STEWARDSHIP AND STAGING OF INFORMATION FOR ENTERPRISE MINING

(Research-in-Progress)

Jon R. Wright
AT&T Labs - Research
jrw@research.att.com

Gregg T. Vesonder
AT&T Labs - Research
gtv@research.att.com

Tamraparni Dasu
AT&T Labs - Research
tamr@research.att.com

Abstract: In an enterprise setting, a major challenge for any data mining operation is managing the information, be it data or metadata, to ensure a stable and certifiably accurate flow of data. The data feeds are complex, numerous and often quite opaque. The management of frequently changing data, metadata and other information feeds presents a considerable challenge. In this paper, we articulate the technical issues involved in the task of managing enterprise data and propose a multi-disciplinary solution derived from fields such as knowledge engineering and statistics, to understand, standardize and automate information quality management for enterprise mining.

Key Words: Data Quality, Data Mining, Feed Management, Knowledge Engineering

1 INTRODUCTION

A substantial amount of effort in industrial applications is spent on assembling, preparing, managing and interpreting data and its metadata. Current work on data mining focuses on extracting relations from data in preexisting databases or repositories. Much of the research pre-supposes a collection of databases that are available to the community, such as the World Wide Telescope (Gray, 2002) [4]. In most industrial applications, the data is not neatly accessible and its initial form is frequently not amenable to data mining. In addition, since the data, the metadata and the knowledge needed for interpretation are constantly changing, data mining and related activities are a continuous process. Finally, unlike many traditional mining projects, the work often occurs in an environment that differs from the original "home" of the data, because mining activities cannot be allowed to interfere with the production environment.

The TDQM program at MIT has addressed the issue of data quality and information management in a series of publications including Data Quality Assessment by Pipino, Lee, and Wang [9] and Quality Information and Knowledge Management by Huang, Lee and Wang [6]. In this paper, we focus on tools and methodologies to implement information quality management techniques such as those discussed in these publications in the context of enterprise information quality management, from data gathering to data mining.

We call the process associated with assembling, preparing, managing and interpreting the data for the data mining process, *mise en place*, a cooking term, meaning "put in place," which describes the activities of measuring, chopping, peeling, etc., in preparation for the actual cooking process. In a similar fashion, enterprise data must be prepared for data mining. Such activities include a) assembling time-dependent metadata, b) providing and managing a long-term repository for ephemeral data and metadata, so that longitudinal trends may be deduced and, c) monitoring and improving the quality of the data. In many industrial settings the bulk of the effort for a data mining project focuses on these *mise en place* activities. Without these activities, meaningful data mining would be impossible. Data preparation for data mining has been addressed by Pyle in [10].

Our goals are a) to develop awareness for the important role these activities play, and b) to standardize and automate preparation activities through supporting tools, methodology, and techniques, so that less effort is spent on preparation and more on data mining. Key techniques we are using to understand, standardize and automate this process are derived both from artificial intelligence and statistics, and include: knowledge engineering, planning, constraint management and rule based programming.

The remainder of this paper: describes the differences between traditional and industrial data mining, highlights the industrial issues, discusses the technology used and presents a case study of our approach, focusing on data assembly and information quality management issues.

2 PREPARATION OF INFORMATION FOR DATA MINING

The World Wide Telescope project (Gray, 2002) [4] is typical of many research oriented data mining projects. As Gray reports, in projects such as this, the data is well documented, software exists to process the data and federate with other archives, and the database plumbing and metadata management are in place. Indeed, Gray remarks that gathering the astronomical data and producing the catalogs comprises 75% of the effort. It is only then that data mining can begin.

In most industrial projects rarely does such preparation exist. Since the data that support different aspects of the business (e.g., billing, provisioning, and contracts) are scattered across the enterprise, the data mining team must tap into feeds of data from appropriate enterprise data streams and assemble that data and relevant metadata in a local resource that can be devoted to data mining. Furthermore, if longitudinal mining is necessary, the project must arrange for data archives, since they likely do not exist on the enterprise site. Only after all these elements are in place can the industrial team begin the process of constructing software and documentation to support the data mining process. These ongoing *mise en place* activities consume the majority of the effort in most industrial data mining projects and may be the most important activity to engineer and automate. Our goal is to devise and promote techniques that reduce the time and effort for these activities, transforming the “one-of’s” to repeatable, automated processes and abstract general techniques**.**

3 THE *MISE EN PLACE* PROBLEM IN INDUSTRIAL DATA MINING APPLICATIONS

As mentioned, traditional data mining techniques assume that a data matrix has been prepared and put in place. Frequently, the matrix includes attributes whose values change over time, but the data itself is considered to be *terminal* in the sense that it has reached maturity and stabilized in terms of the attributes and inter-relationships. The underlying structure characterized by statistical distributions could be arbitrarily complex but any changes over time are assumed to be minor and captured by tuning the model parameters.

