Valentine in Action: Matching Tabular Data at Scale

Christos Koutras*
Kyriakos Psarakis*
Delft University of Technology
{c.koutras,k.psarakis}@tudelft.nl

George Siachamis
Delft University of Technology
g.siachamis@tudelft.nl

Andra Ionescu
Delft University of Technology
a.ionescu-3@tudelft.nl

Marios Fragkoulis
Delft University of Technology
m.fragkoulis@tudelft.nl

Angela Bonifati
Lyon 1 University
angela.bonifati@univ-lyon1.fr

Asterios Katsifodimos
Delft University of Technology
a.katsifodimos@tudelft.nl

ABSTRACT

Capturing relationships among heterogeneous datasets in large data lakes – traditionally termed schema matching – is one of the most challenging problems that corporations and institutions face nowadays. Discovering and integrating datasets heavily relies on the effectiveness of the schema matching methods in use. However, despite the wealth of research, evaluation of schema matching methods is still a daunting task: there is a lack of openly-available datasets with ground truth, reference method implementations, and comprehensible GUIs that would facilitate development of both novel state-of-the-art schema matching techniques and novel data discovery methods.

Our recently proposed Valentine is the first system to offer an open-source experiment suite to organize, execute and orchestrate large-scale matching experiments. In this demonstration we present its functionalities and enhancements: *i*) a scalable system, with a user-centric GUI, that enables the fabrication of datasets and the evaluation of matching methods on schema matching scenarios tailored to the scope of tabular dataset discovery, *ii*) a scalable holistic matching system that can receive tabular datasets from heterogeneous sources and provide with similarity scores among their columns, in order to facilitate modern procedures in data lakes, such as dataset discovery.

PVLDB Reference Format:
Christos Koutras, Kyriakos Psarakis, George Siachamis, Andra Ionescu, Marios Fragkoulis, Angela Bonifati, and Asterios Katsifodimos. Valentine in Action: Matching Tabular Data at Scale. PVLDB, 14(12): 2871-2874, 2021.
doi:10.14778/3476311.3476366

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/delftdata/valentine-system.

ACKNOWLEDGMENTS

This work has been partially funded by the H2020 project Opertus-Mundi No. 870228, and the ICAI "AI for Fintech Lab" project.

*Both authors contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476366

1 INTRODUCTION

Data integration is one of the most critical and difficult tasks in modern data science pipelines. Schema matching, one of the most important integration tasks, is the process of identifying correspondences between different schemata, and makes for an integral part of data discovery, data migration, and feature engineering. Schema matching has been studied extensively for the last twenty years, yielding a plethora of effective and efficient matching methods [8]. In addition, efforts have been made to make schema matching more easily employable through dedicated GUIs [1–3, 6, 9].

Despite the rich literature, the schema matching community lacks openly available implementations, while evaluation remains a difficult task due to limited datasets with ground truth and standardized baselines. Moreover, the existing schema matching systems that come with a GUI are oftentimes outdated, and provide the users with a limited selection of matching techniques and difficult to comprehend visualizations. In addition, they are not suitable for deployment at scale within a data lake as they do not offer the ability to evaluate methods with modern schema matching scenarios or to find matches among various datasets of a data lake, i.e. holistic matching. To alleviate these issues, we recently proposed Valentine [4], the first extensible open-source[1] experiment suite for schema matching that consolidates the state-of-the-art methods and alleviates the need for dedicated datasets by offering a fabrication process tailored to modern matching scenarios.

In this work, we demonstrate Valentine through a comprehensible GUI that enables easier-than-ever extensive experimental evaluation on schema matching for tabular data. On top of that, we expand its functionalities to holistic matching at scale in data lakes. In specific, the contributions of this work are summarized as follows:

- We offer a dataset fabricator that given a source tabular dataset is able to create numerous pairs of configurable characteristics and with respect to modern matching scenarios.
- We enable users to conduct comprehensive evaluation of schema matching methods on tabular data (both state-of-the-art and novel ones that can be easily integrated) and inspect and compare their effectiveness through easy-to-understand visualizations.
- We provide users with the ability to run holistic matching at scale on heterogeneous sources of tabular datasets and easily review results for verification of matchings.

