

Data Finds Data

Jeff Jonas and Lisa Sokol

Introduction

NEXT-GENERATION “SMART” INFORMATION MANAGEMENT SYSTEMS WILL NOT RELY ON USERS DREAMING UP smart questions to ask computers; rather, they will automatically determine if new observations reveal something of sufficient interest to warrant some reaction, e.g., sending an automatic notification to a user or a system about an opportunity or risk.

An organization can only be as smart as the sum of its perceptions. These perceptions come in the form of observations—observations collected across the various enterprise systems, such as customer enrollment systems, financial accounting systems, and payroll systems. With each new transaction an organization learns something. It is at the moment something is learned that there exists an opportunity, in fact an obligation, to make some sense of what this new piece of data means and respond appropriately. For example, does the address change on the customer record now reveal that this customer is connected to one of your top 50 customers? If an organization cannot evaluate how new data points relate to its historical data holding in real time, the organization will miss opportunities for action.

When the “data can find the data,” there exists an opportunity for the insight to find the user.

How data finds data is a statement about *discoverability*, the degree to which previous information can be located and correlated with the new data. Discoverability requires the ability to recall related historical data so that an arriving piece of data can find its place, similar to the way each jigsaw puzzle piece is assessed relative to a work-in-progress puzzle. Each new puzzle piece incrementally builds upon what is knowable, at each given point in time relative to the evolving puzzle picture. Often new pieces, although important to building out the bigger picture, do not themselves bring new critical information. (On the other hand, some pieces may change the shape of the puzzle in a way that warrants ringing the bell—finding that one piece that connects the palm tree scene to the alligator scene.) It is at this moment in time, when the new puzzle piece presents the opportunity to reshape the picture, that discoveries are made. Real-time discovery replaces the need for users to think up and pose the right question at just the right time.

Organizations that are unable to switch to the “data finds data” paradigm will be less competitive and less effective.

The Benefits of Just-in-Time Discovery

Advanced information management systems that play this “data finds data” game will not rely on users to dream up the correct, relevant, timely questions to ask computers. While this technology will initiate new policy debates, such as which data will be permitted to find which data and who is notified of what relevance, here are some examples of what a “data finds data” system can do:

Guest convenience

After tossing and turning in bed all night in a hotel room, the guest finally decides at 7 a.m. to call for a late checkout and schedule a wake-up call at noon. Shortly after the guest sinks into a deep sleep, disaster strikes when the maid carelessly knocks on the door to clean the room. Regrettably for hotel travelers worldwide, this most basic inconvenience occurs all too often. When the data finds the data, the late checkout and wake-up call requests converge with maid scheduling information. This “data finds the data” instance would trigger an automatic text message, notifying the maid *not* to clean this room until after 2 p.m.

Customer service

With interest in a soon-to-be-released book, a user searches Amazon for the title, but to no avail. The user decides to check every month until the book is released. Unfortunately for the user, the next time he checks, he finds that the book is not only sold out but now on back order, awaiting a second printing. When the data finds the data, the moment this book is available, this data point will discover the user’s original query and automatically email the user about the book’s availability.

Improved child safety

A parent wants to ensure that her young children are safe while walking to school. The parent might search the community website to ensure that no registered sex offenders

are living along her children’s walking route to school. Will the parent check this website every day, to determine whether a new address of an offender is added to a street on the route? Using the “data finds the data” paradigm, should a new sex offender register an address on the children’s walking route, the new data will immediately connect with the earlier query. The parent will be instantly notified of the relevant discovery.

Cross-compartment exploitation

The government uses “compartments” to intentionally isolate data. Isolating data helps prevent highly sensitive data from escaping. Despite new mandates for information sharing, the traditional data protection practices for highly classified data prevent the government from discovering that two such compartments are dealing with the same subject or have subject overlap. An example of this might be one unit that is working on counterterrorism and another on counternarcotics. The government has hundreds of compartments, and the practicality of locating relevant data in another compartment is remote, because one never knows who has what information. When the data finds the data, the moment a record is added to the counternarcotics database of relevance to the counterterrorism unit (e.g., data involving the same person), notification is immediately published to the appropriate user.

Corruption at the Roulette Wheel

This is a true story where bad-guy data finds good-guy data—causing an unexpected discovery and resulting in a surprise outcome.