The problem is much different in industry. Businesses change, products change, technologies that support the operations change. In response, the data feeds themselves are in a state of constant change. Therefore, given multiple data feeds, each consisting of many files, with incomplete data description and unknown data flaws, we must develop a framework that: (1) insures that all the relevant data and metadata have arrived at the data mining center, (2) transmits the data and metadata to appropriate machine resources for analysis, (3) assembles the data and metadata on those resources, (4) checks relevant constraints between the data for a given analysis run and (5) archives the data appropriate to its phase in the life cycle. We consider this to be a significant part of the metaphorical iceberg (Figure 1), the tip of which corresponds to conventional data mining activities.

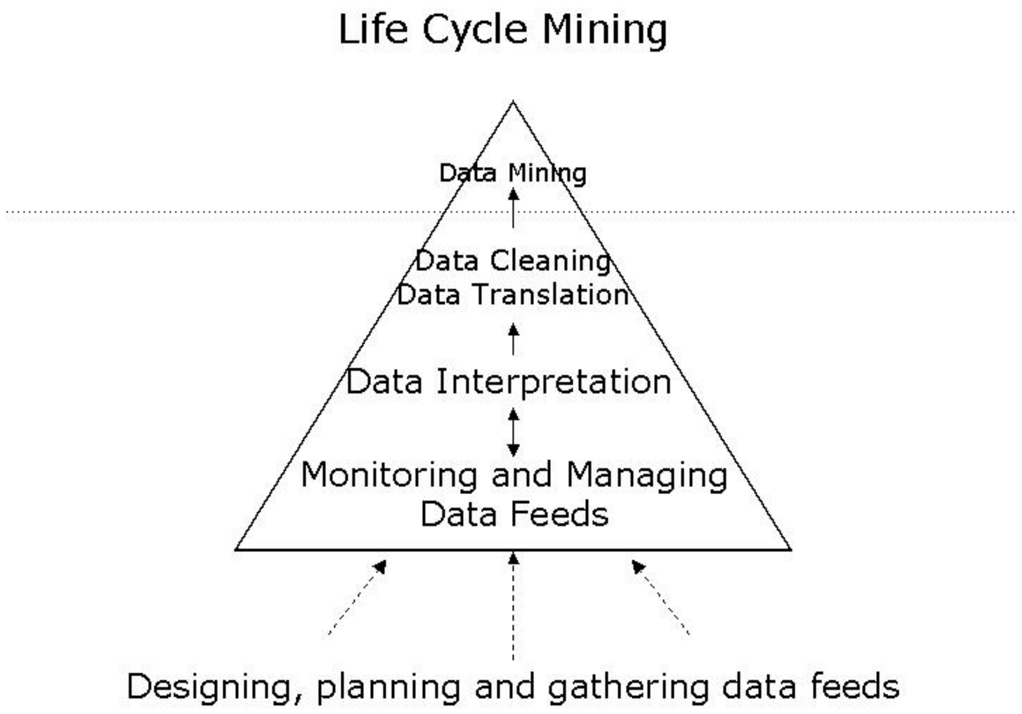

Life Cycle Mining
Data Mining
Data Cleaning
Data Translation
Data Interpretation
Monitoring and Managing Data Feeds
Designing, planning and gathering data feeds

FIGURE 1

3.1 Data Assembly

Data miners who analyze data during all stages of its life cycle have the daunting challenge of interpreting data streams that are very minimally documented at best. Assumptions of a reasonable schema and well-behaved data are unrealistic. Hundreds of data feeds with complex semantics and attributes arrive continuously (Figure 2). They have inter-relationships that have to be inferred because at times even the systems that create them are unaware of these inter-relationships, as they see only a small part of the overarching process that produces and interprets the data. Synchronizing feeds is always an issue. To complicate matters further, data feeds are corrupted or broken and there may only be a small window of time (hours) to recover before losing the data forever. We will see concrete examples in the Case Study section.

