Loki: Streamlining Integration and Enrichment

William Spoth
wmspoth@buffalo.edu
University at Buffalo

Poonam Kumari
poonamku@buffalo.edu
University at Buffalo

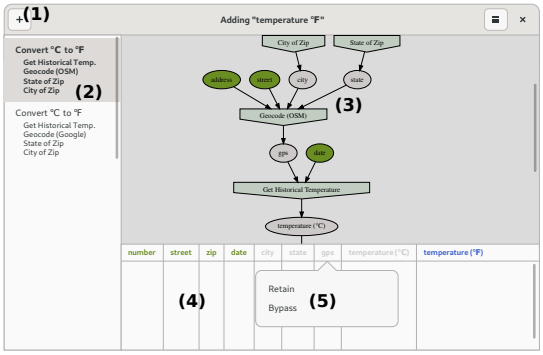
Oliver Kennedy
okennedy@buffalo.edu
University at Buffalo

Fatemeh Nargesian
fnargesian@rochester.edu
University of Rochester

(0) The user opens Loki with a new dataset.

(1) The user selects an attribute to add

(2) Valid, minimal sequences of mappings needed to reach the attribute are computed and shown to the user.



(3) Loki vizualizes the transformation needed to obtain the target attributes.

(4) Data examples help the user to see the transformation flow.

(5) The user can retain (add) intermediate attributes into the target schema, or request that an attribute or mapping step be excluded from the search.

ABSTRACT

Data scientists frequently transform data from one form to another while cleaning, integrating, and enriching datasets. Writing such transformations, or "mapping functions" is time-consuming and often involves significant code re-use. Unfortunately, when every dataset is slightly different from the last, finding the right mapping functions to re-use can be just as difficult as starting from scratch. In this paper, we propose "Link Once and Keep It" (Loki), a system that consists of a repository of datasets and mapping functions. It uses this repository to relate new datasets to datasets it already knows about to help data scientists to quickly locate and re-use mapping functions built for previous datasets. Loki represents a first step towards building and *re*-using repositories of *domain-specific* data integration pipelines.

HILDA'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-8022-5/20/06…$15.00
https://doi.org/10.1145/3398730.3399198

KEYWORDS

data integration, data lakes, table-union

ACM Reference Format:
William Spoth, Poonam Kumari, Oliver Kennedy, and Fatemeh Nargesian. 2020. Loki: Streamlining Integration and Enrichment. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3398730.3399198

1 INTRODUCTION

Translating data values from one representation into another is a common task in data science. For example, in data integration, data scientists map multiple datasets into a common representation, starting with a set of correspondences between attributes. A correspondence can be a simple one-to-one mapping or a translation between units (celsius vs farenheit), measurement techniques (axial vs oral), formats (2-letter country code vs full name), and much more. Similarly, in data enrichment, data scientists enhance a dataset by mapping in supplemental attributes or performing table joins [10] to augment the dataset, for example by supplementing location data with population statistics or weather information.

Unfortunately, every new dataset brings with it a minefield of special cases, making it difficult to *fully* automate data mapping. In this paper, we present "Link Once and Keep It" (Loki), which takes a lighter-weight, human-in-the-loop approach: Instead of automatically mapping attributes, Loki

helps the data scientist find and re-use mapping functions she created in the past. For example, once she writes a function mapping zip (postal) codes to states, she should be able to re-use that function on any datset with a zip code attribute.

Example 1.1. Alice the environmental health scientist is analyzing data gathered from several dozen research labs. While the data being collected by each lab is similar, each is subtly different. For example, one lab might measure blood iron with a serum iron test, while another measures it through a serum ferritin test. These measurements are not directly comparable, but can be easily translated. Alice has already mapped the datasets into a common schema, but needs to bring in data from a brand new lab to her analysis. Normally, she would either have to manually write the code to map this new dataset in, or manually search through her prior work to find mapping code that she can re-use.

We formally define "*type-mapping search*" as the problem of building (from a repository of mappings) a composite mapping that translates attributes from one table to an attribute of another table. Loki treats type mappings as first-class citizen mappings [1]. Loki assumes that mappings defined on a set of attributes also apply to attributes that are similar or "unionable." To this end, Loki builds on prior work in *table-union search* [9], which allows users to search a repository for tables with unionable attributes. Our first contribution involves supporting numerical attributes in table-union search, through the Kolmogorov-Smirnoff test [7]. However, the source data may also need translation (e.g., °F to °C) or enrichment (e.g., demographic information for spatial data). Our second contribution is a baseline algorithm for building such sequences of mappings by exploring the combinatorial space of possible sequences of mappings in a forward and greedy manner. To this end, Loki represents a first step towards building and *re*-using repositories of *domain-specific* data integration pipelines.