In the mid-1990s, riverboat gaming became legalized in many new jurisdictions, Louisiana being one of them. One of the challenges of a new gaming jurisdiction is the lack of available local employee candidates with deep gaming experience. New jurisdictions must therefore train the local workforce in a wide range of specialty job categories, ranging from dealers to surveillance room operators. Bossier City, Louisiana is one such community that had to make the transition from no casino business to casino riverboat operations.

Today is like any other day in any other casino. The dealers are watching the players. The floor supervisors are watching the dealers and the players. The casino manager is watching the floor supervisors, the dealers, and the players. And the surveillance room is watching everyone—even the casino executives. The surveillance room has an obligation to watch gaming transactions, not only to protect the house but also to protect the customers. Surveillance focuses both on gambling transactions as well as evidence of other criminal activity. For example, a purse-snatcher is bad for business because he interferes with the customer experience.

Surveillance is the last line of defense.

Hundreds of cameras (thousands in the Vegas mega-casinos) are piped into a remarkably tiny surveillance room. Twenty, thirty, maybe forty monitors cover an entire wall like a scene out of *CSI*. So how is it that only a few operators cleared for access to this room make sense of this information overload? Answer: *tripwires* and *attentive browsing*.

Tripwires come in many forms, ranging from a tip on a hotline to a floor supervisor asking for the surveillance room to evaluate the play of a customer (e.g., to determine whether she is counting cards).^{*} On this day, the surveillance operator is browsing—performing what might be called a random audit—zeroing in on one table after another, watching a short while to see if anything seems out of place, and then moving on.

Wait! What is that? Can't be. The attentive surveillance operator has just observed a player blatantly cheating on the roulette table. Today the observed scam is known as "past posting"—a player who is placing bets after the roulette ball has already landed in its number. Past posting occurs when the player notices the ball has landed on a number (e.g., 32) and seeing this outcome, quickly places a late, sure-to-win bet on that number. Unlike card counting, which is legal but discouraged, "past posting" is actually cheating and is definitely illegal.

What is so peculiar today is that the table has one dealer and one player. Usually the "past posting" scam involves a team of players—players working together to prevent the dealer from detecting the late bet. Such team activity might involve two or more "players" at the table who appear completely unrelated (e.g., acting like they don't know each other). After the ball finds its number, one player (e.g., a nice lady with a grandma disposition) will reach all the way across the board as if to place a bet. The dealer, of course, says, "Madam, it is too late to place your bet." Nonetheless, the reach across the table (toward a losing number) has enabled the other team member (say, a punk-rock-looking dude with a mohawk) to place a late bet on the winning number—hidden from the view of the dealer.

There is no such team on the table today. This is curious and warrants immediate inspection. The tapes are rewind. There it is! The dealer seems a bit distracted and misses the past post event. At this point it would be common to start looking backward in time by reviewing the earlier moments of the game. (And at the same time, maybe another room operator will begin watching the game go forward live.) Bang! Again and again the player makes late bets, the careless dealer missing this each time—and each time, paying the customer for the win.

Security is notified. The troops mobilize. The player is confronted and detained for arrest by law enforcement. How could this happen?

The dealer is obviously quite embarrassed. Being in a new jurisdiction, this employee is also new to gaming and really only has had classroom training. The dealer makes it clear that she has only heard about this type of scam. She then points out that security has really done a good job today. She is very embarrassed and is quick to guarantee it will never happen again on her watch.

Enter "data finds data."

^{*} Notably, the role of casino surveillance is intelligence. They report their observations and findings, but perform no enforcement; enforcement is the role of the security department.

When the cheater is apprehended, he is required to identify himself. The cheater today, like any other day, presents his name, address, phone number, and some other information. This information is collected on a standard form, with a signature if they can get one, and thereafter is data-entered into the corporate security arrest processing systems.

Of course, the cheater's last name is not the same as the dealer's; that would make life too simple. The address is also not that of the dealer. The phone number, however, happens to be the same as the phone number that the dealer used on her employment application! At this exact moment that the arrest information is presented to the arrest processing system, a secondary system performing data finds data* makes this vital discovery and produces an immediate alert that basically says, "Employee #5764 has the same phone number as Arrestee #44-00321!"

Long story short, the dealer is confronted, she confesses, and they are both processed and handed over to law enforcement for prosecution.

