Data Integration: The Current Status and the Way Forward

Michael Stonebraker
MIT
stonebraker@csail.mit.edu

Ihab F. Ilyas
University of Waterloo
ilyas@cs.uwaterloo.ca

Abstract

We discuss scalable data integration challenges in the enterprise inspired by our experience at `Tamr`[1]*. We use multiple real customer examples to highlight the technical difficulties around building a deployable and usable data integration software that tackles the data silos problem. We also highlight the practical aspects involved in using machine learning to enable automating manual or rule-based processes for data integration tasks, such as schema mapping, classification, and deduplication.*

1 Introduction

In this paper, we comment on the status and the issues in data integration from the perspective of `Tamr`, a commercial company that provides a novel solution to a long-standing challenge, namely, traditional *enterprise data integration*. Rather than relying on a rule-based approach, `Tamr` employs supervised machine learning as the primary way of combining large numbers of data sources. The company is based on the Data Tamer academic system [13].

Most large businesses are decomposed into independent business units to facilitate agility. Such units are typically free to "do their own thing", without being hindered by corporate-wide issues. For example, adopting a specific global schema for a specific entity type (e.g., customers) across all units is often impossible as the needs of these units are different. Waiting for consensus across all the business units means it would take forever to get anything done. This leads to *data silos* (one for each unit), where similar data are stored with different granularity, schema, and even contradicting and inconsistent details. A typical enterprise has many such silos, and a major goal of many enterprises is after-the-fact integration of silos for business gain. Such business gain often involves cross selling, a better understanding of the customer base or lowering the expenses of product lines. Such activities span many business units and require classifying, integrating, linking and aggregating the data from their silos.

Example 1 (GE Supplier Integration): `GE` has approximately 75 active procurement systems, with approximately two million records. The CFO determined that the company could save $\$100M$ per year from the following strategy: when the contract with an external vendor comes up for renewal in one of these procurement systems, empower the purchasing officer to discover the terms and conditions negotiated by his counterparts in other business units, and then allow him to demand most favored nation status. To accomplish this, 75 independently constructed supplier databases must be integrated. In fact, the value is mostly in the long tail since the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1]www.tamr.com

major suppliers are routinely scrutinized. Hence, a strategy that integrates only the major suppliers will fail to provide additional value.

A typical data integration workflow of the 75 different supplier databases is as follows: **(a)** Raw source data is "ingested" into a common system (often a data lake); **(b)** Records are transformed into common units (say Euros to Dollars) via a series of transformation rules, modules and services; **(c)** Errors are cleaned [8]; typically 15% of the values are missing or incorrect, and most integration tools expect these errors to be at least spotted or nulls to be identified; **(d)** The different schemas of the sources are matched to line up the columns. This step is crucial in enabling comparing records across sources; **(e)** Deduplication and record linkage tools [4, 7, 10] are used to cluster records. Each cluster of records is thought to represent the same entity; and finally **(f)** Golden canonical values must be selected for columns in these clusters (e.g., a common address must be selected for each supplier) to produce a final integrated data set that can be consumed by downstream spend analytics.

In effect, 75 supplier data sets are pushed through this pipeline to generate an integrated composite data set, through which purchasing officers can find the most favorable terms and conditions, among other possible queries. The pipeline has to be able to support incremental processing of new suppliers (or deleted suppliers), without the need to redo the whole workflow from scratch on all of the source data.

The previous example focused on a schema matching and record deduplication workflow of data that represents the same real-world entity scattered across silos. However, data integration often involves other tasks, such as classifying input records to known ontologies (e.g., standards or master dictionaries), as we show in Example 2. This classification step is often interleaved with the deduplication and schema mapping steps to carry out an end-to-end integration project.

Example 2 (GE Parts Classification): In addition to the schema mapping and deduplication tasks in Example 1, a second `GE` problem is to classify $20M$ *spend* transactions into an existing `GE` classification hierarchy (a given taxonomy for various parts). A spend category could be *machine parts* and a sub category could be *bolts*. A further classification could be *stainless steel bolts*. `GE` engineers wrote more than 500 rules to specify which bucket a spend record was in (e.g. *taxi rides* should be classified as *travel*).