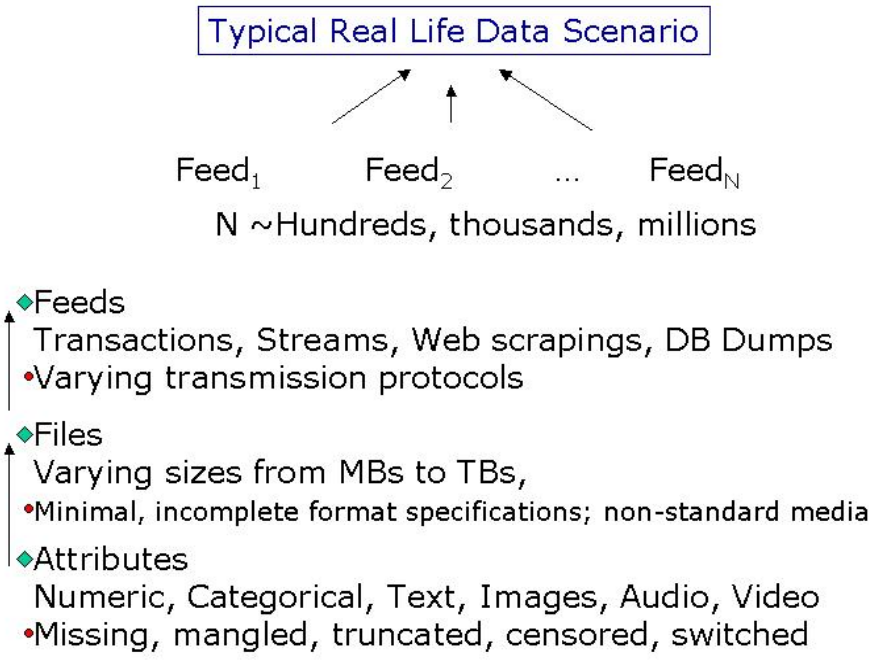

Typical Real Life Data Scenario
Feed1
Feed2
...
FeedN
N ~Hundreds, thousands, millions
Feeds
Transactions, Streams, Web scrapings, DB Dumps
Varying transmission protocols
Files
Varying sizes from MBs to TBs,
Minimal, incomplete format specifications; non-standard media
Attributes
Numeric, Categorical, Text, Images, Audio, Video
Missing, mangled, truncated, censored, switched

FIGURE 2

3.2 Information Management: Archiving

Because the data mining project is often the only long-term repository for enterprise data, loss of any data whatsoever is not acceptable. There are sometimes legal requirements for retaining data, especially in the case of regulated service industries. Stewardship of the data, therefore, is actually an essential aspect of enterprise data mining activities. Because all media can be expected to fail at a certain rate, the chief tool for protecting against loss is replication.

A serious project needs to have replication rules and a process that implements them. For example, one set of rules might be, for all data coming within the data mining umbrella, at all times: (1) if no replications are on tape, three replications must be on hard disk. (2) If 1 replication is on tape, then at least two replications must be on hard disk. (3) If 2 replications on tape, then hard disk replications may be removed. Notice that the rules imply a database that keeps track of the state of each replicated entity.

The scale at which enterprise data is created and the complex system infrastructure on which it resides, transforms this into a significant and challenging problem. We have experience with implementing, and executing on requirements for archiving and safeguarding information arriving at a rate of between 35 and 70 terabytes per year. When the scale of coverage is, say, two million files and 35 terabytes of data, there is no choice but to turn to an automated process to make even these simple rules practical. Later will we discuss an automation technology for implementing archiving rules such as the above.

4 APPROACHES

A key task in knowledge engineering the automation of data and metadata management is managing constraints. Interdependences in feeds of data and metadata, their interpretation, as well as synchronizing and ordering data management tasks, are often expressed by subject matter experts as constraints. For example, “If an item is sold in NJ, send it to Biller #3” or “If a sale is less than $0.20, drop it altogether”. Many data quality metrics too are articulated as constraints, such as “If a data record needs manual intervention in order to complete the process successfully, then it contains bad data”. Managing these constraints is accomplished within the technology context of rule-based programming and planning. ” Please see (Dasu et al, 2003 [2]) for a motivation of the relationship between data interpretation, constraints and rule based programming.

4.1 Constraints Represented as Rules

There are many approaches for managing constraints related to data interpretation. These range from ad hoc, informal techniques to formal ones like the constraints enforced via database triggers. No single solution offers all the answers.

Embedding the constraints in a source code that “prepares” the data for data mining activities is a common approach. Often, this results in a single monolithic body of code that integrates, prepares and analyzes the data. This is particularly true when the data are not resident in a formal database. While this approach is effective for quick and easy projects, it is inappropriate for projects that span several data warehouses and/or have strong temporal aspects. The code becomes complex and opaque. Making changes can result in unpredictable outcomes, as it is difficult to check the impact of the changes and their interaction with the existing code. Retrieving metadata for newcomers or for non-technical experts is next to impossible.

The flexibility of rule based programming, the separation of knowledge and control, and the ability to trace the effect of the rules, make it a nice platform for incorporating and managing constraints. Furthermore, the rules are easily understood by experts who articulate them, so that validating them is easy. This issue is particularly important while resolving conflicting metadata from multiple experts.

In addition to rules, a part of managing metadata and constraints involves incorporating analysis, such as statistical tests for detecting changes in distributions. Such tests can be simple such as univariate tests for a known distribution or can be more complex, seeking to capture multivariate joint distributions. Providing this functionality is an important part of the *mise en place* process. We demonstrate this in the “alerts” discussed in the case study section.