To be clear: the users did not take identity attributes (e.g., name, address, phone number) of the "past posting" cheater and attempt to search the wide array of operational systems (call this a federated search). The applicant, employee, loyalty club, and arrest processing systems, among others, cannot even be searched by address or phone number—they were not designed for that.

Using a "data finds the data" environment, the users do not have to proactively search or pose relevant questions. The users in the security department enter what they have learned into their system, and this new information is then assessed against other enterprise information assets. The new information is found to relate to existing data, and this relationship meets a prespecified condition of interest: when a "bad" guy is related to an employee. Because this condition is met, an alert is triggered because the bad guy and the employee share a phone number (thus, bad guy knows employee). If the cheater arrest data had perhaps found an association with a hotel visitor of three years ago, this noninteresting discovery would not have resulted in an alert.

To drive this point home, let's now imagine the phone number provided by the "past poster" had no relationship to that of the dealer. In this case, no alert would have been produced. The dealer may have been questioned with some suspicion, but there simply would not be enough evidence to make any claim. Might corporate security have opened an investigation of the dealer, or hired a private investigator to determine whether these two individuals were in fact close friends? Who knows?

But what if? What if the phone number does not match and no connection is made? The dealer continues to deal. Time marches on. What if, six months later, the dealer changes

* This technology, formerly known as Non-Obvious Relationship Awareness (NORA), was developed by Systems Research & Development (a company founded by Jeff Jonas) for the Las Vegas gaming industry. SRD has since been acquired by IBM and is now part of IBM's Entity Analytics group. Some additional information is available here: IEEE Paper: Threat & Fraud Intelligence – Las Vegas Style (http://jeffjonas.typepad.com/jeff_jonas/2006/11/ieee_paper_thre.html).

her home address in the employment system—and the new information is the same as the cheater’s address? How would the organization know this? In truth, no organization will ever know this unless it can play this important game called “the data finds the data.” The moment this new information connects the dealer with the closed case, such a system detects an alert condition: bad guy related to an employee.

Alerts, by the way, do not necessarily mean there is criminal activity. Alerts do, however, play an important role in focusing an organization’s finite investigatory resources—in this case, a condition of sufficient interest to warrant a closer (human) look.

Other examples: this riverboat casino operation also found a scam involving a marketing person who figured out a system hack to cash-back comp (a marketing activity whereby the more you play, the more points you get, and points can be redeemed for cash!) his roommate. And in another case, it discovered that the person pulling out the “car a day” winner ticket happened to “select” her sister (of a different last name), with the family members acting as if they had never met.

All three of these scams were revealed as the data became known using “data finds data”—and all three scams were detected in the first 90 days of operation!

The reason why data finds data is essential is that the order in which information arrives is uncertain. Systems and processes that take the order of events for granted have a fatal flaw: out-of-order facts may provide the organization with important knowledge that never gets acted upon. More about this later.

One large retailer with thousands of physical storefronts across the United States analyzed its historical data holdings and was shocked to discover that two out of every thousand employees had in fact already been arrested for stealing from them. Worse yet, these employees were caught stealing from the same store that hired them! Despite the order in which the data is presented, the moment the enterprise has such evidence there is no time to waste. The store should know this immediately.

Traditional remedies to address out-of-order data points are cumbersome. How can corporate security take advantage of new enterprise data that reveals an employee may be a known shoplifter? When should updates to an employee record in the human resource system cause corporate security to reevaluate all its earlier investigations? How can the reevaluation be structured so that the organization can’t miss instances when new or modified records in the internal investigations database are related to employees? One strategy is to periodically test the investigations database against the entire employee database. Another strategy is for corporate security to reinvestigate every employee on some recurring basis. But both of these strategies will miss important discoveries because timing is critical.

Even perfect algorithms running against perfectly reengineered operational systems (e.g., the human resources system and the internal investigations system) will still miss certain discoverable events. What if the data needed to determine that two people are the same or connected exists in an entirely unrelated system? For example, imagine a third record

arriving from an unrelated system, such as a loyalty club enrollment system, which reveals a previously unknown linkage between a home address and a home phone number. What kind of enterprise systems would be required to detect this condition, a condition we will characterize as a nonobvious relationship?

Data finds data, including nonobvious relationships, requires that one first solve the problem of “enterprise discoverability.”

