Explaining Data Integration

Xiaolan Wang Laura Haas Alexandra Meliou
University of Massachusetts, Amherst
{xlwang, lmhaas, ameli}@cs.umass.edu

Abstract

Explanations are an integral part of human behavior: people provide explanations to justify choices and actions, and seek explanations to understand the world around them. The need for explanations extends to technology, as semi-automated and fully-automated systems support crucial activities and increasingly important societal functions. The interpretability of these systems and the ability to explain their decision processes are crucial in developing trust in the systems' function. Further, explanations provide opportunities for systems to interact with human users and obtain feedback, improving their operation. Finally, explanations allow domain experts and system developers to debug erroneous system decisions, diagnose unexpected outcomes, and improve system function. In this paper, we study and review existing data integration systems with respect to their ability to derive explanations. We present a new classification of data integration systems by their explainability *and discuss the characteristics of systems within these classes. We review the types of explanations derived by the various data integration systems within each explainability class. Finally, we present a vision of the desired properties of future data integration systems with respect to explanations and discuss the challenges in pursuing this goal.*

1 Introduction

Human perception of and reliance on explanations shape all aspects of human activity and our interactions with the world: people rely on explanations to make decisions, justify actions, predict events, and understand the world around them [35, 36]. At the same time, advances in technology and the big data revolution have fundamentally impacted human activity and interactions: data and algorithms play a major role in product recommendations, news personalization, social media interactions, autonomous vehicle decisions, and even medical diagnosis and treatment. The computer systems supporting these tasks are becoming increasingly complex, and their function and decision processes are often poorly understood, even by domain experts. This obscurity is detrimental to user trust in the system operation, and can potentially hinder adoption.

Explanations can significantly ease the interaction between humans and systems: they help humans understand, justify, and consequently trust system function, as well as provide humans with support to debug, diagnose, and improve the systems. This has led to a strong push in several research domains to develop support for explanations in computing systems, such as interpretable machine-learning [23], and DARPA's explainable AI initiative [30]. In the data management community we have also seen a fruitful line of research focusing on supporting explanations in relational [60, 51, 43] and non-relational systems [15].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

In this paper, we focus on explanations in data integration systems. Data integration is a crucial component in data analytics and data management applications, involving the acquisition, storage, and management of data gathered from heterogeneous data sources, through a uniform query interface [38]. Unlike fully-automated systems, data integration systems often interact with humans extensively. Many existing data integration systems are semi-automated and require several iterations with human input. Even automated data integration tasks may still rely on humans to remove the uncertainty from the results (e.g., in schema mapping). Support for explanations in data integration systems facilitates the interactions between these systems and human users, improving not only the user experience, but also the overall quality of these systems' function.

In this paper, we offer a review of existing data integration research and systems with respect to their ability to provide explanations. We present a new classification of the existing work into three *explainability* categories, and discuss the characteristics of each class (Section 2). We review existing solutions for four core data integration tasks with respect to our classification (Section 3). We then conduct detailed comparison and analysis over existing explanations in data integration systems (Section 4). Finally, we present a vision of desirable explanation properties for future data integration systems and discuss the challenges in pursing this goal (Section 5).

2 Classifying data integration systems' explainability

In this section, we present a new classification of data integration systems with respect to their explainability and the type of explanations they produce. In general, explanations can be categorized into two major classes: (1) *causal* explanations typically focus on *how and why* an outcome was derived; (2) *non-causal* explanations typically focus on *what* the outcome is. For example, for the schema matching pair `(class, course)`, the explanation *"more than* 90% *of the values in attribute* `class` *and attribute* `course` *match"* is causal: it explains why these attributes were matched. In contrast, the explanation, *"attribute* `class` *in schema A and attribute* `course` *in schema B match with each other"* is non-causal: it explains what the schema matching output represents, but it does not reveal how or why it was derived. Some data integration systems focus on causal explanations as an *explicit* objective. However, causal explanations may also be *implicitly* derived from systems that do not target them directly. Systems that require human involvement often focus on non-causal explanations to make their results understandable to and allow for feedback from human users. Finally, a large number of data integration systems do not derive explanations of any kind (unexplainable systems). Figure 1 provides an overview of our classification of data integration systems. We proceed to describe our classification and discuss the characteristics of systems within each class.

Explaining systems

Our first class includes methods and systems that focus on providing causal explanations as an explicit objective. We call this class *explaining systems*. Such systems build connections between evidence and results in an attempt to explain *how* and *why* particular results are derived. The purpose of some of these systems is only to derive explanations of a data integration process, and they treat data integration results as part of their input. Other systems aim to derive results, as well as their explanations, at the same time. Explaining systems often use conciseness as an objective and performance metric in generating and summarizing explanations. As a result, the causal explanations they produce tend to be simple and understandable, and thus easily accessible to human users. Explaining systems produce explanations that allow users to understand and validate results, and engineers to debug, diagnose, and fix problems in the systems.