4.2 Constraints Represented as Plans

Recent research in AI has amply demonstrated the connection between planning and constraint satisfaction (Weld, 1999 [12]). In addition, there have been important developments in action monitoring and dynamic replanning when failures occur. This research has been applied to some important practical problems such as the Mars lander robot, and we think it all may be coming together to provide some powerful automation tools for the *mise en place* problem in data mining.

A plan is simply a sequence of actions that, if executed, will achieve some goal. Data mining related actions, for example, might include file transfer, format conversion, cataloguing, as well as execution of the analysis and reporting of the analysis itself.

Intuitively, we think that plans could serve as the core structure around which many of the activities associated with data mining could be automated. Aside from the guidance it provides with respect to task execution, it provides structure for monitoring, diagnosing, and even resolving problems that occur during plan execution. The inherent understandability of a plan by humans is an important feature supporting the Knowledge Engineering process. A plan may also serve as a benchmark against which human operators can monitor and judge progress. It could allow precise and accurate status descriptions stated at an appropriate level.

Much more is now known about how to monitor the execution of a plan and how to replan in the face of failure. Even if replanning is not possible or not successful, information gained through automatic replanning could be used to compose alerts for human operators, who might then resolve conflicts and deadlocks.

Finally, for practical applications, detailed plans are needed to accurately predict completion dates and times. It is easy to underestimate the importance of this element unless one has experience with real clients.

One key development in planning has been the dramatic increase in the size of problems that planning algorithms have been able to solve. This has been driven to a significant extent by progress in constraint satisfaction and search technology. In addition, work in execution monitoring and dynamic replanning (Hammond, 1990)[5]) seems to be quite important to our class of applications.

Much of the progress has come about because of the recognition that there is a close connection between planning and constraint satisfaction. Planning requires three things: (1) Description of the **starting state** (2) Description of **the goal** to be achieved and (3) Description of the possible **actions** that can be performed. The set of possible actions is sometimes called a Domain Theory. Descriptions of states, actions and goals are typically expressed in some formal language. Classical planning is based on the STRIPS representation (Fikes and Nilsson, 1971)[3]. In this representation, states are represented by conjunctions of predicates applied to constant terms. Predicates are allowed to be negated. Actions are simply state descriptions with pre-action (precondition) and post-action (effect) clauses. STRIPS representation limits the size of the problems planners can deal with, partly because the use of constants causes the search space to explode. In addition, it lacks the expressiveness needed to implement replanning. However we have found the straightforward STRIPS implementation suitable for our current needs and since it is the basis for modern planning approaches, it provides the flexibility to move to them if necessary.

5 CASE STUDY

In this section, we will describe in broad terms the class of real-world enterprise data mining projects that illustrate the preceding discussion and focus on the *mise en place* activities of data assembly and archiving. Most commonly, data originates within legacy systems that control and support major business processes. The legacy systems execute a large number of transactions, perhaps billions per month, and data records are created with every transaction. It is very important not to impact the ability of these systems to perform their control and support functions. Periodically, perhaps once an hour, transaction records are extracted from the legacy systems and written into files. These files are transmitted for auditing as they are written. There is a narrow window within which a file can be retransmitted if there is a transmission failure or the file is somehow corrupted after a successful transmission. Otherwise, the file and its data are lost, since they are routed to other enterprise systems for parsing, translation and summarization. The quality of this data is very important, because it is the primary means of understanding and analyzing the associated business processes. Strict reporting requirements permit no data ever to be lost. At the very least, lost data at any level causes major concern.

Analysis of these data consists of selecting an appropriate bundle of files, transferring them to an execution platform, performing the analysis by running programs over each file and accumulating the results. While human resources could be applied to the problem, even the best human operators introduce errors and mistakes into the process due to the scale of the task (we are all human, after all). Automation of the process is desirable both from a cost and a quality perspective.

There is continuous, ongoing activity involving transferring, receiving, and cataloguing the incoming files -- over the long haul, millions of files will be received, catalogued, processed and analyzed, and finally archived. Processing also may involve several transformations, for example, compressing or translating from EBCDIC to ASCII format. Header information is extracted, interpreted and saved as part the cataloguing process.

The first step in performing an analysis is bundling an appropriate set of files -- thousands of files could be processed in a single run. The files that are bundled are subject to various correctness and quality checks. Missing files must be located and restored, if possible. Next, the files must be transferred to an appropriate execution platform where an analysis program is applied to each file producing a summary of its contents. The program often requires auxiliary files, for example, tables that are date-sensitive, perhaps valid only for a specific time period. Practically speaking, the analysis routines are often very simple, largely consisting of record filtering and counting. In some cases, matching records arising from various legacy systems is involved.

Errors can be encountered on any file. Each one must be corrected before the analysis routines can be resumed. The individual file summaries must be combined into an overall report, and there is a validation exercise on the overall results. Occasionally, reruns of the process are required.