Enterprise Discoverability

When new information arrives in the organization, whether that is a new employee, a change to an employee record, shoplifter information, or a loyalty club enrollment, one needs to know what other organizational data relates to this information.

One common approach to enterprise discovery involves a technique known as “federated search,” the passing of a query to every relevant operational system. As we shall demonstrate, this does not work for data finds data systems. Discoverability, especially within large-scale, real-time environments, necessitates directories.

Federated Search Ain’t All That

Organizations have numerous operational systems, each with its own dedicated business function, tailored information structure, analytics, and reports. Secondary aggregations of data are common, and include such things as data warehouses, operational data stores, and data marts. There are countless information silos, each particular to its mission or function.

Traditional federated search systems involve the user querying the database of each silo for relevant content. More sophisticated federated search systems use intelligent middleware to broker the individual queries to each database, a model where the middleware processes the query by engaging the myriad of information silos automatically and compiling the findings, returning the collective results to the inquirer.

Federated search assembles cross-silo data “just-in-time,” at the point information is needed. Although this type of federated search is applicable in some settings, it is not well suited to high-performance enterprise discoverability, which is required to deliver on data finds data.

There are *two* primary reasons federated search does not scale:

- Existing systems generally do not have the indexes necessary to enable the efficient location of a record. Payroll systems, for example, will often have prebuilt indexes (defined pointers into the data) to facilitate searches on employee number, tax ID number, and name. Rarely would a payroll system have an efficient way to locate records on address or phone number. Suppose our newly discovered gambling cheat discloses his address and phone number. Our payroll system keeps track of all of the employees’ phone numbers and addresses. (It has to send out paychecks, after all.) This same payroll system would not be able to easily generate (if it can at all) a list of any

employees who share an address or phone number with our previously mentioned gambling cheat. Even if an index were created on employee phone numbers within the payroll system, this still would not allow one to locate the emergency contact phone number in this same system. If we can't compare the gambling cheat's identification data against all relevant employee data, we miss the discovery of the *connection* between employee and cheat.

If a field is not in an index, the method for locating a record in a database is known as a "table scan." In a table scan, the value being searched for is compared against every single row in the table—the first record in the database, then the second, and so on. Therefore, the larger the database, the longer each search takes, and the greater the computational burden placed on the host system.

- To make matters worse, federated search requires recursive processing (some conditions necessitate repeating steps), which is a nightmare for distributed query environments. Suppose you perform a federated query to discover enterprise records related to a specific person—say, starting with a specific person's name and date of birth. If the federated query returns some new attributes for this person, e.g., a few addresses and phone numbers, then you have learned something new. To be thorough, one should leverage the new data learned about this person, i.e., initiate another enterprise-wide federated query in case there are additional records that can now be located based on these new data points. Now, what if this second federated query discovers another address, a few more ways to spell the name, and an alias or two? To be thorough, each time something is learned that might enable the discovery of a previously missed record, the discovery process must perform another enterprise-wide federated query.

Here is a real use case that underscores this point. An organization (a commercial entity, not a government) had 2,000 databases under its control. User queries were directed across these databases to gather related records. Brilliant middleware designed to optimize the search process was created over many years and at the cost of millions of dollars. This very smart system would know which databases could process which queries, determine the ideal order of database access, simulcast queries to all relevant sets, and assemble what has been learned. However, no amount of engineering on this federated search approach could overcome one serious design flaw: every time something was discovered (e.g., an alias or a new phone number), the brilliant middleware had to reissue the queries to many databases. This recursive process, being run on very large computers, eventually had to be programmed to stop processing at eight minutes! Note that the next recursive inquiry might have finally revealed an essential record.

But wait, eight minutes! This means a human or a system is now standing by, unable to act in the moment, as no answer is available for all these minutes. However, when data finds data, *the data is the question*. This now means every piece of new data arriving may see up to eight minutes of latency before processing the next piece of data. Imagine how impractical it would be to use a federated approach at the scale of hundreds or thousands of federated queries a second (real transactional volumes) submitted across the enterprise network, bouncing around and executing recursively through countless operational systems!

If these mentioned factors are not compelling enough, the death blow to federated search is that all the systems that must be searched have to be physically switched on and available, not undergoing maintenance or backups, or in the middle of periodic nightly or month-end batch processes. Connectivity must of course be fully operational as well. Contemplate these mandates in conjunction with an organization composed of hundreds or thousands of systems spread across buildings, time zones, and continents.

