# Big Data: It's Not the Size That Matters

Matthew Gordon*

## Introduction

"Big Data" has become a ubiquitous phrase in technology circles. The internet and media landscape is littered with superlatives, describing data as a "flood," "deluge," as "astronomical," and "exploding." Gorillas, elephants, and skyscrapers all make infographic appearances. However, the promise of big data lies not in the amount of data we can generate, collect, or store, but in the ability to use all the data at our disposal to make informed decisions. And while scale poses unique challenges, the same analytical methodologies apply to data at all scales and across many disciplines, whether you have a single spreadsheet or a yottabyte of documents. In this article, I present 1) a framework for understanding the needs of data analysis projects, based on "The Four Pillars"[1] and 2) a discussion of real-world logistical considerations when implementing such a system. Finally, I will discuss privacy and civil liberties, how proper system design can remove the friction between privacy and sharing, and the appropriate uses of automated data analysis.

## I. Measuring Data

Before embarking on a discussion of big data analytics, we must start by laying out some basic categories and measurement standards as a frame of reference from which to understand what is even meant by "big." Roughly speaking, we will consider three types of data: *structured, unstructured*, and *semi-structured*. *Structured data* is typically represented as "rows and columns": lists of numeric, date, enumerated, or short text values, with headers indicating what each column represents. A typical example might be a spreadsheet containing headers like Name (a text value), Gender (an enumerated value, either Male or Female), Date of Birth (a date), and Income (a numeric value). Such tables of data may contain anywhere from dozens up to hundreds of thousands of rows (Microsoft Excel 2010, for instance, limits users to approximately one million rows).[2] However, practically speaking, spreadsheets greater than a few hundred megabytes get unwieldy pretty quickly.

At larger scales, it becomes necessary to implement systems to make it practical to access and modify data conveniently. While storing data in files on disk is practically unlimited in capacity, the time to search and return results scales linearly with the size of the data, making it an impractical solution at

* Forward Deployed Engineer for Legal Intelligence, Palantir Technologies. 

1. *Palantir as Intelligence Infrastructure*, Palantir Tech., http://www.palantir.com/library/.

2. *Excel Specifications and Limits*, Microsoft, http://office.microsoft.com/en-us/excel-help/excel-specifications-and-limits-HP010342495.aspx?CTT=5&origin=HP005199291.

large scale. Big data storage technology seeks to provide methods that have storage and modification times that are effectively independent of scale, such that retrieving a record from a ten megabyte data store should take about as long as retrieving one from a ten terabyte one (which, ideally, should be fast).

*Relational databases* are the most common form of structured data at this scale, including commercial products like those made by Oracle and IBM, or free software such as MySQL and PostgreSQL. Relational databases are designed to allow thousands of users to simultaneously access and modify the contents while maintaining a consistent state and high performance across the entire system. Practically speaking, such databases become difficult to maintain for most organizations when they grow into the low tens of terabytes, or over a few billion rows. (A good rule of thumb for most common database applications is that a single database row is probably one to ten kilobytes of data, depending on the number of columns and the database content.)

At still greater data scale, *distributed key-value stores* have become the primary way of storing structured data in the hundreds of terabytes to petabyte scale in a way that still allows for rapid access. Software such as Cassandra,[3] BigTable,[4] and MongoDB[5] can be split out across an arbitrary number of servers to improve access times. In exchange for high scalability and performance, the types of operations available are highly restricted compared to relational databases, typically limited to retrieving values and adding new ones (with modifications and deletions being disallowed or carrying a significant performance penalty). Additionally, they do not offer the same stringent guarantees about the correctness of the data; the system is guaranteed to be *eventually* consistent, but at any given moment in time, there may be small inconsistencies between what different users will get from the same query.

*Unstructured data* is data that is primarily natural language text, such as electronic documents, text files, or web pages (though under the hood, web pages are actually highly structured, as pressing Control+U in Firefox or Chrome browsers will demonstrate.) While structured data is usually organized into databases, unstructured data is often found in the wild in the many millions of individual files churned out daily and stored on hard drives, web servers, or in document management systems. Storing documents can be scaled easily by simply adding more hard drives. However, in order to be searchable in a reasonable fashion, such data has to be indexed, a process in which the files are individually processed and broken down by keyword, so that, for instance, Google can give you a comprehensive list of which web pages contain the phrase "big data" in less than a second.