We can tell, even from our high-level description, that these sub-problems can be linearized, which simplifies the planning problem considerably. Each of these can be attacked separately, and we may find that an eclectic approach, allowing the nature of each sub-problem to dictate an approach, works the best. Finally the files must be archived for future analysis and potential longitudinal studies. The disposition of the data is constrained by the phase in the lifecycle and will be discussed further in section 5.2.

5.1 Description of Application

In this section, we present a simplified version of the real life application. We are deliberately vague on specifics for proprietary reasons. A big corporation wants to be able to account for every unit of service it provides to its customers and to bill the customers accurately. The measurement of the service units is transmitted in a standard record layout of considerable semantic complexity. The record is then parsed and retransmitted to downstream applications in simpler, application-specific chunks. For example, a billing system will receive a re-formatted record with just the relevant information it needs.

The feeds are received and staged on a Unix cluster where various enterprise applications (for example, revenue assurance) can have access to a persistent and certifiably healthy set of data during its entire life cycle. The details of the Unix cluster and its architecture are discussed in Hume and Daniels [7]. A simplified diagram of the cluster is represented in Figure 3. There are three nodes in the cluster. At any point in time, one of the nodes is the leader. Applications and requests are submitted to the leader using a high-level interface language. The cluster level job scheduler sends the jobs off to the appropriate node, where the node level job scheduler arranges for the execution of the job. The files are replicated on various clusters and written to tape to safeguard against accidental file loss. The replication manager is a database that keeps track of file replications and their locations. There are other details such as satellite clusters that act as gateways for receiving and vetting files that are not represented in Figure 3. In this case study, we focus on the methodology used to “stage” the data for life cycle data mining.

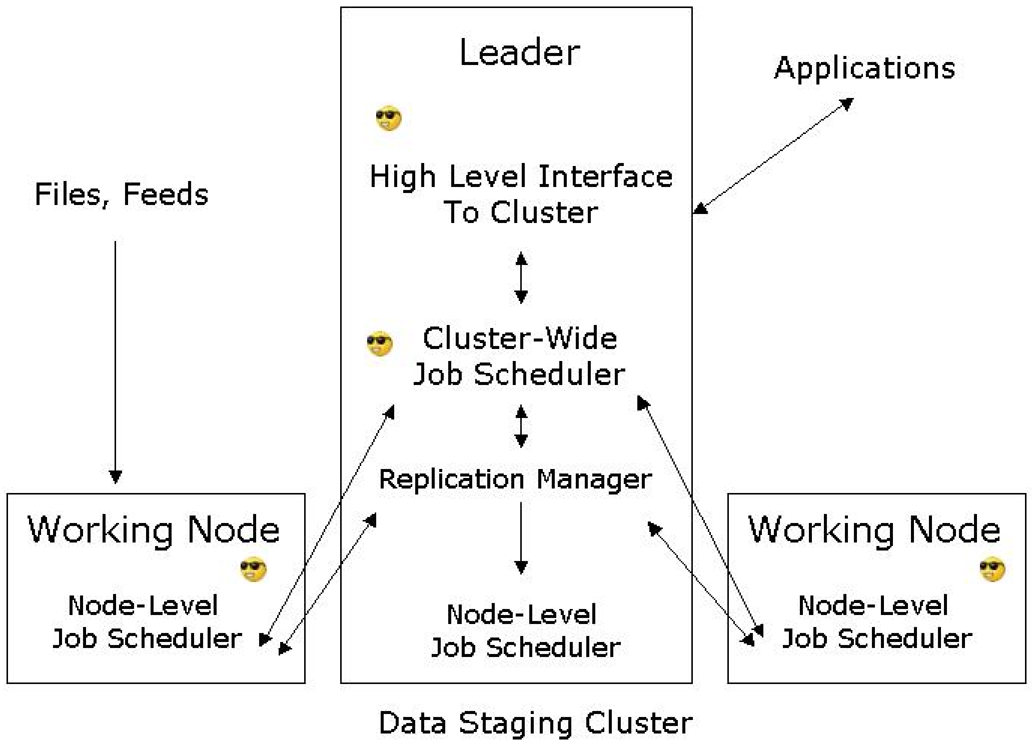
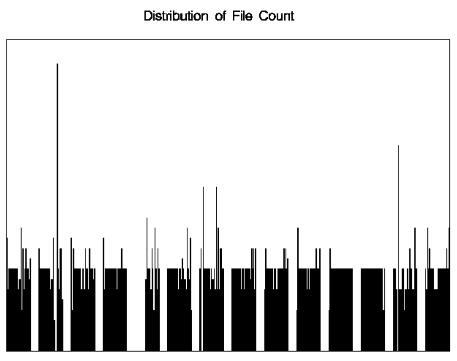


FIGURE 3

The idea is to use "agents" incorporating rules (courtesy Tom Kirk, AT&T Labs) to monitor the cluster for inconsistencies in data and process. The agents are vehicles for implementing an array of statistical and graphical tests for identifying inconsistencies in process and data. Another big part of the task is to implement the process and business rules articulated by the domain experts as constraints that can be expressed and validated in the rule-based programming techniques. Please see [2] for the relationship between data quality and knowledge engineering.