[1] https://delftdata.github.io/valentine/

In the rest of the paper, we first provide an overview of Valentine [4] in Section 2. We then present the novel functionalities that Valentine brings and our proposed demonstration scenarios in Section 3, followed by a short discussion on related work (Section 4).

2 THE VALENTINE EXPERIMENT SUITE

In this section, we provide an overview of Valentine [4] while briefly recalling its main contributions and novelty.

2.1 Modern Schema Matching Scenarios

With Valentine we propose four modern schema matching scenarios tailored to the recent advances of dataset discovery (as cited in [4]). Specifically, we develop a taxonomy comprising of two main categories, *unionable* and *joinable* relations, which are further refined: *i*) Unionable relations might be *unionable* or *view-unionable* depending one whether there is a complete alignment between their columns or there is a slight discrepancy in their respective schemata, and *ii*) Joinable relations can be *joinable* or *semantically-joinable* depending on whether the instance values of the corresponding join columns are syntactically identical or not. This taxonomy guides the dataset fabrication and evaluation, which we describe in the following subsections.

2.2 A Dataset Fabricator for Schema Matching

Since datasets for evaluating schema matching methods are difficult to be found or curated, Valentine extends the approach of *e*Tuner [5] with the goal of fabricating datasets by splitting source tables in a systematic fashion. In order to create fabricated datasets tailored to the four schema matching scenarios (as described in Section 2.1) Valentine uses horizontal (for producing unionable pairs), vertical (for producing joinable pairs), or both kind of splits (for producing unionable/joinable pairs), following [5, 7]. In addition, dataset pairs might be injected with noise in their schemata and instances. Noise at the schema level is achieved with techniques such as abbreviating column names and dropping vowels, while the instance-level noise is added by injecting random typos into the instance sets (based on keyboard proximity) or by transforming numerical values according to their distribution.

2.3 Valentine's Schema Matching Methods

Valentine encompasses six state-of-the-art schema matching methods, which we either implemented from scratch (due to their unavailability) or integrated, as well as our own simple baseline. In detail, it includes[2]: *i*) three schema-based methods - Cupid, COMA and Similarity Flooding - which exploit only schema-level knowledge to capture relevance, *ii*) three instance-based methods - Distribution-based Matching, the instance-based flavor of COMA and our own Jaccard-Levenshtein baseline - which relies solely on data instances, and *iii*) two hybrid methods - SemProp and EmbDI - which leverage both schema and value information. Furthermore, in [4] we showed that these methods cover all types of matchers used in the recent dataset discovery literature, making Valentine suitable for employment in such pipelines.

[2]Citations of original papers can be found in [4]

2.4 Evaluation on Ranked Matches

In Valentine, evaluation of schema matching methods is conducted in a novel fashion, based on ranked matches. Specifically, given the matches that should hold (ground truth), and a ranked list of calculated similarities between all pairs of the corresponding columns between two datasets, we compute effectiveness in terms of $Recall@ground\ truth = \frac{\#\ of\ top\text{-}k\ relevant\ matches}{k}$, where $k = |ground\_truth|$. The above metric shows the quality of the ranking a method produces as it computes the top relevant results with respect to the ground truth. In essence, it reflects how helpful the output list is for a human who wants to assess only a limited list (e.g., a page) of top-k results.

2.5 Scalable Holistic Matching

Finding matches among all columns of tabular data in a data repository, or else *holistic matching*, can become a very time-consuming task, depending on the efficiency of the schema matching methods and the number and size of datasets. Therefore, designing a dedicated system should satisfy the following requirements: *i*) *scalability*, in terms of running schema matching methods without any significant runtime performance degradation while using more machines to handle the workload, and *ii*) *elasticity*, meaning the ability to change the number of machines in the system to meet the demand dynamically without any downtime. To fulfill those requirements, Valentine employs a container-based solution managed by Kubernetes[3], which is an open-source container orchestration system initially developed by Google. Kubernetes fulfills the requirements by providing easy-to-figure service discovery with load balancing and automated rollouts and rollbacks of deployment configurations allowing for seamless scalability and autoscaling. In addition, Valentine uses a task queue for every holistic matching job submitted by users, while it also stores results, i.e. similarities among all pairs of columns according to each algorithm, to a database. Data ingestion supports multiple types of inputs (e.g. .csv files, tables stored in databases etc.), in order to make Valentine suitable for finding matches among datasets coming from heterogeneous sources, as is the case in most data repositories.