*Semi-structured data*, as the name implies, lies somewhere in between the

3. *The Apache Cassandra Project*, THE APACHE FOUND., http://cassandra.apache.org.

4. FAY CHANGE ET AL., BIGTABLE: A DISTRIBUTED STORAGE SYSTEM FOR STRUCTURED DATA (2006) (Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation).

5. MONGODB, http://www.mongodb.org.

two. E-mail is the most common example: while the content is primarily unstructured, e-mails contain a large amount of structured information as well, such as the sender, receivers, the date on which the e-mail was sent and received, and even the exact path through the internet that the e-mail took to reach your computer. Social media such as Twitter is also a rich source of semi-structured data, with half a billion 140 character mini-documents being tweeted each day.[6]

With unstructured and semi-structured data, the number of documents becomes a more relevant measure of size. Some approximate numbers for reference:

- A typical PC may contain several hundred to several thousand documents.
- A typical email account may contain tens of thousands of e-mails.
- The manner in which large organizations use e-mail is highly variable: in some organizations, people may receive only 10,000 e-mails a year and in others, it may be closer to a million. Across a large organization, hundreds of millions of e-mails a year is not atypical.

## II. FROM DESKTOP TO DATACENTER

As storage prices drop radically and processing power increases exponentially, organizations collect data at increasing rates, with the intention of using that data somehow, for some purpose. The number of different databases and document stores proliferate, but often in an organic, quasi-unplanned fashion, making it difficult to know where the enterprises' critical knowledge lives. Worse yet, the data is pushed into databases with only rudimentary user interfaces, and data spread across multiple incompatible databases can't be combined or compared. The ability to answer data-centric questions is often the domain of a handful of individuals at the organization familiar with SQL (*structured query language,* a standard database query language) or *Hadoop* (a platform for gathering statistics from distributed key-value stores). If someone wants to know, for instance, how many widgets were sold in Chicago in the month of June compared to the same month last year, you may have to submit a request to an individual, and then go get a cup of coffee while you wait for the results. Depending on how busy they are, you may be able to get your coffee in Brazil before the results come back. The reason for this state of affairs is that *data warehouses are built for the convenience of the data, not for the users*. If the goal of collecting this data is to make it possible to use it for decision making, we must strive to invert this paradigm: *Big Data is about making insight accessible* by making it as easy to query and understand an entire data warehouse as it is to use a single spreadsheet – or, better yet, easier. By making

6. Daniel Terdiman, *Report: Twitter Hits Half a Billion Tweets a Day*, CNET (Oct. 26, 2012), http://news.cnet.com/8301-1023_3-57541566-93/report-twitter-hits-half-a-billion-tweets-a-day/.

it simple to ask and answer questions in an intuitive way, anyone can make decisions based on *all* the available data.

The problems posed by large-scale data analysis are ubiquitous and span the gamut of government and commercial organizations: intelligence, defense, local and federal law enforcement, counter-fraud, business intelligence, finance, and legal all have their own data problems, and they each have their own unique goals. However, their needs are all quite similar. All organizations require essentially the same four things, which we refer to as "The Four Pillars," to make use of their data: *data integration*, *search and discovery*, *knowledge management*, and *collaboration*. In some senses, the September 11th tragedy was the seminal big data problem of the 21st century, and I will use it as a concrete example several times in helping to bring context to the discussion that follows.