Explainable systems

Outside of the explaining systems class, causal explanations are not an explicit goal. The majority of data integration systems focus on other goals and metrics, typically, higher accuracy, scalability, and efficiency.

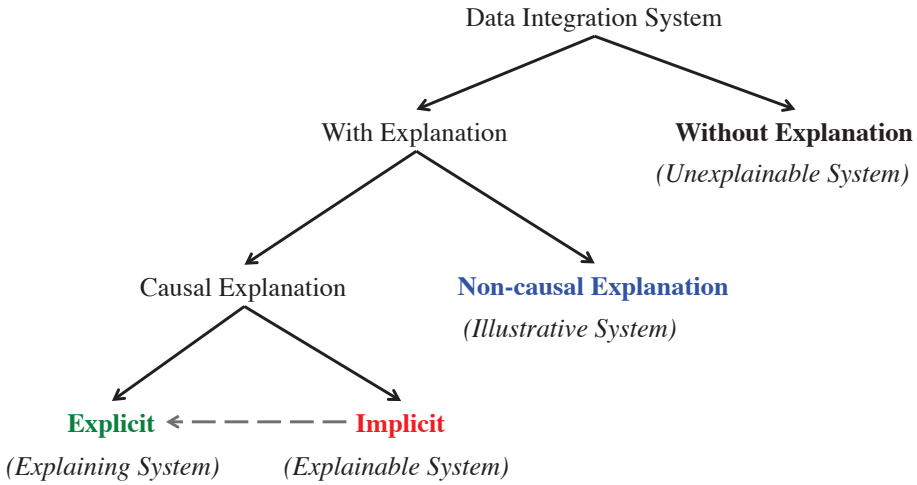

Figure 1: Explainability classification for data integration systems

However, causal explanations may still be present implicitly in approaches that do not target them directly, derived as a by-product of the system processes. These systems comprise the *explainable systems* class. In this class, since explanations are not an explicit objective, the explanations produced are typically suboptimal with respect to common explanation quality metrics, such as conciseness. Frequently, these explanations are complex, long, and hard or even impossible for humans to interpret and digest. However, there is potential to improve these explanations in a post-processing step, which summarizes and abstracts them. Such a transformation would convert explainable systems to explaining systems.

Illustrative systems

Our third class, *illustrative systems*, includes systems that provide non-causal explanations for their results or processing steps. Non-causal explanations are common in data integration systems that prioritize human involvement as a way to improve system function and result quality. In these systems, non-causal explanations, such as simplification, examples, and visual demonstrations, are often used to ease the processes of tapping into human knowledge as an information source and employing human intelligence to solve data integration tasks. For example, non-causal explanations make it possible for a human user to provide feedback for a schema mapping application by simplifying the possible mappings and demonstrating each mapping through examples. Note that human-in-the-loop systems are not necessarily illustrative systems: some human-in-the-loop systems may provide causal explanations and some may not provide any kind of explanation.

Unexplainable systems

The remaining data integration systems are unexplainable systems. Unexplainable systems often use highly complex procedures or purpose-built code to generate results and they do not interact with humans during the integration process. Unexplainable systems may derive high quality results extremely efficiently, but they can be hard to debug and analyze. Note that although many data integration systems, especially those with probabilistic-based approaches, are currently unexplainable, advances in explainable AI [30] could impact their explainability classification in the future and may potentially transform unexplainable systems into explainable ones. We will not discuss this class of systems further in this paper.

		Explaining systems	Explainable systems	Illustrative systems
Schema Level	**schema matching**	iMap [17], S-Match [28, 53] mSeer [9]	CUPID [42], Peukert et al. [48], YAM++ [45], AutoMatch [5]	AgreementMaker [13, 14] COMA++ [3], CCQ [63]
	schema mapping	Spider [1, 12], TRAMP [27]	RiMOM [56], CMD [37]	Clio [44, 62], Muse [2], Tupelo [25], Mweaver [50], Prompt [47]
Data Level	**record linkage entity resolution deduplication**	None	Davis et al. [16], Whang et al. [57], HIL [32], Li et al. [40], ActiveAtlas [54, 55]	Alias [52], GCER [58], DIMA [41], C3D+P [11], D-Dupe [8, 34]
	data fusion truth discovery	Dong & Srivastava [22], Popat et al. [49], PGM [46]	VOTE [18], Wu & Marian [59]	CrowdFusion [10]

Table 1: Explainability classification of core techniques in data integration.