A functional interactive workflow has to support a "rollback" functionality to support false starts, or trying different configurations for each of these steps or components. The workflow must be also able to call external modules, such as web services, locally or externally developed tools, or other software packages (i.e. acting as a master to other workflows). In addition, an integration workflow has to fit into workflows run by other more general pipelines (e.g., business intelligence or analytics workflows).

In our opinion, the most interesting data integration problems are ones that require scalability. Any methodology (including manual integration) is probably capable of integrating 1000 records from each of 10 data sources. However, for problems that deal with larger data scale like the one in the previous example, scalable execution of these pipelines is a must, and in fact, has been the main hurdle in deploying data integration software in real scenarios, despite the long history of the problem in both academia and industry.

This paper discusses the current state of the art and future challenges in *scalable data integration* from the perspective of `Tamr`, and is informed by interacting with large enterprises over the last five years. In Section 2, we discuss why traditional solutions do not scale, the use of machine learning for automation and for replacing/augmenting rule-based approaches, and we discuss why building such systems is hard due to scale, dirtiness of data, and resistance to adoption inside the enterprise. In Section 3, we shed some light on future directions and missing pieces that need to be addressed to build scalable end-to-end data discovery and integration systems. We conclude in Section 4 by a summary and final remarks.

2 Scalable Data Integration

2.1 Traditional Solutions Do Not Scale

The traditional wisdom in dealing with large scale data integration projects is to use what is referred to as a *Master Data Management* (MDM) system, such as those available from Informatica and IBM. Using an MDM system, one performs merge/purge (deduplication and golden value selection) often using a rule-based system, which allows users and experts to declare rules for transforming and classifying input data records. Before this step, a schema mapping exercise is performed typically by drawing lines between matching columns using a GUI in a (semi) manual way.

It is not hard to see that traditional MDM systems do not scale well: manual schema mapping obviously does not scale beyond tens of sources. For example, in one `Tamr` application there are more than 1000 data sets, each with approximately 500 columns. Manually linking $500,000$ columns is clearly not workable, and a scalable and more automated solution is required.

A lesser known fact is that rule systems for classification and record linkage do not scale either, as the system grows rapidly with a large number of correlated, overlapping and sometimes contradicting rules to cover all edge cases, which presents serious maintainability issues. In Example 2, the 500 rules designed by the `GE` engineers is about the limit of what a human can understand. However, these rules only classify $2M$ (10%) of the transactions. It is impossible to imagine a rule system with $5000+$ rules, which would be required to classify all $20M$ transactions.

As a result, MDM technology simply does not scale, and should only be used on small problems that are guaranteed not to require scalability in the future. Unfortunately, non scalability is very difficult to guarantee, since large enterprises reorganize regularly and buy and sell business units like monopoly properties.

2.2 Machine Learning for Scalable Data Integration

An obvious approach to scalability is to utilize machine learning (ML) to automate the various steps, such as the ones mentioned in the previous examples. In Example 2 (parts classification), we can use the $2M$ classified records as training data for an ML classification model, which will then classify the remaining $18M$ records, and this is exactly the `Tamr` solution implemented by `GE`. We see no other alternative to achieve the required scalability. In other words, we need to automate classification decisions (classifying records, matching columns, or linking similar records) using ML models that can leverage training data, which is either collected directly from humans in the loop, or in a weak-supervision fashion via leveraging existing rule systems. There are, of course, many engineering challenges to applying machine learning at scale, and the rest of this paper will consider some of them. However, in this section, we discuss the ML tactics that are likely to be successful in the commercial marketplace.