## III. DATA INTEGRATION

Any organization of non-trivial size will have multiple databases, services, and document stores that contain valuable information. In government organizations, where data is abundant, this often takes the form of databases and document stores owned by multiple organizations, each with differing formats and standards. In the private sector, companies often have transactional data, employee data, marketing data, and network logs (to name a few) in different "siloed" systems. Legal matters with a financial component such as anti-trust or securities litigation will usually have a document management system which contains all of the produced documents, but commingled with the documents are often spreadsheets and databases with highly relevant but hard-to-access transactional information. And, in all of these cases, there is frequently open-source data such as credit reports, marketing data, web pages, and social media which can be added to enrich the available information with vital context, but which are not easily combined with the in-house data stores. All of these are examples of data "stove-piping," in which information sources are housed in separate, incompatible systems, making it difficult to come to a holistic understanding. In discussing the September 11th tragedy, the 9/11 Commission specifically called out the importance of data integration in preventing future terrorist attacks, stating that "[t]he 9/11 story teaches the value of integrating strategic intelligence from all sources."[7]

The goal of data integration should be to provide not only a mechanism for importing and normalizing data from multiple sources, but also a framework for combining both structured and unstructured data together on the same continuum. A simple but powerful example is ferretting out insider trading. Such investigations may rely on trading records from a spreadsheet, phone records

7. NAT'L COMM. ON TERRORIST ATTACKS UPON THE U.S., THE 9/11 COMMISSION REPORT: EXECUTIVE SUMMARY 21 (2004).

from a database, e-mails from an enterprise IT system, and company earnings announcements from the internet. None of these can demonstrate insider trading conclusively, but taken together, they can paint a very compelling picture.

In addition to being able to access and import data from many sources, it is key that there be a mechanism for *normalization*. It is not enough, for instance, to import phone numbers if some systems represent phone numbers as a simple string of ten digits, some include parentheses and dashes, and some include country codes. The system must be able to standardize how a phone number is represented. Otherwise, cross-referencing phone numbers between, say, a list of phone calls and an employee database in order to find out who made a phone call would be impossible.

Finally, *data cleaning* is also crucial. In a large database containing social security numbers, one is guaranteed to find many individuals with SSNs listed as "0," "000-00-0000," "111-11-1111," " - - ," or similarly nonsensical patterns. While it should have been the job of the original database designer to guard against input of such garbage, it is cold comfort to blame it on the consultant who built the system twenty years ago. The presence of such values will often lead to spurious associations of unrelated individuals for the hapless analyst who attempts to cross-reference multiple data sources for matching social security numbers and finds thousands of people with the SSN "000-00-0000." In addition to garbage input, names present another common but deadly data hygiene issue, especially in law enforcement, where information is often collected orally and entered in the field. So-called fat fingering creates many problems when trying to search for individuals, particularly when searching by first and last name.

## IV. SEARCH AND DISCOVERY

Once data has been imported and stored, users must be able to query the data to retrieve answers to specific questions. Keyword searching is one powerful and simple example, wherein users can enter word or phrases and then documents, web pages, or e-mails containing those keywords are brought back. However, with a combination of structured and unstructured data, keyword searching is only a small fraction of the necessary search functionality. Searching phone calls, for instance, requires users to search by relationship in order to find phones owned by a person or organization, and by date and time. When planning a patrol route, military personnel may want to specify a geospatial search along a route for attacks of a particular type and in a particular time range, in order to prepare appropriately. The ability to perform such searches in a timely, accurate, and accessible fashion can literally mean the difference between life and death. There is also a qualitative difference between searching that takes seconds versus searching that takes minutes or hours; with quick, responsive search times, users are quickly able to iterate through multiple queries, and test many hypotheses serially, asking and answering questions in a natural and intuitive way. This stream of analytic consciousness is extremely

important to deriving insight from data. On the other hand, if users must plan queries carefully and wait a long time for the results, creative data exploration becomes almost impossible.

## V. KNOWLEDGE MANAGEMENT

In order to put information to use, it is important that we be able to answer certain questions about the data itself, such as:

- Where did a piece of information come from?
- Who has looked at it?
- How reliable is it?
- When was it last changed, and why?
- Who is allowed to see it?

One application of knowledge management functionality is compliance with regulatory requirements for data deletion such as those contained in 28 C.F.R. § 23, which governs how long certain types of data can be retained by law enforcement agencies.[8] Accurately complying with the regulation requires knowledge of when a piece of data was entered, where it came from, and when it was last accessed. Being able to trace analysis back to its source material and maintaining the security of classified documents also depend on knowledge management. All of these functions are related in that they deal not just with the data itself, but with data *about* the data itself. They must be built into the system at the ground level; failure to plan for knowledge management in the design of a system makes it almost impossible to implement such functionality later on. For instance, if you import large amounts of data, produce an analysis, and use that analysis as the basis for an arrest, you may be required several years down the road to provide justification of your conclusions to a defendant or a court. If knowledge management was not part of the original system, you may be unable to explain where the data came from in the first place, wasting years of effort.