2 PROBLEM DEFINITION

Semantic types [3] identify contextual details of an attribute, like measurement units (Celsuis), or its role in a datasets, like a patient's blood iron concentration. To integrate two datasets, their attributes need to be mapped into schemas of matching semantic types. A *type mapping* is a query that transforms a collection of attributes in a collection of source tables into the semantic type of one attribute of a target table or schema.

Example 2.1. Alice's new dataset includes subject information like street address, city, body temperature in Fahrenheit, and blood iron concentration taken using a serum feratin test. Her analysis needs demographic information, body temperature in Celsius, and blood iron concentration on the serum iron scale. She poses three type-mapping queries, one for each target attribute. Loki finds a mapping she previously used to translate temperatures, and assembles a new mapping from previously developed mappings for geocoding street addresses, and for looking up demographics given GPS coordinates. She has not yet written the required iron concentration mapping, and will need to do so now.

We assume that we are given a universal schema and a set of type mappings (e.g., created manually during prior tasks). Data tables in the Loki repository are subsets of this universal schema.

Definition 2.2 (Loki Repository). A Loki repository $\langle \mathcal{A}, \mathcal{R}, \mathcal{M} \rangle$ is a set of attributes $\mathcal{A} = \{A_1, \ldots, A_N\}$ called the repository's schema, a set of relations $R \in \mathcal{R}$ with schemas drawn from $\mathcal{A}$ (i.e., $sch(R) \subseteq \mathcal{A}$), and a set of type-mappings $m \in \mathcal{M} : \langle S_1, \ldots, S_k \rangle \rightarrow T$ from a tuple of source attributes ($S_i \in \mathcal{A}$) to a target attribute ($T \in \mathcal{A}$).

Our goal is to map the attributes of some source relation into a target schema $\mathbb{T} \subseteq \mathcal{A}$. Without loss of generality, we consider each target attribute $T \in \mathbb{T}$ individually: For each, we need to find a sequence of type-mappings that can be iteratively applied to the source to obtain the target.

Definition 2.3 (Valid Mapping / Sequence). A mapping $m :< S_1, \ldots, S_k >\!\rightarrow T$ is valid for a schema $\mathbb{S} \subseteq \mathcal{A}$ if and only if the mapping's source attributes are a subset of the schema (i.e., $\forall i \in [k] : S_i \in \mathbb{S}$). We call sequence of mappings $M = \{m_1, \ldots, m_\ell\}$ valid for $\mathbb{S}$ if the source attributes of each element of the sequence includes only attributes from the schema and any preceding target attributes (i.e., M is valid for $\mathbb{S}$ iff $\forall i \in [\ell] : m_i$ is valid for $\mathbb{S} \cup \left(\bigcup_{1 \leq j < i} T_j\right)$)

A type-mapping search attempts to find a valid sequence of mappings that ends with a specified target attribute.

Definition 2.4 (Type-Mapping Search Problem). Let $\langle \mathcal{A}, \mathcal{R}, \mathcal{M} \rangle$ be a Loki repository. A type-mapping search query is a pair $q = \langle \mathbb{S}_q, T_q \rangle$ of source attributes $\mathbb{S}_q \subseteq \mathcal{A}$ and target attribute $T_q \in \mathcal{A} - \mathbb{S}_q$. A sequence of mappings $M = \{m_1, \ldots, m_\ell\}$ (where $m_i \in \mathcal{M}$) answers the query if M is valid for S_q and $T_q = T_\ell$. M is a minimal answer if no element can be removed while preserving both properties.

In real-world use, source attributes may not yet exist in Loki's schema (i.e. $\mathbb{S}_q \not\subseteq \mathcal{A}$). Thus, to answer a type-mapping search query, we need to map $\mathbb{S}_Q$ into $\mathbb{A}$, and find a minimal answer.

3 SYSTEM OVERVIEW

We subdivide the high-level goals of type mapping search into two major challenges: (i) *Attribute Resolution*: Mapping the source attributes of $\mathbb{S}_q$ into the repository's universal

schema $\mathcal{A}$, (ii) *Type-Mapping Search*: Finding a valid mapping sequence that can be iteratively applied to $\mathbb{S}_q$ to construct target attributes. Loki considers two kinds of mappings between attributes: (i) *declared* mappings explicitly translating between attributes, and (ii) *inferred* mappings between unionable attributes.