In order to simplify the explanation, we focus on a particular set of feeds that serve a single application. In reality, there are many applications that access hundred of feeds of thousands of files. For this application, we receive approximately 200 files a day of varying sizes, shapes and content. An important consideration is that we have a small window (a few hours) to ask for retransmission of lost or damaged files, to avoid permanent data loss.

As an initial alerting mechanism, we built a simple univariate, nonparametric control chart for detecting holes in feeds – either due to missing files or due to smaller file sizes than usual. Smaller file sizes indicate data that has never been sent or data that got lost in transit. While we show simple univariate charts in this paper, we are developing multivariate nonparametric tests as well. Figures 4 and 5 show the fluctuation in files received and the total file size by hour. We can clearly see the daily and weekly cycle in file sizes. The big spike corresponds to a large transmission that compensates for the data gap that just precedes it.



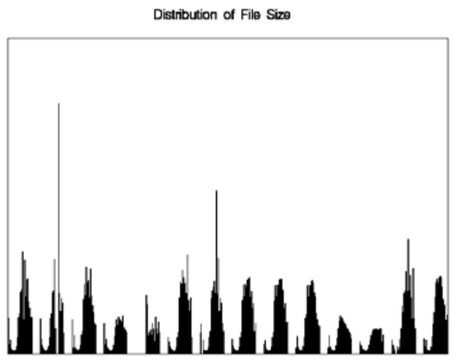


FIGURE 4 FIGURE 5

Figures 6 and 7 show the confidence bounds for a daily cycle of files to be received. We computed the predicted value and confidence bounds using metrics based on the median and the inter quartile range over a stable period of time. Figure 6 shows a normal cycle where the file sizes received were within acceptable bounds of the expected. Figure 7 shows an abnormal day with two hours of no transmission followed by a spike of cumulative transmission. This is often due to problems with transmission protocols. Note that the systems should be designed to handle such spurts in transmission.

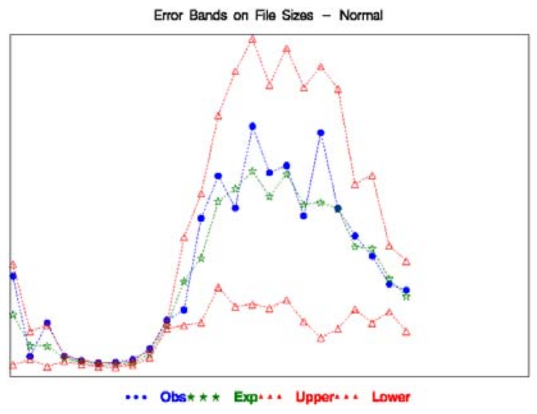


FIGURE 6

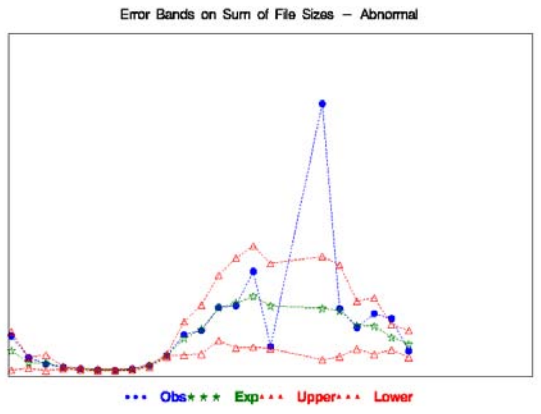


FIGURE 7

Next, we developed a set of rules for safely storing the data using replication by consulting domain experts. The rules, which represent the *desired state of the system*, can be roughly stated as: (1) Every file should be written to exactly two distinct tape volumes within 24 hours of being received. (2) Depending on the application, each file should be replicated on the appropriate cluster and be available in a certifiably accurate state for a specified period of time. The experts provide the specification parameters. (3) The *md5sum*, which is an encoding that uniquely represents the contents of the file, should be identical on all the replications. (4) The file sizes should be identical on all the replications, barring differences in compression algorithms. (5) There should be a one-to-one mapping between a file_id and its md5sum. This is to ensure that we are not burdened with file duplication other than mandated by the replication scheme. (6) Certain events happen in the life cycle of a file. Some of these are file received, file written to tape, failure in receiving tape, and being "consumed" by an application. Unexpected sequence of events e.g. file written to tape more than 3 times, multiple failures in receiving a file, or a file being consumed without being written to tape first, are worthy of scrutiny.