3 Data integration tasks through the lens of explainability

The field of data integration has expanded in many directions [31, 21, 29], including tasks such as schema matching, schema mapping, record linkage, data fusion, and many more. In this section, we study and review existing approaches for four core data integration tasks, two for schema level integration and two for data level integration, and summarize them based on the classification of Table 1.

3.1 Schema matching

Schema matching, a fundamental requirement for schema mapping, focuses on finding the correspondence among schema elements in two semantically correlated schemata. Existing schema matching solutions leverage a wide variety of techniques, from heuristics, to rules, to learning-based approaches. These techniques give them different abilities to provide explanations. Here we discuss some representative systems in each of the three explainability classes.

Explaining systems. Explaining systems for schema matching have various goals, such as result justification or system debugging. **iMap** [17] is a semi-automated schema matching system that leverages users to finalize matching decisions. To assist users in making matching decisions, iMap provides insight into the derived matches by presenting *causal explanations* that are associated with each match. iMap does this by constructing a dependency graph that abstracts the process and the major decisions in deriving a match; the explanations reflect causality through the way they are generated. iMap further visualizes the dependency graph explanations such that they can be easily interpreted by users. **S-Match** [28, 53] uses explanations to gain trust from the users: it has a multi-level explanation module that generates high-level, concise explanations, as well as more verbose explanations. A high-level explanation is typically a short piece of natural language without any technical detail about the matching process; verbose explanations exploit both background knowledge and the logical reasoning of the matching process. **mSeer** [9] is an analytic tool that uses explanations to debug schema matching systems. mSeer does not generate matches itself. Instead, it takes the output of a matching system and produces a concise matching report, in which it displays all the derived matches, and the aggregated positive (e.g., matchability statistics) and negative (e.g., common matching mistakes) evidence for each match. iMap, S-Match, and mSeer belong to the class of explaining systems as they all treat deriving causal explanations as an explicit goal:

S-Match and iMap explain how a match is derived, and mSeer explores why a match is true (or false).

Explainable systems. Rule-based techniques, such as **CUPID** [42] and **Peukert et al.** [48], are known to be explainable and interpretable as rules are often self-explaining. For example, a rule "two schema elements match with each other if they are semantically similar" directly explains why two attributes are considered as a match. Machine-learning techniques have also been widely applied to schema matching problems. Some of these machine-learning techniques are likewise known to be explainable and interpretable, such as decision tree [26] and naïve Bayes [39]. For example, **YAM++** [45] leverages decision trees to combine different terminological similarity measures in order to justify the candidate matches. In each decision tree, non-leaf nodes represent the similarity measures and the leaf nodes indicate whether the match is valid or not. Thus, the matching decision is explainable by tracing the path through the decision tree. **AutoMatch** [5] uses naïve Bayes analysis and data samples of the attributes to determine the matching relationship. Their results are explainable since the conditional probability for each matching decision is calculated independently and the probability value directly reflects its contribution to the final matching decision. The key factor that distinguishes the above systems from explaining systems is that they do not produce explanations as part of their output. In addition, their explanations are often hard to understand as they do not restrict the number of rules ([48]), the size of the decision tree (YAM++), or the number of data samples (AutoMatch).

Illustrative systems. Due to the complexity and intricacy of the schema matching process, researchers have realized the limitations (in terms of precision and recall performance) of fully-automated schema matching techniques [24]. Therefore, many schema matching systems require human intervention and user validation [17, 63, 14, 47, 3, 33] to further refine their results. Most of these human-in-the-loop systems will iteratively ask users to accept or reject a selected set of candidates and then update their matching results accordingly. Meanwhile, they also provide a variety of non-causal explanations to facilitate these interactive processes. Some systems visualize the matches through user interfaces in order to help users understand the matches. Such systems include **AgreementMaker** [13, 14], and **COMA++** [3]. Another thrust of such systems focuses on leveraging the intelligence of the crowd, instead of domain experts, to answer the questions (e.g., **CCQ** [63]). Since the crowd often has little or even no background knowledge, these crowd-based systems typically simplify their questions to make the requested matching feedback understandable and accessible. Instead of explaining how the results are produced, illustrative systems in schema matching focus on explaining and demonstrating the results themselves through visualization (AgreementMaker, COMA++) or question simplification (CCQ).

3.2 Schema mapping

Schema mapping, a core operation for all data integration systems, describes how a source database schema relates to a target database schema [6, 4]. The mappings, typically generated from the matches between schema elements, form the basis for transforming the data from the source into the target. Schema matching and mapping are two strongly correlated components in data integration, and many schema mapping systems are also equipped with a schema matching module.

