Interactive Media (FIM) is described in detail. Using a combination of Hadoop and the Greenplum database system, the team at FIM has built a familiar platform for data processing in isolation from our work at Facebook.

The paper's title refers to three tenets of the FIM platform: Magnetic, Agile, and Deep. "Magnetic" refers to the desire to store all data from the enterprise, not just the structured data that fits into the enterprise data model. Along the same lines, an "Agile" platform should handle schema evolution gracefully, enabling analysts to work with data immediately and evolve the data model as needed. "Deep" refers to the practice of performing more complex statistical analyses over data.

In the FIM environment, data is separated into staging, production, reporting, and sandbox schemas within a single Greenplum database, quite similar to the multiple tiers inside of Hadoop at Facebook described earlier.

Separately, Microsoft has published details of its data management stack. In papers titled "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks" and "SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets," Microsoft describes an information platform remarkably similar to the one we had built at Facebook. Its infrastructure includes a distributed filesystem called Cosmos and a system for parallel data processing called Dryad; it has even invented a SQL-like query language called SCOPE.

Three teams working with three separate technology stacks have evolved similar platforms for processing large amounts of data. What's going on here? By decoupling the requirements of specifying structure from the ability to store data and innovating on APIs for data retrieval, the storage systems of large web properties are starting to look less like databases and more like dataspaces.

Information Platforms As Dataspaces

Anecdotally, similar petabyte-scale platforms exist at companies such as Yahoo!, Quantcast, and Last.fm. These platforms are not quite data warehouses, as they're frequently not using a relational database or any traditional data warehouse modeling techniques. They're not quite enterprise search systems, as only some of the data is indexed and they expose far richer APIs. And they're often used for building products and services in addition to traditional data analysis workloads. Similar to the brain and the library, these shared platforms for data processing serve as the locus of their organization's efforts to ingest, process, and generate information, and with luck, they hasten their organization's pace of learning from empirical data.

In the database community, there has been some work to transition the research agenda from purely relational data management to a more catholic system for storage and querying of large data sets called a "dataspace." In "From Databases to Dataspaces: A New Abstraction for Information Management" (*http://www.eecs.berkeley.edu/~franklin/Papers/dataspaceSR.pdf*), the authors highlight the need for storage systems to accept all data formats and to provide APIs for data access that evolve based on the storage system's understanding of the data.

I'd contend that the Information Platforms we've described are real-world examples of dataspaces: single storage systems for managing petabytes of structured and unstructured data from all parts of an organization that expose a variety of data access APIs for engineering, analysis, and reporting. Given the proliferation of these systems in industry, I'm hopeful that the database community continues to explore the theoretical foundations and practical implications of dataspaces.

An Information Platform is the critical infrastructure component for building a Learning Organization. The most critical human component for accelerating the learning process and making use of the Information Platform is taking the shape of a new role: the Data Scientist.

The Data Scientist

In a recent interview, Hal Varian, Google's chief economist, highlighted the need for employees able to extract information from the Information Platforms described earlier. As Varian puts it, "find something where you provide a scarce, complementary service to something that is getting ubiquitous and cheap. So what's getting ubiquitous and cheap? Data. And what is complementary to data? Analysis."

At Facebook, we felt that traditional titles such as Business Analyst, Statistician, Engineer, and Research Scientist didn't quite capture what we were after for our team. The workload for the role was diverse: on any given day, a team member could author a multistage processing pipeline in Python, design a hypothesis test, perform a regression analysis over data samples with R, design and implement an algorithm for some data-intensive product or service in Hadoop, or communicate the results of our analyses to other members of the organization in a clear and concise fashion. To capture the skill set required to perform this multitude of tasks, we created the role of "Data Scientist."

In the financial services domain, large data stores of past market activity are built to serve as the proving ground for complex new models developed by the Data Scientists of their domain, known as Quants. Outside of industry, I've found that grad students in many scientific domains are playing the role of the Data Scientist. One of our hires for the Facebook Data team came from a bioinformatics lab where he was building data pipelines and performing offline data analysis of a similar kind. The well-known Large Hadron Collider at CERN generates reams of data that are collected and pored over by graduate students looking for breakthroughs.

Recent books such as Davenport and Harris's *Competing on Analytics* (Harvard Business School Press, 2007), Baker's *The Numerati* (Houghton Mifflin Harcourt, 2008), and Ayres's *Super Crunchers* (Bantam, 2008) have emphasized the critical role of the Data Scientist across industries in enabling an organization to improve over time based on the information it collects. In conjunction with the research community's investigation of dataspaces, further definition for the role of the Data Scientist is needed over the coming years. By better articulating the role, we'll be able to construct training curricula, formulate promotion hierarchies, organize conferences, write books, and fill in all of the other trappings of a recognized profession. In the process, the pool of available Data Scientists will expand to meet the growing need for expert pilots for the rapidly proliferating Information Platforms, further speeding the learning process across all organizations.

Conclusion

When faced with the challenge of building an Information Platform at Facebook, I found it helpful to look at how others had attempted to solve the same problem across time and problem domains. As an engineer, my initial approach was directed by available technologies and appears myopic in hindsight. The biggest challenge was keeping focused on the larger problem of building the infrastructure and human components of a Learning Organization rather than specific technical systems, such as data warehouses or enterprise search systems.

I'm certain that the hardware and software employed to build an Information Platform will evolve rapidly, and the skills required of a Data Scientist will change at the same rate. Staying focused on the goal of making the learning process move faster will benefit both organizations and science. The future belongs to the Data Scientist!