While any language can be used to implement these rules, we feel that a rule based programming language is ideal for capturing the constraint satisfaction flavor of these rules while providing flexibility in capturing and updating the ever-changing rules.

Inconsistencies Revealed: We implemented a variety of rules derived from the desired goal-state of the system mentioned above. We give a brief account of the inconsistencies that were revealed by validating the data against the rules. Of the 86,500 application-specific files accumulated during our study, 1350 files had mismatched md5 sums. A majority of these could be traced back to a bug in the file transmission protocol, where an empty file and the corresponding correct file would be transmitted within a short time of each other. The remaining were truly damaged files, truncated or otherwise mutilated during transmission.

We analyzed the sequence of events that happened in the life cycle of files. The four events that we mention here are Transmission Failure, Status, Tape Replication and Consumed by Application. Note that in reality there are many more events that happen. A file can be consumed by multiple applications. The interaction between applications can sometimes be inferred from these sequences, helping us to design a more efficient process.

In Table 1, we show a symbolic representation of the frequency of events and the number of files that exhibited that particular frequency of events. For example, the first data row indicates that there were 52,388 that had 0 transmission failures, that had 0 updates on the Status flag, that were written to tape once and that were consumed 0 times. Note that this is a form of data reduction where we represent the sequence of events by a frequency of counts of the events. While there is a loss of information (the exact sequence of events is lost as is the knowledge of the duration of the inter-event intervals), the frequency counts help us to compare the files based on event sequences in a simple manner. More complex analyses using point process methodologies are possible as well. We skipped many rows (around 90) in the table to include two peculiar examples. The last two rows are instances of files that experienced transmission failure 53 and 97 times respectively! Fortunately there are not too many such files. A further examination of the files revealed that they were caught in a software glitch in the file transmission software that was soon fixed by the vendor of the software.

Table 1. Multiple Events in the Life Cycle of Files

Event Frequency Sequence	File Frequency
0.0.1.0	52388
0.0.2.0	16222
0.0.2.1	7556
0.1.1.0	1848 **…**
53.1.2.1	**1**
97.1.2.1	**1**

5.2 Data Management: File Archiving

The next concern we had was with files being written to tape. Our desired state for the system is that a file should be written to tape exactly two times within 24 hours of receipt. Figure 8 indicates that a majority of files are not written to tape more than once (red region, written to tape once and no replications on hard disk), well past the recommended 24-hour deadline.