Explaining systems. Explaining systems in schema mapping primarily focus on system debugging. **Spider** [1, 12] is a debugging tool for exploring and understanding schema mappings. It does not identify mapping errors, but rather generates *routes* that describe the causal relationship between source and target data with the schema mapping. Thus it provides a platform that allows users to find the errors more easily. On the other hand, **TRAMP** [27] assists users in identifying mapping errors; it uses mapping provenance to explore common mapping errors and report the analysis, together with the evidence, to the users. Both systems consider deriving concise explanations as one of their objectives: Spider ranks the explanations and only returns the top ones; TRAMP always highlights the conclusion of the explanations – the identified errors – in its results.

Explainable systems. As with schema matching techniques, systems that leverage interpretable learning-based approaches are also explainable. For example, **RiMOM** [56] uses naïve Bayes to decide whether there exists

a mapping and, further, the type of the mapping between schema elements. **CMD** [37] poses a new direction for solving schema mapping problems by taking advantage of both rule-based and learning-based approaches. It leverages probabilistic soft logic (PSL), a scalable probabilistic programming language over weighted first-order logic rules, to derive the schema mapping. CMD first expresses the candidate mappings through first-order logic and then infers the weights of the mappings. The results of CMD are explainable since one can learn the contribution of a candidate mapping through learned weights. As with other interpretable learning-based approaches, RiMOM and CMD do not return these explanations, hence they are explainable, but not explaining; moreover, these explanations could be hard to understand due to the complexity and number of the sub-decisions (RiMOM) and rules (CMD).

Illustrative systems. There are many schema mapping techniques that rely on users to reduce the uncertainty of the mapping and to make the final mapping decisions. However, as the mappings are often complex programs, explaining the mappings to the users becomes a critical issue. Some systems, such as **Clio** [44, 62], **Muse** [2], **Tupelo** [25] and **Mweaver** [50], use data examples to explain the mappings; other systems, e.g., **Prompt** [47], rely on graphical interfaces to visualize the mapping between the source schema and the target schema.

3.3 Record linkage

Record linkage (also known as entity resolution and deduplication) is the problem of identifying records that refer to the same logical entity, and it is an important task in integrating data from different data sources. Unfortunately, to the best of our knowledge, *none of the existing record linkage systems explicitly provide causal explanations* of their results. However, many record linkage systems are explainable because of the method they use in determining the matching records.

Explainable systems. Explainable systems for record linkage often rely on approaches that naturally contain explanations. Rule-based record linkage methods typically expose understandable logic. For example, "$same\_name(t_1, t_2) \rightarrow duplication(t_1, t_2)$", intuitively means: "two records refer to the same entity if they have the same name". There are many rule-based record linkage systems, and they often rely on different methods to derive the linkage rules. For example, **ActiveAtlas** [54, 55] uses decision trees to learn the mapping rules over the records; **Whang et al.** [57] and **Li et al.** [40] leverage dynamic rules, learned from an iterative process, for discovering duplicate records; **HIL** [32] applies SQL-like syntax rules to accomplish entity resolution, mapping, and fusion tasks. Meanwhile, some record linkage solutions use explainable probabilistic-based approaches, combined with rules, to resolve the uncertainty in the mappings. Such approaches can be explainable as well. For example, **Davis et al.** [16] is explainable since it combines two explainable components, rules and naïve Bayes, to solve the record linkage problem. Again, the explanations are implicit in the specifications, not produced by the systems, so these are explainable, not explaining systems. Further, these systems usually include many rules, sub-decisions, or explainable components, which increase the overall complexity of their explanations. Therefore, in most cases, their explanations are still hard for humans to understand.

Illustrative systems. Researchers have developed a wide variety of record linkage solutions [52, 8, 58, 11, 41] that employ human intelligence. However, obtaining user input efficiently and effectively remains challenging. Many record linkage systems try to reduce the number of record pairs that need to be justified by humans. **Alias** [52] attempts to limit the manual effort by reducing the number of record pairs according to the information gained from them; **GCER** [58] measures the quality of questions (or record pairs) based on their impact over the system and selects questions that lead to the highest expected accuracy; **DIMA** [41] evaluates the questions according to their impact on the monetary cost and selects those that reduce the cost. We classify these as illustrative systems since they offer, in the form of questions, non-causal examples of possible linkages, and try to reduce the number of such questions to make it easier for the user to understand. **C3D+P** [11] eases the tasks handed to humans along another dimension – it simplifies and summarizes record descriptions by selecting a small subset of critical attributes. **D-Dupe** [8, 34] provides a visual interface that shows the relational

neighborhood for each record in a candidate pair to facilitate users' decisions.

3.4 Data fusion

Data errors and conflicting information are common occurrences across different data sources. Data fusion focuses on resolving conflicts and determining the true data values, leveraging information in heterogeneous data sources [7, 20].