Classified data handling provides another excellent example of the importance of knowledge management. Consider the "manila envelope" model of data classification: imagine that I have an envelope with a document that contains nineteen facts about a dangerous individual that are classified secret, and one fact that is classified top-secret. If you have a secret clearance only, I can't share the document with you at all, even if 95 percent of the material in the envelope is both sharable and relevant to your investigation. On the other hand, a knowledge management system that provides granular access control to data allows me to share data with other people seamlessly, without worrying about their access level: the system ensures that they are only given access to the information that they are permitted to see, and automatically hides the

8. Criminal Intelligence Systems Operating Policies, 28 C.F.R. § 23, *et seq*.

top-secret information, giving me the freedom to share important and relevant information without having to worry about data leaks.

These types of considerations loom large among the reasons that agencies find it difficult to share intelligence while maintaining policy, security, privacy, and civil liberties controls. Conversely, though, failure to address them directly led to the intelligence failures that led up to the September 11th attacks.[9]

## VI. COLLABORATION

Lack of collaboration and coordination within and among agencies was a key finding of the 9/11 commission: "The FBI lacked the ability to know what it knew; there was no effective mechanism for capturing or sharing its institutional knowledge."[10] Collaboration within an enterprise allows users to avoid duplicating effort and to build off of the insight of others. When analysts, detectives, or lawyers keep notes on a memo pad or on their personal computers, the opportunities for others to learn from their insight is limited; sharing this knowledge so that it is accessible to the entire organization makes it easier to get a handle on the "unknown unknowns" by leveraging the insights and research of others. And sourcing that data back to the analyst who entered or curated it generates opportunities for organic collaboration, by helping connect experts on particular subjects with those within their organization who require their help.

On a larger scale, collaboration between agencies opens opportunities for connecting the dots because bad actors don't respect jurisdictions or geographic boundaries. And enabling collaboration means not just making it possible, but making it practical. Working with the Urban Areas Security Initiative to improve data sharing among municipal police departments, I found that, for the most part, data sharing still took place via the old "gumshoe method": calling a buddy at another department to see if they "knew anything" about a case. Under this arrangement, the barrier to collaboration was relatively high, meaning that it was only used for highly important cases.

Giving detectives the ability to seamlessly search their own records management systems and those of surrounding police departments made it possible for detectives from small, resource constrained departments to run every lead, not just for important cases, but for every case. In one example, officers planning to execute a search on a suspect's home ran a phone number associated with the suspect, and found that it was associated with a police report from another agency. The report indicated that the suspect's girlfriend had been involved in a

9. "Each intelligence agency has its own security practices, outgrowths of the Cold War . . . . Current security requirements nurture overclassification and excessive compartmentation of information among agencies. Each agency's incentive structure opposes sharing, with risks (criminal, civil, and internal administrative sanctions) but few rewards for sharing information." NAT'L COMM. ON TERRORIST ATTACKS UPON THE U.S., THE 9/11 COMMISSION REPORT: FINAL REPORT OF THE NATIONAL COMMISSION ON TERRORIST ATTACKS UPON THE UNITED STATES 417 (2004).

10. *Id.* at 77.

drive-by shooting using the suspect's gun. Officers used this information to obtain a no-knock warrant, and the SWAT team found several weapons in the home, in addition to drugs. Easy and quick access to collaboration made a material contribution to officer safety by shedding light on the unknown unknowns.

## VII. GETTING FROM HERE TO THERE: THE THREE P'S

Big data frequently means big project, requiring coordination between many individuals, agencies, and departments. How does an organization get from theory to execution on such a project? There are essentially three facets common to almost all such projects: *Plumbing*, *Payment*, and *Politics*. I will examine each of these in turn, and discuss some strategies for handling them.