Example 3.1. Alice's new dataset, illustrated in Figure 1, includes street address (S_1), city (S_2), body temp. in °F (S_3), and serum feratin (S_4). She needs average income (T_1), body temp. in °C (T_2), and serum iron (T_3). Loki determines that S_3 is unionable with a previously imported temperature (A_3), adding an inferred mapping through which Alice's existing temperature conversion becomes valid. Similarly, inferred mappings relate S_1 and S_2 to the geocoder's expected inputs. Loki had already inferred mappings between geospatial coordinates in existing datasets (A_6 and A_7), connecting the demographic lookup. There is no mapping for the serum feratin scale (S_4), but Loki mistakenly equates it with a hemoglobin test for iron (A_5), which has mappings both with test temperature (A_4) and without. Loki presents Alice with potential mappings, and she decides whether to re-use or adapt an existing mapping, or whether to write a new one.

3.1 Attribute Resolution

Attribute resolution discovers pairs of *unionable* attributes A_m and A_n and creates inferred mappings between them. NL-Unionability has previously been studied [9] in the context of non-numerical (i.e., text-typed) attributes through the natural language association of data values. To evaluate NL-unionability, an attribute is represented as the empirical mean and covariance of the embedding vectors of its values. These vectors are generated such that the semantic similarity of two data values is translated into the similarity of their embedding vectors, which can be evaluated by angular distance measures. NL-unionability assumes two attributes are unionable if they are samples of the same Gaussian distribution on embedding vectors. Hotelling's T-squared test gives the NL-unionability score using the mean and covariance of the embedding of pairs of non-numerical attributes [9]. We now generalize unionability to numerical attributes.

Unionability of Numerical Attributes. To evaluate unionability of two numerical attributes, we use the two-sample Kolmogorov-Smirnov test over each pair of attributes. Intuitively, the test judges whether two samples of data are the same by estimating the likelihood that differences between their empirical distribution functions are due to random chance. For two numerical attributes A_n and A_m, we derive a KS-unionability score $U(A_n, A_m) \in [0, 1]$ as follows. We first compute a distance score $D(A_n, A_m)$ for the two attributes as the maximum distance between their empirical distribution functions F_n and F_m respectively: $D(n, m) = max_x(F_n(x) - F_m(x))$ If the samples are drawn from the same distribution, we expect the distance score to asymptotically approach zero for larger samples. Accordingly, the Kolmogorov-Smirnov test sets a confidence threshold α based on an attributes sample size using $\alpha = \frac{1.63}{n^{\frac{1}{2}}}$. Two samples are distinct if $D(A_n, A_m) < \frac{1}{\sqrt{|A_n|}} \cdot \sqrt{-ln(\frac{\alpha}{2}) \cdot \frac{1+\frac{|A_n|}{|A_m|}}{2}}$ [7]. To define the unionability score, we invert the formula for α to compute the probability (p-value) that A_n and A_m are random samples from the same distribution, given $D(n, m)$.

$$U(A_n, A_m) = e^{\frac{-2 \cdot |A_n||A_m| \cdot D(A_n, A_m)^2}{|A_m|+|A_n|}}$$

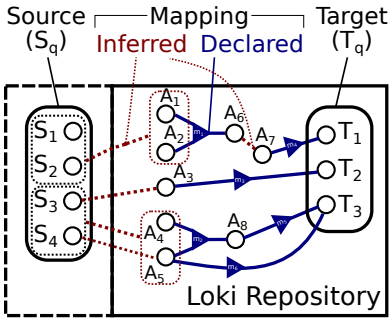


Figure 1: Example query to map $\mathbb{S}_q$ to targets T_1, T_2, T_3

3.2 Type-Mapping Search

An inferred mapping $\langle S_1, \ldots, S_k \rangle \rightarrow_m \langle A_1, \ldots, A_k \rangle$ resolves attributes $S_1, \ldots, S_k$ to their unionable attributes $A_1, \ldots, A_k$. Suppose X and Y are sets of attributes. The set X is mapped to Y if we can find one-to-one mappings between their attributes. The *resolution score* of a one-to-one mapping $\langle A_i \rangle \rightarrow_m \langle A_j \rangle$, namely $res(m)$, is the unionabiliy score $U(A_i, A_j)$. A mapping $X \rightarrow_m Y$ is defined as an alignment $C : X \rightarrow Y$ that is a set of one-to-one mappings $\langle S_i \rangle \rightarrow_{m'} \langle A_j \rangle$, where $S_i \in X$ and $A_j \in Y$. We evaluate an alignment C by the joint unionability of mapped attributes in the alignment.