3 DEMONSTRATION OVERVIEW

We now describe the main functionalities of Valentine and how we plan to allow the attendants to engage with it. First, we focus on how we make every component of Valentine (Section 2) easily accessible and applicable for evaluation of schema matching methods, by providing an intuitive GUI and a compact way of presenting experimental results. Then, we introduce Valentine's holistic matching capabilities for facilitating dataset discovery methods, backed by a system architecture for ingesting heterogeneous sources of tabular data and easily applying schema matching at scale.

3.1 Scenario 1: Schema Matching Evaluation

In Figure 1 we see the different frames of Valentine for fabricating datasets and evaluating schema matching methods. In the following we provide with details about the functionalities that each of them provide and how the user can interact with the system.

[3]https://kubernetes.io/

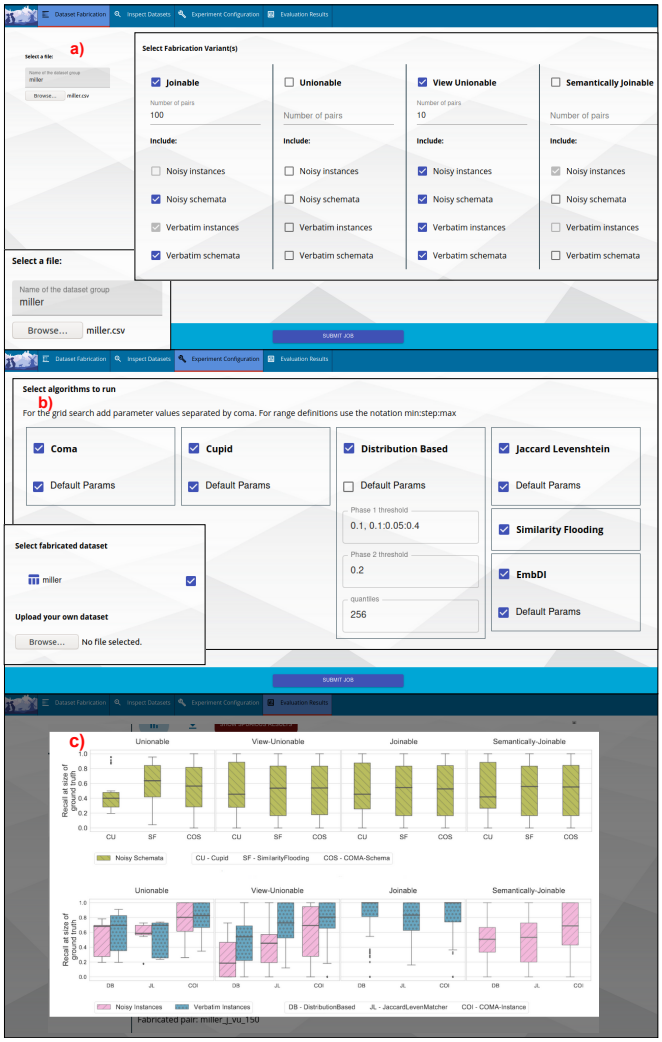

Figure 1: Screenshots from dataset fabrication (a), configuration of experiments (b) and presentation of findings (c).

Part 1: Dataset Fabrication. First, the users are given the option to fabricate *their own* schema matching dataset pairs in the dataset fabricator section shown in Figure 1a. There we see that they can upload any tabular dataset (in .csv format) containing the corresponding attribute names and instance sets. Next, they can *i*) choose the schema matching scenarios which the dataset pairs will adhere to (as discussed in Section 2.1), *ii*) decide whether they desire noise to be injected in some of the respective schemata and/or instances, *iii*) give the number of dataset pairs for each scenario, and *iv*) provide with a name for the group of datasets to be produced. By clicking the submit button, Valentine invokes the dataset fabricator with the given parameters and provides the user with the ability to inspect and download the fabricated dataset pairs in the form of a .zip file. In addition, Valentine automatically updates the list of available dataset groups with the newly fabricated pairs.