Federated search cannot support the “data finds data” mission, because it has no ability to deliver on enterprise discoverability at scale.

Directories: Priceless

Think about a library. Think of the library’s floors, hallways, and shelves as silos of information. Valuable information is tucked away, waiting to be discovered. No one roams the halls when looking for a specific book. Instead, one uses the card catalog—cross-referenced on subject, title, and author—to facilitate discovery of relevant documents.

Let’s say that directories, indices, and catalogs are all basically the same thing: a thing used to locate other things. Some examples of locators include the card catalog at the library, phone directories, Google, eBay, and so on. In each case, the directories are equivalent to locator services: they return reference information (pointers) after being provided one or more search terms. At the library, the card catalog is a special-purpose directory used to enable efficient enterprise discovery, providing the user a specific pointer to a document (e.g., using the Dewey Decimal system). After the user is provided a pointer, the activity becomes “federated fetch.” Note the difference between federated *search* (not useful) and federated *fetch* (useful).

A Google search does not scour the planet for the results; rather, a specialty directory created by Google is searched and the results, pointers to the real documents (e.g., URLs) are returned to the inquirer.

It would follow that the *only* scalable solution to enterprise-wide discoverability involves the use of directories. No surprise there; a special-purpose directory is therefore a fundamental component that permits “data finds data” systems to handle discoverability at scale and determine relevance in real time.

All directories are not created equal. There is a big difference between traditional “context-less” directories versus directories capable of accumulating and persisting context. Contextualized directories enable data to find data in remarkably unexpected (nonobvious) ways, in real time, at great volume, and with extraordinary efficiency.

Context-less directories are the most common type of directory: each document is indexed indifferent to all other documents. In other words, new enterprise transactions (documents) update the directory without any attention to how this transaction (metadata for the index) might relate to any other transaction. Context-less directories are designed to provide users with the most basic ability to locate documents (e.g., all books related to “Billy the Kid”).

*Semantically reconciled** directories are directories that attempt to exploit synonyms, things that use different words to mean that same things. This means users looking for one thing (e.g., “Billy the Kid”) should find other “same” things (e.g., “William Antrim,” one of his aliases). Semantically reconciled directories recognize when a newly reported entity references a previously observed entity. Directories that contain semantically reconciled data can be thought of much like a library card file, with one big difference: cards relating to like entities are rubber-banded together. This means if a search locates one card, as a bonus, all other related cards are discovered without any additional effort. Most notably, some of the cards in the rubber-banded clump of library cards may not even contain the original data item being searched.

Quite frankly, this can look like magic. When attempting to discover what the enterprise knows about an email address, one can discover a record with the email address as well as other records in the enterprise about the same person—for example, loyalty club activity, despite the fact that the loyalty club record never contained an email address. Algorithms that semantically reconcile identities (for example, people or organizations) are sometimes referred to as *identity resolution*.[†] Algorithms that determine when multiple entities are in fact the same entity require a deeply nuanced discussion, and as such are beyond the scope of this chapter.

Semantically reconciled and relationship-aware directories[‡] are a type of directory that provide an even higher degree of context by allowing users to discover additional documents, such as those related by intimate association (e.g., Billy the Kid aka William Antrim). It may also be important to understand there was a real William Antrim, who happened to be Billy the Kid’s stepfather. The way to visualize this is to picture in your mind’s eye a library card catalog with some cards already bundled in rubber bands as described earlier, plus threads that connect some bundles to other cards and other bundles. One can search any single set of terms and locate a bundle, and at that instant learn how that bundle is associated (related) to other bundles. Some association threads are thicker than others, indicating a stronger association. By following a thread to another bundle, one can then instantly see what threads lead from the next bundle. In this manner one can observe degrees of separation, as in the six degrees of Kevin Bacon.

When contemplating “data finds data” systems, keep in mind that the moment a new transaction is placed in context, possibly adding to the context of an existing entity, there is the potential that the new information may change the shape of the picture as the bundles and the threads reorganize.

* Defined as “Recognizing when two objects are the same despite having been described differently.”

† Don’t confuse identity resolution with a quasi-related set of methods sometimes referred to as match/merge or merge/purge. More about the distinction here: “Entity Resolution Systems vs. Match Merge/Merge Purge/List De-duplication Systems” (http://jeffjonas.typepad.com/jeff_jonas/2007/09/entity-resoluti.html).