Explaining systems. Explaining systems for data fusion explore explanations from both structured and unstructured data. **Dong & Srivastava** [22] leverage Maximum A Posteriori (MAP) analysis to make data fusion decisions about data stored in a DBMS. MAP analysis is explainable since a probability value for each candidate decision is calculated independently. However, the raw MAP explanations are complex and excessively detailed; Dong & Srivastava [22] abstract them through categorization and aggregation to produce succinct and understandable explanations for the decisions. Unlike the majority of data integration tools, which focus on structured or semi-structured data, **Popat et al.** [49] fuses data in unstructured format. Instead of fusing conflicting data values, this work tries to determine the credibility of natural language claims and explore the evidence, identified from articles on the Web, to support or refute the claim through Conditional Random Field (CRF) analysis. Explanations in this system are easy to understand as they are all in natural language. In a similar vein, **PGM** [46] also targets discovering the truthfulness of natural language claims. PGM further improves the understandability of the explanations by aggregating the evidence and visualizing them through a user interface.

Explainable systems. Many data fusion systems are based on probabilistic analysis, such as Bayesian analysis [18], probabilistic graphical models [19, 64] and SVM [61], and most of these models are hard to explain to humans with little domain knowledge. **VOTE** [18] discovers true values by incorporating both the data source dependence, i.e. the copying relationship between data sources, and data source accuracy. VOTE is explainable since it relies on several explainable steps: First, it uses Bayesian analysis to calculate the probability of dependence and accuracy among data sources; second, it estimates the vote count of values, combined with the dependence probabilities, through a greedy algorithm; finally, it combines the vote count and accuracy of the data source through a weighted sum formula. **Wu & Marian** [59] use a scoring mechanism, based on the score of the source and scores of values within the data source, to decide the actual result of a query. Similar to other explainable systems, VOTE [18] and Wu & Marian [59] do not derive explanations directly and their explanations may also be too complex to interpret in practice.

Illustrative systems. Incorporating human intelligence in solving data fusion problems is challenging as humans may be misled by conflicting information in data sources and the Web. **CrowdFusion** [10] tackles this challenge by allowing mistakes in the crowd answers: it assigns a probability for the correctness of an answer, and assumes that crowd answers follow a Bernoulli distribution. CrowdFusion adjusts the fusion results upon receiving relevant answers from the crowd and optimizes its questions by maximizing the questions' entropy.

4 Properties of explanations

Explanation coverage and understandability. So far, in this paper, we have categorized explanations in data integration as one of two types: causal or non-causal. A causal explanation is essentially evidence, or a summary of evidence that supports or refutes a data integration outcome. A non-causal explanation does not provide any evidence of the data integration process, but helps users understand the results. In this section, we further examine these explanations along two metrics, coverage and understandability. *Coverage* measures the amount of evidence with respect to the results of a data integration process. More precisely, 100% *coverage* means that one can replicate all decisions of the algorithm purely based on the explanations, whereas *lower coverage* means that one can only replicate a smaller portion of such decisions. *Understandability* intuitively measures the difficulty in understanding the explanations.

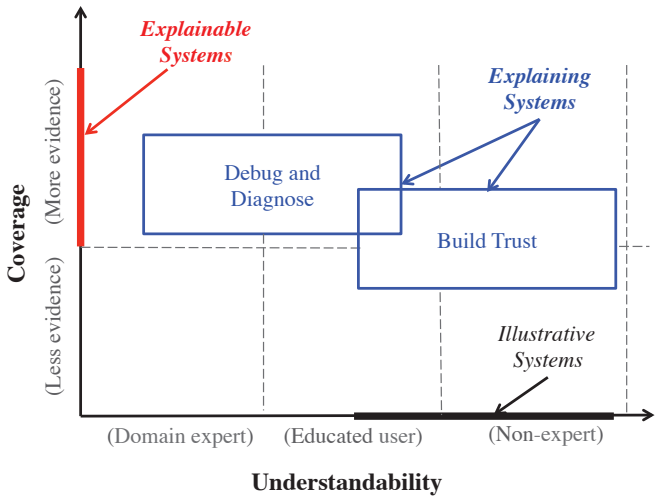

Figure 2: Coverage vs. understandability

Figure 2 presents our three explainability classes with respect to coverage and understandability. Explainable systems have high coverage since every step in their algorithm is interpretable. However, they are typically hard to understand, even for domain experts, since the explanations are usually complex and long. Illustrative systems present the opposite behavior: they do not cover any evidence of the data integration processes—their goal is to explain what the results are, not how they were produced—but are easy to understand by most users. Explaining systems show more diverse behavior depending on their motivation for providing explanations. Systems whose goal is to build trust in the data integration process tend to have lower coverage but higher understandability. In contrast, systems that focus on diagnosis and debugging typically produce explanations with higher coverage but lower understandability, as they target more educated users and domain experts.

We proceed to review these properties in more detail for explaining systems and highlight differences in their objectives and results.