$$U_C(X, Y) = \Pi_{S_i \rightarrow A_j \in C} U(A_i, A_j)$$

Given attribute sets X and Y, we define the resolution score of mapping $X \rightarrow_m Y$ ($res(m)$), as the unionability of the alignment that has the highest unionability, $res(m) = max_C(U_C)$. Note that declared mappings do not need attribute resolution, so we fix their resolution scores to be one. We define the resolution score of mapping sequence $M = \{ m_1, \ldots, m_\ell \}$ as the product of individual scores (i.e., $res(M) = \Pi_{m_i \in M} res(m_i)$). Given a Loki repository $\langle \mathcal{A}, \mathcal{R}, \mathcal{M} \rangle$ and a query $q = \langle \mathbb{S}_q, T_q \rangle$, our goal is to find the valid mapping $M = \{m_1, \ldots, m_\ell\}$ from $\mathbb{S}_q$ to T_q with the highest mapping score. As shown in Figure 1, an answer starts with inferred mappings from $\mathbb{S}_q$ and ends with target T_q

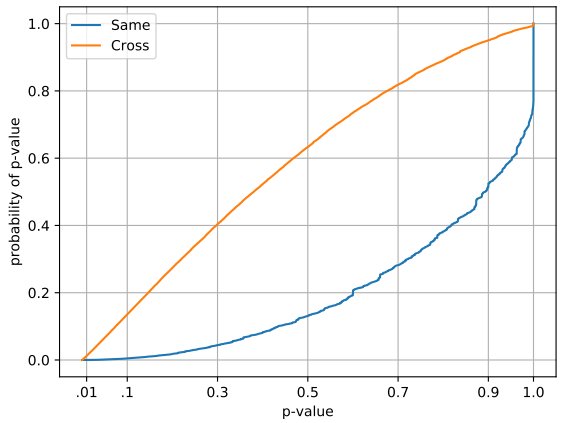


Figure 2: p-values for each pair of columns. Higher values indicate greater similarity (i.e., H_0 is that columns are sampled from the same distribution).

We solve the type mapping search problem with a greedy forward search algorithm. The search starts with $\mathbb{N} = \{\mathbb{S}_q\}$ and at each step, it ranks mappings on any subset $N \subset \mathbb{N}$ by resolution scores and applies the mapping with highest resolution score on $\mathbb{N}$. Every time a valid mapping $m_i : \langle A_1, \ldots, A_k \rangle \rightarrow A_j$ is applied we update $\mathbb{N} = \mathbb{N} \cup A_j$. Suppose at step t the sequence of mappings is $\{m_1, \ldots, m_t\}$. If no valid mapping can be applied on $\mathbb{N}$ and $T_q \notin \mathbb{N}$, the algorithm backtracks to $\{m_1, \ldots, m_{t-1}\}$, updates $\mathbb{N}$ accordingly, and applies the valid mapping with the second highest mapping score and so on. The search terminates when search succeeds (i.e., $T_q \in \mathbb{N}$), a maximum number of search steps have been taken, or the search space is exhausted.

To speed up search we leverage heuristics based on existing workloads and type mappings in Loki repository. The first heuristic is a threshold on the attribute resolution scores. We only consider the attribute mappings that have unionability score above a threshold. Groups of attributes co-occur as inputs to mappings in successful workloads. The second heuristic is to form *hyper attributes* which are partitions of attributes in $\mathcal{A}$ and $\mathbb{S}_q$ that often form inputs of mappings collectively. Another source of complexity for the search is the large number of sequences of mappings to be considered. Some declared mappings are often applied in a sequence. These sequences can be reused for later workloads. The second heuristic is *hyper mappings*; sequences of mappings with high resolution scores, extracted from successful search iterations, and applied atomically. For example, in Figure 1 the sequence $\{m_2, m_5\}$ is a hypermapping.