‡ The NORA (Non-Obvious Relationship Awareness) is an example of a semantically reconciled and relationship-aware directory.

Persistent context is the term used to refer to the current state of the reconciled and relationship-aware index—essentially, the present status of the ever-changing and incrementally improving puzzle. Persistent context is the memory of how things relate, and it trumps the just-in-time context that is delivered in federated search systems.

Persistent context (semantically reconciled and relationship-aware directories) enables high-performance discovery, streaming contextualization, and the opportunity to detect relevance in real time. It is also worth noting that the detection of relevance is computationally cheapest if it can be assessed at the moment the data ingestion is taking place. In this respect, the librarian (the function that stitches the new data into the directory) is the first to notice if arriving data is of enough relevance to be published (e.g., to a user).

Sense-making on streams beats boiling the ocean. A clear way to envision this is imagining an organization with 4 exabytes of historical data that receives 5 terabytes a minute. Do you think they run a process over the weekend to reveal what has been learned? There may not be enough computers or energy on Earth to do this. (Note: Envisioned behavior at scale proves to be very helpful when designing highly efficient systems.)

Relevance: What Matters and to Whom?

Now we have lots of data coming at us. Take a moment to assess all the sights and sounds in your immediate environment: the background hum of the computer, the music playing, the cartoons on the TV, and the kids trying to steal one another's toys. As humans, we have internal relevance-detection algorithms that assess all of the presented data, and alert us when some observations cause us to take notice and possibly react. Sort of like ignoring the kids and their ongoing bickering until it threatens to escalate into violence. It's that one move, by one kid, that alerts us that it is time to intercede.

How one exploits technology to calculate relevance and when to allow it to automatically register an alarm is crucial. On the technology side, the objective is a state in which the very next item in an alarm queue is the next most important item for review. There is also no reason to produce more alarms than there are available resources (for example, analysts, systems) to deal with them. Risk-assessment engines, for example, must be configured to produce alarms appropriate to one's individualized risk, staffing, and ability to respond. If resources increase, one can increase alarm sensitivity.

Components and Special Considerations

Our intelligent “data finds data” environment must contain eight essential building blocks:

- The existence of, and availability of, observations
- The ability to extract and classify features from the observations
- The ability to efficiently discover related historical context
- The ability to make assertions (same or related) about new observations
- The ability to recognize when new observations reverse earlier assertions

- The ability to accumulate and persist this asserted context
- The ability to recognize the formation of relevance/insight
- The ability to notify the appropriate entity of such insight

The Existence of, and Availability of, Observations

If there is no data, then there is no chance one can make sense of it. And if there is data, it has to be “sensed” (collected) by some sensor system for it to ever be of potential use. And even if data is collected, one must have access to it to have any hope of making sense of it.

The Ability to Extract and Classify Features from the Observations

For the sake of argument, let’s say a grain of sand contains too few features to extract and classify. Grains of sand are frequently the same color, size, weight, shape, and so on. Therefore, the lack of discriminating features would prevent one from identifying the same piece (semantic reconciliation) later. The point being, for data to be placed into context, one must be able to extract and classify its key features. Structured data is rather easy when address information is contained in one column and first and last name are in another. Unstructured data, such as newspapers and blogs, take a lot more work; extracting the right names and addresses is a challenging field, often called *entity extraction*. Feature extraction from video, such as car license plate readers, can be done in certain cases.

Long story short, one must be able to extract and classify the key features of observations if one hopes to stitch observations together into context.

The Ability to Efficiently Discover Related Historical Context

As new observations arrive, the extracted and classified key features are used to look into the contextualized historical data (what we have called persistent context) to discover how this new piece fits. For this to occur in real time in support of high-volume data streams, this discovery requirement must be extremely fast.

The Ability to Make Assertions (Same or Related) About New Observations

When a new observation is contextualized with respect to the historical data holdings, the algorithms must make one of several assertions: (a) this is a new entity, the first of its kind (e.g., a new person); (b) this is a known entity (e.g., this is an observation about somebody we already know), in which case the new entity is “resolved” with the existing entity; or (c) how this entity (the new one or known one) now relates to other entities.