Properties of explaining systems. Explaining systems provide evidence, in the form of explanations, that supports or refutes the results. Such evidence can be found in the input, the algorithm, or the combination of the two, and it is presented in various formats, such as examples, rules, or statistics. Discovering the evidence is fundamental to deriving causal explanations, but many explanation systems explore beyond this point in order to further improve their understandability. Some systems score the importance of the evidence and only report the top indicators; some systems aggregate the evidence and report the aggregated values instead of other details. In Table 4, we enumerate the properties of explanations for explaining systems along five dimensions, including the goal of the explanations, whether the explanations include details of the algorithm, the audience for the explanations, their format, and whether there is any post-processing associated with the discovered evidence.

Although all explaining systems provide causal explanations of their result, their coverage may be different due to differing explanation objectives. Systems that derive explanations for debugging purposes usually provide evidence not only from the input but also from the algorithms in order to reveal enough detail. However, this is not the case for external analytic tools, including mSeer, Spider, and TRAMP. With the goal of justifying the results and debugging the functionality of an existing data integration algorithm, these external analytic tools often conduct independent analysis over a data integration algorithm and reason about the correctness and the causes of the correctness of its results. Meanwhile, systems that use explanations to build trust typically do not reveal many details from the algorithm, and often only highlight evidence discovered from the input. The audience for the explanation correlates with the explanation objectives. Explanations meant for debugging usually target educated users, who have some background knowledge of the problem and algorithm, and may

System	Goal	Algorithm?	Audience	Format	Post-processing?
iMap [17]	Get feedback	Yes	Educated user	Visualization	None
S-Match [28, 53]	Build trust	No	Non-expert	Text	Summarization
	Debug	Yes	Domain expert	Rules	None
mSeer* [9]	Debug	No	Domain expert	Statistics	Summarization
Spider* [1, 12]	Debug	No	Domain expert	Rules	TopK
TRAMP* [27]	Debug	No	Domain expert	Tuples, rules	None
Dong & Srivastava [22]	Build trust	Yes	Non expert	Statistics	Summarization
Popat et al. [49]	Build trust	No	Non expert	Natural Language	Summarization
PGM [46]	Build trust	No	Non expert	Statistics	Summarization

Table 2: Analysis of properties of explaining systems. Highlighted systems (mSeer, Spider, and TRAMP) are external analytic tools; the others are a build-in component of the algorithms.

even be domain experts. In contrast, systems aimed at building trust are designed to be understandable even for non-expert users, with little background knowledge of the problem and algorithm. Even with the same explanation goal, systems may use different explanation formats, or post-process the discovered evidence to further improve the understandability of their explanations. In general, visualizing explanations may help users understand the explanations. In addition, condensing and summarizing the evidence may significantly reduce the volume of the explanations, and therefore improve their understandability.

5 Summary, next steps, and challenges for explaining data integration

In this paper, we presented a new classification of data integration systems by their explainability, and studied the systems and their characteristics within each class. The rich literature on data integration already demonstrates a strong and established focus on explanations. Over the past few years, data integration systems have developed a variety of explanations that serve diverse objectives. For example, human-in-the-loop systems frequently use non-causal explanations to illustrate the meaning of their results, and many explaining systems leverage causal explanations, associated with their results, to gain trust from the users. However, the portion of data integration systems that provide explanations, in particular causal explanations, is still small. For example, causal explanations are not an explicit objective of any existing approach in record linkage. The lack of explainability impedes the usability and evolution of data integration systems: humans cannot understand the results, provide feedback, or identify problems and suggest improvements.

We envision a future generation of data integration systems where *flexible* explanatory capabilities are a primary goal. Our study showed that there is a natural trade-off between coverage and understandability (Figure 2), and it is often driven by the underlying goals of the explanations (e.g., debugging, trust). Explanations that are used for debugging and diagnosis tend to be less understandable, however, they include more details of the data integration process. In contrast, non-causal explanations focus on explaining what the result is, and they typically offer little or no coverage of the process itself. The explanation goals also correlate with the target audience and the explanation type and content. To cater to varied users, uses, and goals, data integration systems need to support ***flexible, diverse, and granular explanation types***. Such systems would seamlessly transition from non-causal explanations facilitating interactions with non-experts, to causal explanations of the integration process facilitating debugging by domain experts. We identify the following challenges in achieving this vision.

Interactive explanations. To allow flexibility in navigating across explanations of different granularities and types, systems need to facilitate user interactions with the explanations. Users should be able to drill down,

generalize, and even ask for explanations of the explanations themselves.

From explainable to explaining. Explainable systems produce rich causal explanation content. However, these explanations are often not exploited, because they are typically complex, long, and hard or even impossible for humans to interpret and digest. External application tools could process, summarize, and abstract these explanations to different granularities, increasing the utility of explainable systems and essentially transforming them into explaining systems.