4 EVALUATION

As a proof of concept, we implemented KS-unionability described in Section 3.1. We evaluated the test on a collection of 53,953 numerical attributes with at least 50 distinct values drawn from 499 of the open datasets used in [9]. Figure 2 plots a CDF of p-values (i.e., $U(A_1, A_2)$) for KS tests for two trials. The **Same** trial uses a synthetic ground-truth dataset created by randomly splitting every attribute column into two parts A_1 and A_2. Nearly 30% of attributes are an exact match to themselves and 70% of attributes have a unionability score of 0.7 or higher. The **Cross** trial tests the corresponding false positive rate, comparing 22 million distinct attribute pairs sampled uniformly at random from the numerical test attributes. Virtually no attributes are an exact match, roughly 80% of attributes have a unionability score of 0.7 or lower, and unionability scores are generally significantly lower than for the true matches. This indicates that KS-unionability is a reliable metric for finding matching attributes.

5 RELATED WORK

Attribute Resolution - For attribute resolution, Loki builds on existing work in Query Unionability [9]. Prior work focuses primarily on unionability of string and enumerable data, using word embeddings to discover related attributes, while Loki adds support for numerical attributes. Resolving numerical attributes has been previously explored [2, 5], though prior work relies on prose to contextualize the numbers. Attribute resolution is also analogous to *semantic type detection* [3], although Loki considers pairwise similarity between attributes rather than trying to map attributes into a common taxonomy [4]. This difference allows Loki to more easily adapt to new and unexpected domains.

Type-Mapping Search - Type mapping can be thought of as a generalized form of Joinability search [11], which has also been explored in the context of touch [6] and natural language interfaces [8]. Moreover, existing linkage navigation work are concerned with single attribute joins [12]. The most significant difference from this prior work is the sheer size of the search space — any valid sequence of mappings is fair game. By contrast, prior scalable joinability searches typically only need to traverse one hop [6, 11] or (e.g., [8] only searches the foreign key references of a schema).

6 FUTURE WORK

Our next step is to incorporate source priors into resolution scores. Heuristics such as increasing attribute weighting for mappings within base files increase our ability to expand from the single table assumption of S_q to include table joins, and multiple $\mathcal{T}_q$. While attribute search is already parallelizable, for larger data sources we can incorporate dynamic programming tactics in combination with batch q.

Scaling Loki for extended use will require consolidating overlapping attributes either through human interaction or statistical methods, limiting repository growth over time. Additionally, we look to incorporate user uncertainty into resolution scores to prevent repository dilution as our knowledge base grows.

7 CONCLUSION

In this paper we presented Loki, a system to automate mapping and pipeline reuse through learned human interactions. Capable of improving accuracy of additional downstream systems such as table-union, Loki is a welcomed addition to any data scientists toolbox.

ACKNOWLEDGMENTS

This work is supported in part by NSF Awards ACI-1640864 and IIS-1750460. Opinions, findings and conclusions expressed in this material are those of the authors and may not reflect the views of the NSF.

REFERENCES

[1] P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A vision of management of complex models. *SIGMOD Rec.*, 29(4):55–63, 2000.

[2] V. T. Ho, Y. Ibrahim, K. Pal, K. Berberich, and G. Weikum. Qsearch: Answering quantity queries from text. In *ISWC (1)*, volume 11778 of *Lecture Notes in Computer Science*, pages 237–257. Springer, 2019.

[3] M. Hulsebos, K. Z. Hu, M. A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. A. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *KDD*, pages 1500–1508. ACM, 2019.

[4] Y. Ibrahim, M. Riedewald, and G. Weikum. Making sense of entities and quantities in web tables. In *CIKM*, pages 1703–1712. ACM, 2016.

[5] Y. Ibrahim, M. Riedewald, G. Weikum, and D. Zeinalipour-Yazti. Bridging quantities in tables and text. In *ICDE*, pages 1010–1021. IEEE, 2019.

[6] L. Jiang, M. Mandel, and A. Nandi. Gesturequery: A multitouch database query interface. *PVLDB*, 6(12):1342–1345, 2013.

[7] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition.* Addison-Wesley, 1981.

[8] F. Li and H. V. Jagadish. Nalir: an interactive natural language interface for querying relational databases. In *SIGMOD Conference*, pages 709–712. ACM, 2014.

[9] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *PVLDB*, 11(7):813–825, 2018.

[10] Y. Wang and Y. He. Synthesizing mapping relationships using table corpus. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 1117–1132, 2017.

[11] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller. JOSIE: overlap set similarity search for finding joinable tables in data lakes. In *SIGMOD Conference*, pages 847–864. ACM, 2019.

[12] E. Zhu, K. Q. Pu, F. Nargesian, and R. J. Miller. Interactive navigation of open data linkages. *PVLDB*, 10(12):1837–1840, 2017.