Part 2: Configuration of Experiments. In Figure 1b we see Valentine's frame for configuring schema matching experiments. On the left, the user can choose the dataset groups on which the schema

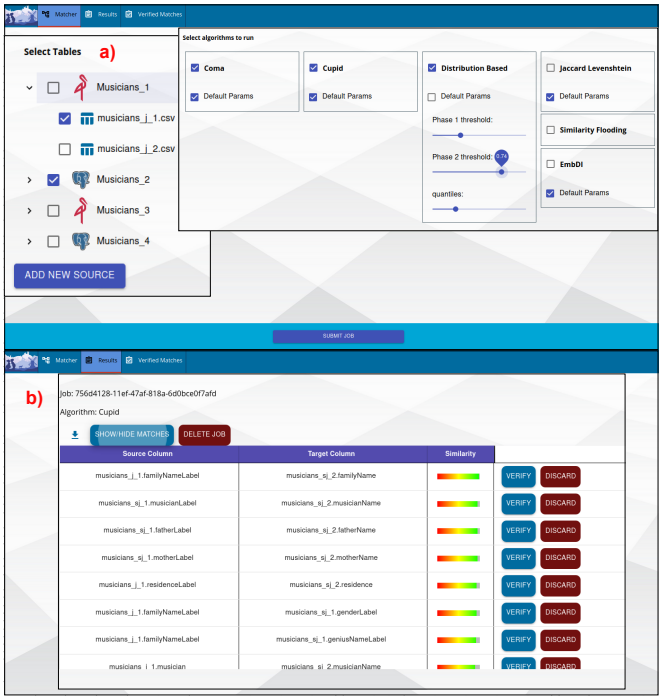

Figure 2: Screenshots of system configuration for holistic matching (a) and presenting results (b).

matching methods will run. The groups can either originate from the dataset fabricator or might be dataset pairs that the user uploaded. Next, users are able to decide which schema matching methods to apply, which can be either the ones that Valentine offers (Section 2.3) or methods that the user integrated into Valentine's framework through our defined input/output abstractions[4]. For each of these methods the user might choose specific parameters or specify ranges for Valentine to run a grid search on them. Finally, by clicking the submit button in the bottom, Valentine creates a job containing the specified configurations, which is added in a task queue and is given a specific identifier for the user to able to browse its results in the results frame.

Part 3: Presentation of Findings. Figure 1c shows the frame containing the results of the finished jobs. In particular, the user is presented with a list where each item is distinguished by its job identifier. Clicking on a particular item causes it to expand into a new view, containing a list with all dataset group names on which the user decided to run the experiments in the previous step. By clicking the *View Results* action button associated with each dataset category, Valentine visualizes the effectiveness results of each method in the form of box plots. This way Valentine is able to show the range of recall at ground truth values that each selected method exhibits across all pairs of the specified dataset group and categorized by the matching scenarios. Results for each method are shown for the parameters specified from the previous step or, in the case of grid search, for the parameters providing with the best recall at ground truth scores for each method and dataset pair. Therefore, users are presented with an intuitive visualization that

[4]More details in our Valentine repo https://github.com/delftdata/valentine

summarizes how well each method is able to rank correct matches at the top, while it also shows how consistent it is with respect to different matching scenarios. Moreover, the user is able to download detailed results, containing ranked matches for each dataset pair and method configuration, by clicking the *Download* icon.

3.2 Scenario 2: Holistic Matching at Scale

We enhance Valentine to extend the application of schema matching methods in the case of a data repository consisting of multiple heterogeneous sources of tabular datasets. In what follows we present how Valentine is able to scale holistic matching in multiple machines and the GUI that complements it for facilitating employment by the users.

Part 1: Executing Holistic Matching. Figure 2 shows the frames associated with Valentine's employment as a holistic matching system. First, the user is prompted to select the data sources and datasets to apply the schema matching methods on. Specifically, for each data source the user is given the option to select which of the included datasets should be regarded for execution by the system (Figure 2a). Furthermore, users can select which of Valentine's SotA schema matching methods to run on the specified datasets, while prescribing their configurations; to ease the execution for users that are not familiar with each method's tunable parameters, we also provide default configurations was in the original corresponding papers of the methods. By clicking the *Submit* button, a holistic matching job is queued with the specified configurations and is given an identifier.