Evaluating understandability. Understandability is an important factor in formulating explanations, but it is a difficult factor to evaluate and measure. The fundamental challenge is that judgments of understandability are often subjective and could vary among people with different levels of background knowledge. System designers, knowledgeable users, and complete non-experts have different expectations from explanations. An explanation that is understandable and useful for one group may not be for another. To evaluate understandability in practice, researchers will have to rely on user studies with varied target users, ideally producing and generalizing guiding principles in explanation design.

Accuracy vs. explainability. Explaining data integration systems need to balance the tradeoff between accuracy and explainability. Sophisticated machine-learning approaches are widely used for solving data integration tasks. These methods achieve high accuracy, but are typically not explainable. Any solution in this domain will need to consider and balance these often conflicting objectives, to produce systems that are both effective and usable.

References

[1] B. Alexe, L. Chiticariu, and W.C. Tan. Spider: a schema mapping debugger. *VLDB*, 1179–1182, 2006.

[2] B. Alexe, L. Chiticariu, R.J. Miller, and W.C. Tan. Muse: Mapping understanding and design by example. *ICDE*, 10–19, 2008.

[3] D. Aumuller, H.H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. *SIGMOD*, 906–908, 2005.

[4] Z. Bellahsene, A. Bonifati, F. Duchateau, and Y. Velegrakis. On evaluating schema matching and mapping. *In Schema matching and mapping*, 253–291, 2011.

[5] J. Berlin, and A. Motro. Database schema matching using machine learning with feature selection. *In International Conference on Advanced Information Systems Engineering*, 452–466, 2002.

[6] P.A. Bernstein, and L.M. Haas. Information integration in the enterprise. *Communications of the ACM*, 51(9):72–79, 2008.

[7] J. Bleiholder, and F. Naumann. Data fusion. *ACM Computing Surveys (CSUR)*, 41(1):1, 2009.

[8] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. *In Visual Analytics Science And Technology*, 43–50, 2006.

[9] X. Chai, M. Sayyadian, A. Doan, A. Rosenthal, and L. Seligman. Analyzing and revising data integration schemas to improve their matchability. *PVLDB*, 1(1):773–784, 2008.

[10] Y. Chen, L. Chen, and C.J. Zhang. CrowdFusion: A Crowdsourced Approach on Data Fusion Refinement. *ICDE*, 127–130, 2017.

[11] G. Cheng, D. Xu, and Y. Qu. C3d+ p: A summarization method for interactive entity resolution. *Web Semantics: Science, Services and Agents on the World Wide Web*, (35):203–213, 2015.

[12] L. Chiticariu, and W.C. Tan. Debugging schema mappings with routes. *VLDB*, 79–90, 2006.

[13] I. Cruz, F. Antonelli, and C. Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *PVLDB*, 2(2):1586–1589, 2009.

[14] I. Cruz, C. Stroe, and M. Palmonari. Interactive user feedback in ontology matching using signature vectors. *ICDE*, 1321–1324, 2012.

[15] M. Das, S. Amer-Yahia, G. Das, and C. Yu. Mri: Meaningful interpretations of collaborative ratings. *PVLDB*, 4(11), 1063–1074,2011.

[16] J. Davis, I. Dutra, D. Page, and V. Costa. Establishing identity equivalence in multi-relational domains. *In Proc. ICIA*, 2005.

[17] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: discovering complex semantic matches between database schemas. *SIGMOD*, 383–394, 2004.

[18] X.L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1),550–561, 2009.

[19] X.L. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. *PVLDB*, 2(1):562–573, 2009.

[20] X.L. Dong, and F. Naumann. Data fusion: resolving data conflicts for integration. *PVLDB*, 2(2), 1654–1655, 2009.

[21] X. L. Dong, and D. Srivastava. Big data integration. *ICDE*, 1245–1248, 2013.

[22] X. L. Dong, and D. Srivastava. Compact explanation of data fusion decisions. *WWW*, 379–390, 2013.

[23] F. Doshi-Velez, and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

[24] Z. Dragisic, V. Ivanova, P. Lambrix, D. Faria, E. Jiménez-Ruiz, and C. Pesquita, C. User validation in ontology alignment. *In International Semantic Web Conference*,200-217, 2016.

[25] G.H.L. Fletcher, and C.M. Wyss. Data Mapping as Search. *EDBT*,95–111, 2006.

[26] M.A. Friedl, and C.E. Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[27] B.Glavic, G. Alonso, R.J. Miller, and L.M. Haas. TRAMP: Understanding the behavior of schema mappings through provenance. *PVLDB*, 3(1-2):1314–1325, 2010.

[28] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. *In European semantic web symposium*, 61–75, 2004.