There comes a point, just the same as when people put jigsaw puzzles together, where one concludes that nothing more can be done. It is at this point that one abandons the current item wherever it has been placed and moves one’s attention to the next puzzle piece (observation). Note that you may have made the last assertion in error, but at this point it is unlikely you will ever discover this unless a new piece arrives to reveal this potentiality.

The Ability to Recognize When New Observations Reverse Earlier Assertions

Sometimes an observation contains new information that provides the evidence that an earlier assertion should be reversed. Possibly this new information proves that entities previously deemed not the same are in fact the same entity. Conversely, a new observation may reveal that two entities previously determined to be the same are now believed to not be the same at all (for instance, the new data point indicates that two Bob Joneses in our database are not the same person, and instead are a case of junior versus senior).

Using new observations to reverse earlier assertions is one of the most complicated aspects of semantic reconciliation algorithms. But without this important feature, databases drift from the truth over time. The disadvantage of this is that periodic database reloads are used to correct for this phenomenon. And for very large data sets, obviously this presents a scalability nightmare.

The Ability to Accumulate and Persist This Asserted Context

When the assertion process is complete—in other words, when new observations are reduced to assertions (new, same, or related entities) and new observations are used to remedy these same earlier assertions—the newly learned knowledge must be captured in a database so that the very next transaction can benefit from the new knowledge. In some respects, this begins to feel like a very basic incremental learning system.

The Ability to Recognize the Formation of Relevance/Insight

Only after a new observation is applied to the historical data such that no more computation is warranted, this is the moment the system must ask itself, “Have I learned something that matters?”, much in the same way a person will incrementally look to see what has been revealed after each puzzle piece is placed onto the board.

The work we have done involves the detection of prespecified patterns of interest. For example, it is relevant to discover whether the good guys know bad guys or if the cash transactions for one person exceed \$10,000 per day.

However, new relevance parameters can be set based on external processes, which might include human insight or secondary pattern-discovery/data-mining engines.

The Ability to Notify the Appropriate Entity of Such Insight

When insight is detected, who or what system should be notified? In our existing implementations this is trivial, as each relevance rule (for example, if a prospect is a close associate of one of my top 50 customers) and the dissemination rule (i.e., send a courtesy message about this to the casino host) is established at the same time.

Privacy Considerations

Smarter systems, like those capable of performing “data finds data,” require very close attention to privacy and civil liberties protections. How these next-generation systems are built and deployed, and what policies (including accountability and oversight) govern their use deserves close attention and vigorous debate. Some of the core issues include: defining what data should be indexed for discoverability, how the data will be stitched together (e.g., what constitutes a relationship?), what constitutes relevance, what relevance is disclosed to whom, who can search the index, how the system will be monitored for unauthorized use, and how errors will be detected and corrected.

Fortunately, the directory-based model has a number of nice privacy-enhancing characteristics, including:

- Urges to share more data with more parties are replaced by transferring less information to fewer places (card catalogs).
- Who searches for what and what they found can be logged (for instance, using tamper-resistant logs) in a consistent manner, thus facilitating better accountability and oversight.*
- Information sharing between parties is now reduced to just the records that they need to know and to share (sharing less by sharing only the information that must be shared).
- It is now possible to make the index anonymized, which means the risk of unintended disclosure of even the limited metadata in the index is drastically reduced.†

Conclusion

“Data finds data” systems determine how new observations relate to what is known and detect relevance/insight worth special attention. Such insight ideally occurs in real time, thus enabling real-time reaction.

This emerging technology will ultimately differentiate one organization from another across a spectrum of enterprise interests, ranging from better recognition of opportunity to better estimation of risk.

Data finds data will likely become yet another building block from which next generations of advanced analytics will benefit. This new paradigm will inevitably lead to the emergence of even smarter systems, potentially even contributing to advances in cognitive computing.

* Tamper-resistant logs are also often called *immutable audit logs*. An interesting paper published by the Markle Foundation on the subject as related to national security, especially in relation to nontransparent government systems, is located here: http://www.markle.org/downloadable_assets/nstf_IAL_020906.pdf.

† More about anonymized directories can be found in a chapter entitled “Anonymized Semantic Directories. A Privacy-Enhancing Architecture for Enterprise Discovery,” by Jeff Jonas and John Karat, in a book entitled *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*. Robert L. Popp (editor), John Yen (editor), published in 2006 by Wiley-IEEE Press (<http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471776157.html>).