### *A. Plumbing*

The process of getting data from point A to point B is, theoretically, the simplest of the obstacles to implementation. Plumbing is what is referred to in the software world as a "Small Matter of Programming" (or a *SMOP*): All of the pieces exist (data integration platforms, database connectors, network protocols, etc.) – they simply must be strung together in the proper fashion. It almost goes without saying, of course, that the term SMOP is meant sardonically: systems integrators and IT professionals often underestimate the difficulty of this piece of the process by orders of magnitude, because its apparent simplicity belies the bewildering array of standards and specifications,[11] the inevitable problems caused by taking solutions that work in the lab and scaling them up to real-world conditions, and the broad and unanticipated range of problems that present themselves with real-world data, which is much less clean than data in a test environment. In addition, the issues of data normalization and cleaning described in the previous section about data integration are non-trivial, and require experience to be handled well and performed efficiently.

Plumbing also concerns getting the data back out, including, but not limited to: search; visualizing results in different modalities such as graphs, charts, and maps; and generating reports. The world's best analysis is worthless unless you can convince someone else of your findings. Just as importantly, systems should export data in well-documented and commonly acceptable formats, as well as exposing functionality through a well-documented *application programming interface* or *API.* An API is simply a specification for how pieces of software can communicate. Many legacy systems still in existence did not concern themselves with such niceties in their original specification, often by design: by making it impossible to export data from the system, vendors ensure that the

11. "The nice thing about standards is that you have so many to choose from." ANDREW TANENBAUM, COMPUTER NETWORKS 254 (2d ed. 2003).

system won't be replaced, or at least not without costing huge amounts of money to the end users. This is referred to as *vendor lock-in*.

### *B. Payment*

A discussion of contracting and cost-benefit analysis of software projects is well beyond the scope of this article, but a few comments on trends in enterprise software development are worth mentioning. The traditional model of enterprise software procurement, especially within government, follows a predictable pattern: Months or years are spent hashing out giant binders full of specifications. One of a handful of well-known systems integrators are contracted to take on the project and create a custom software system. Gantt charts are presented with timelines and budgets ranging from optimistic to delusional. By the time software development starts, trends have overtaken the original specification, scale has grown, and the project specification must be modified midstream. Downstream pieces of the specification that depended critically on the upstream functionality must now be scrapped, expanded, or wholly rethought. Gantt charts are redrawn; budgets limits are shredded; garments are rent; performance bonuses are awarded all around. In due course, the switch is turned on, and, lo and behold, the system grinds to a halt before it even starts up. More Gantt charts are produced, more budgets are rewritten, and the process continues until the goals of the project have been defined downward far enough to consider it a success, or until it is deemed a failure and abandoned.[12]

The Age of Austerity (roughly, 2008 to the present) demands greater accountability and better results for less money, and a*gile software development* seeks to address these failures. Agile development emphasizes a tight, iterative development cycle and close customer collaboration to develop systems that solve problems efficiently and in a cost-effective fashion. Instead of laying out all the requirements in excruciating detail up front, simple, mission-focused goals are enunciated in place of detailed specifications, such as, "We would like to be able to search all of our data sources from one place," or, "We need to be able to figure out where all our officers are at any given time, so that we can respond to emergencies efficiently." The developers deliver multiple releases in close succession, working with the end users of the system and the decision makers of the organization to gauge whether or not the delivered product is meeting their needs and what their top priorities are for the following releases. This process greatly reduces the risk of delivering a product which never works or meets any of the users' needs, and it brings useful software to the end users quickly. Modularity (software pieces that can function independently of each other), reliance on proven off-the-shelf and open-source solutions over custom-built

12. For a very amusing and only semi-fictional take on a similar government project, see the film *The Pentagon Wars*, about the development of the ill-fated Bradley Fighting Vehicle. THE PENTAGON WARS (HBO 1998).

systems, and emphasis on open APIs and open standards are also important ways to reduce risk and improve outcomes.

### *C. Politics*

Every large software project threatens someone else's budget, department, or project. Under-appreciation of this fact means that people will almost always underestimate the ferocity and tenacity with which a cornered IT administrator will lash out against new software projects. As a result, almost every data integration project I have been involved in has followed a pattern of procuring software first, and negotiating terms of data access and memoranda of understanding later. This is a fatal mistake, because it puts IT in charge of critical policy decisions, such as what data sources will be included in the project, how often the systems will exchange data, who will have access to the systems, and how secure and/or convenient that access will be. These are decisions that should be made based on considerations about the objectives driving the project and the needs of the users, and not whether or not you gave your IT administrator a generous enough holiday bonus last year.