Conventional ML models, such as regression, decision trees, random forests, SVM, and Naïve Bayes models, are well understood, and often require elaborate feature engineering exercise to provision. In the last few years, there has been dramatic interest in "deep learning" (neural networks). The pioneering work came from Google, which successfully applied neural networks to find images of cats, and this technique has been successfully applied in a variety of other image understanding and text-oriented problems. In `Tamr` applications, there is little-to-no image and text data, rather, it is essentially all structured data such as found in relational DBMSs.

As we discuss below, using deep learning in enterprise data integration applications remains challenged by the scarcity of training data, since most of these models require large quantities of labeled data to learn the classification task; and by the lack of reasonable explanations of the output decisions.

- *Training Data:* In general, generating training data in enterprise data integration tasks is a huge problem: consider three plausible terms; *IBM-SA*, *IBM Inc* and *IBM*. Intuitively, these might represent the Spanish subsidiary, the US subsidiary and the overall company. Deciding if these should be consolidated as duplicates or kept as separate meaningful related entities can be performed by a domain expert. Even then, the decision to consolidate or not could be determined by the question being asked. However, automatically "learning" these complex relationships among these entities, and the right merge/separate decision will probably require massive number of labeled data in a deep neural network model.

In the hundreds of applications we have seen through `Tamr`, domain expert time is a scarce commodity that must be ruthlessly economized and cannot be simply used to label millions of training examples. These experts are typically well-paid, busy business experts, who view generating training data as a low priority task. Most of `Tamr` customers have a specific budget for domain expert time, and it is the "long pole in the tent" for the project. Hence, it is easy to see why ML models with less training data requirements are highly preferable than those with large training data demands (e.g., deep learning models). Advances in automating training data collection, such as the `Snorkel` project [11], might relax this constraint. However, in `Tamr` applications, we have not seen the type of rule coverage and the significant overlap of rules that `Snorkel` requires to produce good results. Until we effectively solve the problem of generating large and high-coverage training data, enterprise solutions will likely depend more on conventional ML classifiers with modest training requirement.

- *Explainability:* In many enterprise integration problems, one must be able to explain why the ML model took a particular action. For example, a predictive model that generates approvals for real estate loans must be explainable: *you were denied a loan because of such and such reasons*. If this explanation is not forthcoming, then lawsuits are inevitable, and adoption of these models is highly unlikely. Conventional ML is at least marginally explainable; deep learning models are not. Again, this will be an impediment to the applicability of deep learning in enterprise integration applications. It is conceivable that deep learning models will become more explainable in the future, and there is considerable research in this area. However, until this happens, we do not foresee the adoption of these models at a wide scale, or as a primary classification method in enterprise data integration tasks.

For these reasons, at least in the short-term, these conventional ML models with modest training data requirements will prevail when integrating structured data at scale.

2.3 Scalable Data Integration is Fundamentally Difficult

Consider a second example, that of `Carnival Cruise Lines`. It is a holding company with 10 operating brands (e.g., *Princess*, *Carnival*, and *Costa*). Each has its own mechanism of identifying spare parts and has a parts classification hierarchy. Each has its own mechanism for depoting spare parts (e.g., on the boat, on the dock, or in the warehouse). `Carnival` wishes to share spare parts across brands (since everybody uses the same straws, for example) and to optimize spare part placement. This requires performing parts integration at scale ($4M$ total parts).

There are two reasonable ways to proceed. First, one can directly integrate the $4M$ source parts records. Another alternative is to integrate the various classification schemes into a single master classifier, then classify the $4M$ parts, and finally, deduplicate each bucket in the master scheme. It is not obvious which way will work better, and the best choice is clearly data dependent. We do not expect such issues to be tackled by unskilled personnel anytime soon. In the meantime, engineering of data integration pipelines is not for the faint of heart; `Tamr` has a substantial collection of trained field engineers who help construct customer pipelines.