Part 2: Result Presentation. Valentine's frame for presenting results of finished holistic matching jobs is depicted in Figure 2b. The users see a list with the different jobs for which the respective holistic matching with the given configurations has succeeded in finishing. It is possible to immediately view (or hide) the list of matches provided by each corresponding schema matching method that the user selected in the previous step, by clicking the *Show/Hide Matches* button. In detail, matches are displayed between every pair of columns among all datasets and ordered by each method's similarity measure which is indicated with a gradient color bar moving from red to green as the similarity increases. To not overwhelm the users, Valentine paginates the resulting proposed matches. In addition, the *Download Results* action button allows users to receive a .csv file containing the ranked list of similarities between all pairs of columns coming from the repository's selected datasets for each schema matching algorithm employed. These results can be then fed into any dataset discovery pipeline, making Valentine an easily deployed schema matching component and a very reliable one, since it consolidates the best of schema matching efforts.

Part 3: Result Verification. Lastly, Valentine enables manual verification of match pairs. Users can verify or discard match pairs by clicking the corresponding *Verify* / *Discard* buttons as shown in Figure 2b. In order to facilitate verification of matches, users can click on a match pair and inspect a representative sample of instance sets drawn from the inspected columns. Verified matches are then stored in a separate database, which can be regarded as holding the ground truth of matches for the specific datasets that Valentine was applied upon. Therefore, Valentine could be deployed by data scientists in order to facilitate and accelerate capturing matches among columns of different datasets.

4 RELATED WORK

Schema Matching Benchmarks. To the best of our knowledge, XBenchMark [3] is the only known attempt in the literature for providing with a tool to run schema matching experiments. Nonetheless, it provided a very limited set of matching methods and datasets, while it supported only XML data. On the contrary, Valentine offers a wide gamut of SotA schema matching methods which cover the needs of modern applications such as dataset discovery, a dataset fabricator which alleviates the lack of publicly available datasets, and supports the most prevalent format of data, i.e. tabular data. Furthermore, Valentine is open-source and extensible, with respect to new schema matching methods and datasets.

Schema Matching Systems with a GUI. There have been multiple schema matching systems proposed that come with a GUI. Clio [6] used a naive Bayes classifier on a character level for matching categorical data and quantile based classification methods for matching numerical data. The COMA 3.0 Community Edition based on COMA [2], encompasses instance and schema-based simple matchers which the users can select for matching a pair of schemata. In a similar manner, BizTalk [1] was proposed, with the exception of improving GUI to help the users understand the proposed matches. Schema-based linguistic-based matchers are incorporated to find matches in OpenII's Harmony [9], which exploited also available textual definitions of schema elements. While each of the aforementioned systems has helped to progress the field of schema matching, they come with certain limitations: *i*) execution of matching only between certain pairs of schemata, and not on large numbers of pairs at once, *ii*) scalability issues, *iii*) limited and not up-to-date collections of matchers, *iv*) inability to execute holistic matching, and *v*) outdated and difficult to comprehend visualizations and presentations of results in their respective GUIs. Our proposed Valentine overcomes all these issues, and provides a user-centric intuitive GUI to execute schema matching at scale.

REFERENCES

[1] Philip A Bernstein, Sergey Melnik, and John E Churchill. 2006. Incremental schema matching. In *VLDB*, Vol. 6. Citeseer, 1167–1170.

[2] Hong-Hai Do and Erhard Rahm. 2002. COMA: a system for flexible combination of schema matching approaches. In *VLDB*.

[3] Fabien Duchateau, Zohra Bellahsene, and Ela Hunt. 2007. XBenchMatch: a Benchmark for XML Schema Matching Tools. In *VLDB*.

[4] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery (to be presented in IEEE ICDE 2021). arXiv:2010.07386 [cs.DB]

[5] Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon S. Rosenthal. 2007. ETuner: Tuning Schema Matching Software Using Synthetic Scenarios. *VLDBJ* 16, 1 (2007), 97–122.

[6] Renée J Miller, Mauricio A Hernández, Laura M Haas, Lingling Yan, CT Howard Ho, Ronald Fagin, and Lucian Popa. 2001. The Clio project: managing heterogeneity. *ACM Sigmod Record* 30, 1 (2001), 78–83.

[7] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. In *VLDB*.

[8] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDBJ* 10, 4 (2001), 334–350.

[9] Len Seligman, Peter Mork, Alon Halevy, Ken Smith, Michael J Carey, Kuang Chen, Chris Wolf, Jayant Madhavan, Akshay Kannan, and Doug Burdick. 2010. Openii: an open source information integration toolkit. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 1057–1060.