While internecine warfare is a natural part of the ecosystem of large organizations, there are a few things you can do to grease the rails of large projects to make the process less painful.

1) *Align everyone's incentives*. Some people in your IT organization will see budgets for their pet projects shrink as a result of your new project. Make sure they receive a stake in the success of the new project by giving them new direct reports, more resources, and partial credit for success, encouraging them to dedicate resources to making it happen. Other people in your organization already have projects that may overlap with yours, and will want to see yours fail. Avail yourself of their expertise early on, and make it clear that you think their input is invaluable to the success of the project. Find ways to interoperate with their systems, and emphasize the differences, not the commonalities.

2) *Work out MOUs and NDAs in advance*. Before you even get started on a project, talk to all the stakeholders and work out a framework for data sharing. Discuss security data. Bring in their IT team to vet the security plan. Make sure the goals of the project are agreed upon by all the top stakeholders, and make sure they let their organizations know that it's a priority. When everyone's agreed, sign a joint memorandum of understanding (MOU) and/or a non-disclosure agreement (NDA) that lays out what everyone has agreed upon. This will at least give everyone a frame of reference for what to expect down the line.

3) *Early successes are important: Start fast out of the gate.* Start out with the low-hanging fruit. Find the biggest problems that you can tackle with the least resistance, and show that your new system addresses them. Train users to adopt the new system, and leverage their experiences and mission wins to energize the rest of the organization. Nobody wants to be far

behind on the Next Big Thing™. If everyone starts hearing that your project saves time/catches bad guys/improves efficiency by 300%, everyone will want to get on board.

4) *Be willing to compromise, but not too much.* You will inevitably be asked to compromise your plan in the name of security, policy, or both. Having a flexible system with an open architecture makes it easier to accept compromises: if one part of the organization refuses you direct access to their database, but instead wants to upload "cuts" of data to a server every night, this can be inconvenient, but at least it's a starting point for data sharing, which can be built on later (see #3: it's better to get out there fast than it is to die on the vine). On the other hand, if they offer a grand plan to build a data warehouse that you can access, and it will only cost an extra million dollars and take six months, it may be time to put your foot down.

Similarly, security credentialing and auditing can be an important source of standardization. It can also be a bottomless pit from which projects never emerge. Be willing to comply with standard and reasonable security requirements, but also be insistent that they be tied to a realistic assessment of what's appropriate for the size and scope of the project. This is where having a *threat model* is extremely important: it's easy for information security officers to run down a procrustean checklist of requirements and find one that you can't or don't satisfy. But keeping the discussion tied to realistic expectations about cost/benefit tradeoff of particular security requirements helps ground the discussion in a common language that IT, contractors, and management can understand. For instance, it's not uncommon to require that sensitive data be encrypted "at rest" (i.e., in the database or on the hard drive). This is a good precaution if the data is being stored on a laptop. However, if it is stored on servers that are racked in a secured server room requiring two-person biometric access, requiring encryption-at-rest might be overkill.

## VIII. TECHNOLOGICAL APPROACHES TO PROTECTING PRIVACY AND CIVIL LIBERTIES

Often, privacy and civil liberties (PCL) have been afterthoughts in data sharing systems, bolted on after the system has been created in order to satisfy regulatory concerns. As a result, the perception among many is that PCL lives at one end of a one-dimensional continuum, anchored by data sharing and functionality at the other: any improvement in privacy requires a reduction in functionality, and vice versa. However, this perception arises largely because of the clumsy way in which most of these systems are implemented. If technology for accommodating PCL concerns is built in from the start, there is no need for a trade-off. *Privacy and civil liberties should not be an afterthought.* For users to make optimal use of the data available, PCL considerations must be dealt with during the earliest phases of planning. There are two reasons for this: First, making PCL controls central to the design of the system makes them harder to

circumvent. If they are built on top of an existing system, it is much easier to find ways around them. That said, individuals with nefarious intent are often not the primary concern. They represent a very small fraction of the problem, and you can never truly secure a system against a persistent and dedicated internal attacker. The primary vectors for PCL violations come instead in the form of casual disregard and inconvenience.