In addition to the large number of design choices, many of the needed algorithms have at least a quadratic complexity. For example, clustering of similar source records for deduplication has an $O(N^2)$ computation step to compare all pairs of records. For $10M$ source records, that entails close to 10^{14} comparisons. If each one requires 10 microseconds, the complete calculation will consume more than three years. This is obviously a non-starter. A serious product that includes deduplication must knock down the complexity by blocking and parallelism [2]. Products with an $O(N^2)$ algorithms, even on custom hardware, are likely to take an unacceptable amount of time at scale.

2.4 Data Cleaning in Integration Workflows

As noted above, it is reasonable to assume that 15% of all data is missing or wrong in a typical enterprise repository. The best first steps are to look for missing values and outliers using standard techniques, and perform data wrangling transformations [9]. Specifying a collection of data cleaning rules will also bear fruit. However, to do better, we often need application-specific cleaning tools, since "one size does not fit all". There is promising work in this area, such as the `HoloClean` project [12], which uses statistical inference to impute missing values and to suggest repairs for erroneous cells. However, it is not clear how to integrate these tools in a scalable way in workflows like the ones provisioned by `Tamr`, and in which order these cleaning tools should be used when interleaved with the main integration components [1].

A promising data cleaning tactic is to generate clusters of records, thought to represent the same entity using a deduplication tool. For many attributes, only one value is appropriate, and one can use a golden record system to pick it. For example, in the `GE` supplier integration task, a single supplier must have the same Dow Jones identifier or maybe the same address. If various records contain non-duplicate values for these fields, then a golden record system can be an effective data cleaning tool.

However, to use it, one must perform deduplication to create clusters, which is fundamentally inaccurate and requires some human involvement as noted above. The subsequent golden record system also requires human input and is operating on, perhaps, inaccurate data. In effect, a human can improve the training data, improve the clustering of records or work on the golden record system. These tactics are intertwined, and it is complex to decide what to do next. Since the golden record system is a cleaning tool, these tactics must also be compared against other cleaning tactics in any end-to-end model.

In summary, the various data cleaning tactics are not independent, complicating a general model to optimize which tool to use next. A deeper understanding and explicitly modeling of this interaction among cleaning, integration, and benefit of human involvement is essential in devising a self-configurable workflow.

2.5 Human Involvement

In `Tamr`, there are two kinds of user roles; there is a workflow designer and/or manager who is typically an IT professional who designs and manages the execution of a workflow. In addition, there are domain experts, who can answer questions, sometimes via basic channels such as e-mail. Such questions include assisting validating training data, validating clusters and validating data cleaning and golden record decisions. The two roles are carefully separated. In our opinion, one cannot have a single interface for both kinds of roles. Any system with a single interface is not going to work well.

2.6 Antibodies

Inevitably a data integration project changes the power dynamics inside an enterprise. In engineering, there are often custom one-off systems that are rendered obsolete by a successful integration effort using `Tamr` software. Such engineering groups will often throw up "engineering antibodies" to try to kill off the new effort. In addition, there are often business groups that are affected positively or negatively by an integration project. The negatively affected groups often throw up business antibodies.

`Tamr` projects are invariably instituted by business people who own a particular business problem. Invariably, it is up to them to deal with both kinds of antibodies. In summary, most data integration projects at scale have a political component, which must be dealt with.

3 The Future of Data Integration

In this section, we highlight future directions and challenges that are likely to have a big impact in building end-to-end scalable data integration in the large enterprise settings.

High Provisioning and Tuning Cost: Right now, enterprise data integration is a high touch environment, requiring highly trained personnel to be involved in authoring the right workflows, engineering the right configurations in each component, choosing the most effective cleaning tactics, and overseeing the availability of sufficient and meaningful training data. Over time, we expect the problems indicated in the previous section to be rendered easier by better tools. As such, we expect this research area to be ripe for future innovation.

Data Discovery: In addition, we expect a new application area to be commercialized that of supporting the work of data scientists. For example, `Merck` has approximately 4000 Oracle databases, a large data lake, uncountable individual files and the company is interested in public data from the web. `Merck` data scientists spend at least 90% of their time finding data sets relevant to their task at hand and then performing data integration on the result. Roughly speaking, data scientists have a discovery problem to identify data sets of interest, followed by the same data integration challenge faced by traditional enterprises. No data scientist we have talked to reports spending less than 80% of his time on discovery and integration. It is an obvious challenge to lower this percentage.