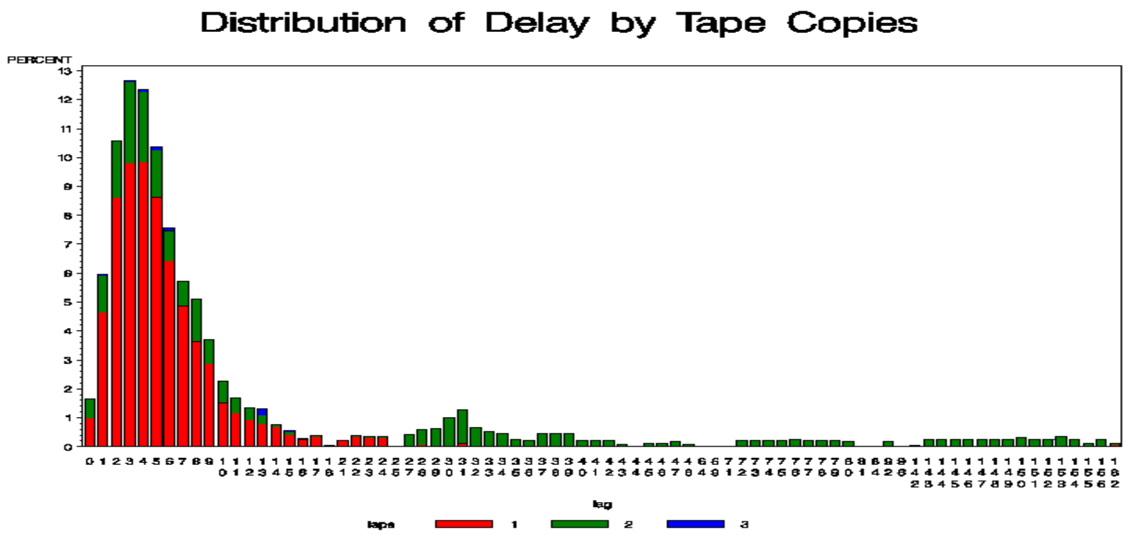


FIGURE 8

An examination of the process revealed that once every few months, a manual intervention would write hundred of thousands of files *en masse* to tape. However, in the interest of data safety, the process should be modified suitably. A simplification of the scheme we are currently implementing is to consider a data file to be in a certain life cycle phase. Although the actual categorization is more dynamic, a file can be considered in one of three phases: new files that have just been received by the cluster, in-process files that are the current objects of analysis (consumed files), and archive files that are ready to be placed on tape. For the planning analysis **initial states** may vary and must be computed by the rules. **Goal states** depend on the lifecycle phase of the tape. For example, for new files the goal is to have at least two copies of each file on each cluster in the data center, each on a different node. On the other hand, for archive files, the goal state is to have at least two copies of files, each on a different tape. Finally for in-process files there must be two copies of the files on a cluster, each copy on a different node and a copy on tape. These initial and goal states for each of the life cycle phase are shown in Table 2. Variables and constants in STRIPS begin with a "?" and, in our description, ?c1x refers to a specific node on the cluster c1 and ?t refers to tapes.

Table 2. Initial and Goal States in STRIPS notation.

Lifecycle Phase	Initial State	Goal State
New	{Havefile(?c1x)}	{Havefile(?c1x,?c1y,?c2x, ?c2y, …)}
In Process	{Havefile(?c1x)}	{Havefile(?c1x,?c1y,?tx)}
Archive	{Havefile(?c1x)}	{Havefile(?tx, ?ty)}

Space does not permit us to provide similar tables for the operators for this application and their preconditions and effects. We are currently in the process of implementing this in STRIPS.

6 CONCLUSIONS

In this paper, we have proposed an important area of research in managing and mining constantly changing, large-scale information feeds in production settings. The task of data mining in such a setting offers challenges that are very distinct from traditional data mining scenarios. Much of the ongoing work in data mining is analysis focused, and does not address the areas that seem to us the most problematic in a practical setting. The *mise en place* analogy is an accurate one. Most of the work in all the data mining projects we have been associated takes place in the preparation stage, prior to analysis. We feel that there are interesting problems to be solved in this stage, and the touchstones between constraint satisfaction, constraint management, and planning suggest ways that we could work together.

Adding domain knowledge to a planning system can be surprisingly effective, even when simple planning algorithms are used (McDermott, 1996 [8]; Bacchus and Teh, 1998)[1]). One interesting possibility might be to merge the new planning algorithms with methods that have proven effective in engineering expert systems (such as rule-based programming, e.g. Vesonder, 1988 [11]).

Because generalized search can be implemented as production rules, we think it will be possible to implement the planning algorithms in a rule-based system. In that case, the use of the rule representation would provide an effective platform for knowledge engineering, and would directly support a variety of methods that have proved effective in developing the practical rule-based systems of the past.

The success of such a scheme probably hinges on how much benefit can be gained from incorporating domain knowledge into the planning algorithm, and we would expect this to vary from application to application. In any case, constraint rules provide a natural path for incorporating domain knowledge into a planning system.

Solutions for this complex and dynamic problem can be potentially drawn from the areas of constraint management and planning. While the constraint database community addresses constraint management, there is a gap between the existing functionality and the problem that can be bridged through collaboration with the constraint management philosophies in the knowledge engineering community of AI. Our future work is in this direction.

REFERENCES

1. Bacchus, F. and Y. W. Teh, Making forward chaining relevant. In 4th International Conference on AI Planning Systems, pp 54-61 (1998).
2. Dasu, T., Vesonder, G. T. and Wright, J. R. Data Quality through Knowledge Engineering. In *Proceedings*

of the KDD, 2003, 705-710.
3. Fikes, R. E. and Nilsson, N. J.: STRIPS: A new approach to the, application of theorem proving to problem solving. Artificial Intelligence, 2 (3-4): 189—208 (1971).
4. Gray, J.: KDD 2003 - Invited Talk, On-Line Science: The Worldwide Telescope as a Prototype for the New Computational Science, (2003).
5. Hammond, K.: Explaining and repairing plans that fail. Artificial Intelligence 45:173-228 (1990).
6. Huang, K., Lee, Y. and Wang, R.: Quality Information and Knowledge Management.
7. Hume, A. and Daniels, S.: Ningaui: A Linux Cluster for Business. Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, June 10-15, 2002, Monterey, California, USA, 195-206.
8. McDermott, D.: A heuristic estimator for means-ends analysis in planning. In Proceedings of 3rd International Conference on AI Planning Systems, pp 142-149 (1996).
9. Pipino, L., Lee, Y. and Wang, R.: Data Quality Assessment, *Communications of the ACM*, April 2002. pp. 211-218.
10. Pyle, D.: Data Preparation for Data Mining, Morgan Kaufmann, San Fransisco, (1999).
11. Vesonder, G. T.: Rule-based programming in the Unix System. AT&T Technical Journal 67(1): 69-80 (1988).
12. Weld, D. S.: Recent Advances in AI Planning. AI Magazine, 20:2, 93-123 (1999).