[29] B. Golshan, A. Halevy, G. Mihaila, and W.C. Tan. Data integration: After the teenage years. *SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 101–106, 2017.

[30] D. Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA)*, 2017.

[31] L. Haas. Beauty and the beast: The theory and practice of information integration. *ICDT*, 28–43, 2007.

[32] M. Hernández, G. Koutrika, R. Krishnamurthy, L. Popa, and R. Wisnesky. HIL: a high-level scripting language for entity integration. *In Proceedings of the 16th international conference on extending database technology*, 549–560, 2013.

[33] E. Jimenez-Ruiz, B. Cuenca Grau, Y. Zhou, and I. Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. *ECAI*, 444–449, 2012.

[34] H. Kang, L. Getoor, B. Shneiderman, M. Bilgic, and L. Licamele. Interactive entity resolution in relational data: A visual analytic tool and its evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):999–1014, 2008.

[35] F.C. Keil, and R.A. Wilson. Explaining explanation. *Explanation and cognition*, 1–18, 2000.

[36] F.C. Keil. Explanation and understanding. *Annu. Rev. Psychol.*, 57:227–254, 2006.

[37] A. Kimmig, A. Memory, and L. Getoor. A collective, probabilistic approach to schema mapping. *ICDE*, 921–932, 2017.

[38] M. Lenzerini. Data integration: A theoretical perspective. *SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 233–246, 2002.

[39] B. Letham, C. Rudin, T.H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[40] L. Li, J. Li, and G. Hong. Rule-based method for entity resolution. TKDE, 27(1):250–263, 2015.

[41] G. Li. Human-in-the-loop data integration. *PVLDB*, 10(12):2006–2017, 2017.

[42] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with cupid. *VLDB*, (1):49–59, 2001.

[43] A. Meliou, S. Roy, and D. Suciu. Causality and explanations in databases. *PVLDB*, 7(13), 1715–1716, 2014.

[44] R.J. Miller, L.M. Haas, and M.A. Hernändez. Schema mapping as query discovery. VLDB, (2000):77–88, 2000.

[45] D.H. Ngo, and Z. Bellahsene. YAM++:(not) Yet Another Matcher for Ontology Matching Task. *In BDA: Bases de Données Avancées*, 2012.

[46] A.T. Nguyen, A. Kharosekar, M. Lease, and B.C. Wallace. An Interpretable Joint Graphical Model for Fact-Checking from Crowds. *AAAI*, 2018.

[47] N.F. Noy, and A.M. Mark. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies* 59(6):983–1024, 2003.

[48] E. Peukert, J. Eberius, and E. Rahm. A self-configuring schema matching system. *ICDE*, 306–317, 2012.

[49] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. *WWW*, 1003–1012, 2017.

[50] L. Qian, M.J. Cafarella, and H.V. Jagadish. Sample-driven schema mapping. *SIGMOD*, 73–84, 2012.

[51] S. Roy, D. Suciu. A formal approach to finding explanations for database queries. *SIGMOD*, 1579–1590, 2014.

[52] S. Sarawagi, and A. Bhamidipaty. Interactive deduplication using active learning. *SIGKDD*, 269–278, 2002.

[53] P. Shvaiko, F. Giunchiglia, P.P. Da Silva, and D.L. McGuinness. Web explanations for semantic heterogeneity discovery. *European Semantic Web Conference*, 303–317, 2005.

[54] S. Tejada, C.A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Special Issue on Data Extraction, Cleaning, and Reconciliation, Information Systems Journal*, 26(8):607–633, 2001.

[55] S. Tejada, C.A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. *SIGKDD*, 350–359, 2002.

[56] J. Tang, J. Li, B. Liang, X. Huang, X., Y. Li, and K. Wang. Using Bayesian decision for ontology mapping. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(4):243–262, 2006.

[57] S.E. Whang, and H. Garcia-Molina. Entity resolution with evolving rules. *PVLDB*, 3(1-2):1326–1337, 2010.

[58] S.E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.

[59] M. Wu, and A. Marian. A framework for corroborating answers from multiple web sources. *Information Systems*, 36(2):431–449, 2011.

[60] E. Wu, and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8), 553–564, 2013.

[61] X. Yin, and W. Tan. Semi-supervised truth discovery. *WWW*, 217–226, 2011.

[62] L.L. Yan, R.J. Miller, L.M. Haas, and R.Fagin. Data-driven understanding and refinement of schema mappings. *In ACM SIGMOD Record*, 30(2):485–496, 2001.

[63] C.J. Zhang, L. Chen, H.V. Jagadish, and C.C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, 2013.

[64] B. Zhao, B.I.P. Rubinstein, J. Gemmell, and J. Han. A bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 5(6):550–561, 2012.