Integrating data discovery activities into traditional data integration operations is a challenging task. For example, there are often many current data catalogs, which collect metadata about enterprise objects, with rudimentary metadata such as object names, and attributes types. To become interesting discovery products, such systems must include relationships between objects (e.g., *Column A in Table 1 is related to and generalizes Column B in Table 2*). In this way, if a data scientist wants to know about data sets that deal with the effect of the drug *ritalin* on mice, he will likely find several data sets dealing with the topic. In order to proceed, this user will need to know relationships between the various data sets. This will require a much more elaborate catalog system, and a step in this direction is indicated in the `Aurum` project [5, 6], where such relationships are automatically discovered using syntactic and semantic features of the various tables and columns.

Moreover, there are often several possible ways to connect data sets of interest. For example, in the MIT data warehouse, there are several ways to connect students to departments, for example, *majoring in*, *took a course from*, or *interested in*. Unlike traditional enterprise integration, where there is usually a well-defined collection of data sets to be integrated in a well-defined way, in data science, neither is well-defined. Hence, there is a join path discovery problem which must be solved. Some of these challenges are handled in the `Data Civilizer` project [3] that builds on the relationships discovered via `Aurum` to devise possible and semantically coherent join paths, which join relevant raw data sources together to feed the analytics of a specific task.

Human Involvement: A data integration project is often run by a single data scientist. The same data scientist must be both the domain expert and the project manager. To support this application, data integration systems have to be much easier to use and require much less human interaction. These are topics for future research.

Batch vs. Exploratory Workflows: A typical enterprise data integration project starts with a collection of data sets, say the 75 supplier data bases in Example 1, and the project manager pushes these data sets through a well-defined pipeline, with a well-defined goal. In contrast, a data scientist is typically in exploration mode. He tries various things, sees what works and then uses this information to explore other related tasks. We expect supporting such users will require more serious support for provenance, rollback and conditional workflows.

Model Reusability: We fully expect other uses for data integration will occur over time. Specifically, it is clear that every large enterprise on the planet has a supplier integration problem. Moreover, there is substantial commonality between the applications of enterprises. After all, everybody buys office supplies from a small number of large vendors. Hence, imagine performing data integration for a certain project, then, use this result as training data for a second project and so forth. Hence, transfer learning and the reuse of machine learning models holds enormous promise. The impediment is that enterprises tend to view their supply chains as highly confidential. How to get the benefit of temporal machine learning without violating confidentiality is an interesting research question.

4 Conclusion

Integrating the large structured data repositories in the enterprise, which are often scattered across many silos, requires building end-to-end scalable workflows. Current semi-manual and rule-based systems simply do not scale and cannot accommodate the continuously growing data across the various business units. While machine learning techniques is an obvious way to go, multiple practical considerations arise, such as, the scarcity of training data, the need to explain the results to business owners, and the high cost of involving domain experts. These factors play a big role in choosing the right machine learning models to deploy, and in exploring the large number of design choices in provisioning data integration workflows. Moreover, data discovery and exploration tools will become essential to help data scientists find relevant data sets, and navigate through the messy data lakes that are currently built by the large enterprises.

References

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[2] X. Chu, I. F. Ilyas, and P. Koutris. Distributed data deduplication. *PVLDB*, 9(11):864–875, 2016.

[3] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*, 2017.

[4] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[5] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *34th IEEE International Conference on Data Engineering, ICDE, Paris, France*, 2018.

[6] R. C. Fernandez, E. Mansour, A. Qahtan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *34th IEEE International Conference on Data Engineering, ICDE, Paris, France*, 2018.

[7] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[8] I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[9] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 3363–3372, 2011.

[10] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. 2010.

[11] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, 2017.

[12] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[13] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.