Casual disregard can take the form of "fishing expeditions," where users poke around the system aimlessly, trying to find "something interesting," or misuse the system to spy on friends or family. Inconvenience is more pernicious, because it is not really a vice; clumsily designed and implemented controls make it difficult for users to make use of the data at their disposal, requiring workarounds that often leave data exposed. For instance, if your primary analytical application doesn't have mapping capabilities (or they are poorly implemented and hard to use), you may be tempted to export your data onto a flash drive and import it into a less secure system (e.g., a laptop) where you can perform the necessary analysis. If this becomes standard practice, the likelihood that data will get misplaced or lost approaches a certainty given enough time.

There are a number of things one might consider in the design and implementation of a software system that will help us walk the line between privacy and functionality in such a way that users can both have their cake and eat it too.

1. *Data sourcing*: The system should track the provenance of every piece of data at a granular level: where it comes from, when it was changed, and why it changed. Data sourcing not only helps us assess the accuracy of data, it also enables redress in cases in which an individual wishes to challenge his or her inclusion in a database.

2. *Granular access controls*: We discussed granular access controls above as a core piece of knowledge management, but it is also key to PCL: the ability to share data without revealing identifying information allows analysts to make full use of the data at their disposal while maintaining maximum privacy. There's no reason to share an entire dossier (or an entire database of dossiers) if it is just as easy and useful to share a redacted version. Lack of granular access control forces users to choose between privacy and sharing.

3. *Audit logs*: Flexible and configurable audit logging can curb casual disregard by making it easy to spot when users are fishing or using the system for inappropriate purposes. Requiring users to enter a case number or a predicate before running a search ensures that they know that they must have a justifiable reason – or suffer the consequences. Audit logging also enables redress in the case of data leaks, by allowing administrators to know who had access to the leaked data.

4. *Open APIs and Open Standards*: Closed systems force users to work around problems in creative ways, often to the detriment of privacy

controls. Agile design and open standards reduce the necessity for such workarounds: if it's possible to build on or tweak the existing system, there's less incentive to push data to less secure systems.

## IX. NEVER SEND A COMPUTER TO DO A PERSON'S JOB

Big data can be an amazing tool for decision making and investigation: tiny threads of information can be brought together from innumerable sources to create a rich tapestry of intelligence. Done correctly, such insight can be accessible to everyone, not just trained "data scientists." And with a single, central system in place for integrating, normalizing, and accessing data, it becomes possible to go beyond search and discovery at the individual scale and implement algorithms to automatically surface "suspicious" individuals or investigative leads. There are a number of approaches for doing so, ranging from simple link and keyword analysis to sophisticated machine learning algorithms. A discussion of the pros and cons of these approaches to automated search and discovery are beyond the scope of this review.

But one guiding principle is useful to bear in mind: *problems involving adaptable, human adversaries require context that can only be provided by human analysts.* The years since 9/11 have left us with countless examples of failed projects to create an automatic "Find Terrorist" button.[13] Contrast this with PayPal's ascendance in the micropayments space: while other companies relied on automated algorithms to detect and shut down fraud, PayPal developed a tool for leveraging the huge amount of data at their disposal and gave it to human analysts, which allowed it to keep fraud-related losses down much lower than its competitors.[14] This approach was successful precisely because they were competing against adaptive human adversaries: algorithms alone could not adapt fast enough to prevent fraudsters from finding ways to outsmart the system. But the combination of powerful computers and talented analysts was key to their success. Big data is only as useful as the expertise and understanding brought to the table by its users. Stated colloquially: it's not how big your data is, it's what you do with it.

13. *See, e.g.*, Eric Lichtblau & James Risen, *Hiding Details of Dubious Deal, U.S. Invokes National Security*, N.Y. TIMES, Feb. 19, 2011, at A1.

14. Siobahn Gorman, *How Team of Geeks Cracked Spy Trade*, WALL ST. J., Sept. 4, 2009, at A1.

***