Feedback Driven Improvement of Data Preparation Pipelines

DOI:
10.1016/j.is.2019.101480

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):
Konstantinou, N., & Paton, N. (2019). Feedback Driven Improvement of Data Preparation Pipelines. *Information Systems*, *92*(0), [101480]. https://doi.org/10.1016/j.is.2019.101480

Published in:
Information Systems

Feedback Driven Improvement of Data Preparation Pipelines

Nikolaos Konstantinou, Norman W Paton

Department of Computer Science, University of Manchester, Oxford Road, Manchester, UK

Abstract

Data preparation, whether for populating enterprise data warehouses or as a precursor to more exploratory analyses, is recognised as being laborious, and as a result is a barrier to cost-effective data analysis. Several steps that recur within data preparation pipelines are amenable to automation, but it seems important that automated decisions can be refined in the light of user feedback on data products. There has been significant work on how individual data preparation steps can be refined in the light of feedback. This paper goes further, by proposing an approach in which feedback on the correctness of values in a data product can be used to revise the results of diverse data preparation components. The approach uses statistical techniques, both in determining which actions should be applied to refine the data preparation process and to identify the values on which it would be most useful to obtain further feedback. The approach has been implemented to refine the results of matching, mapping and data repair components in the VADA data preparation system, and is evaluated using deep web and open government data sets from the real estate domain. The experiments have shown how the approach enables feedback to be assimilated effectively for use with individual data preparation components, and furthermore that synergies result from applying the feedback to several data preparation components.

Keywords: data preparation, data wrangling, extract transform load,

Email addresses: nikolaos.konstantinou@manchester.ac.uk (Nikolaos Konstantinou), norman.paton@manchester.ac.uk (Norman W Paton)

dataspace, feedback

1. Introduction

Data preparation is the process of transforming data from its original form into a representation that is more appropriate for analysis. In data warehouses, data preparation tends to be referred to as involving an *Extract Transform Load* (ETL) process [1], and for more *ad hoc* analyses carried out by data scientists may be referred to as *data wrangling* [2]. In both cases, similar steps tend to be involved in data preparation, such as: *discovery* of relevant sources; *profiling* of these sources to better understand their individual properties and the potential relationships between them; *matching* to identify the relationships between source attributes; *mapping* to combine the data from multiple sources; *format transformation* to revise the representations of attribute values; and *entity resolution* to identify and remove duplicate records representing the same real world object.

This is a long list of steps, each of which can potentially involve data engineers: (i) deciding which data integration and cleaning operations to apply to which sources; (ii) deciding the order of application of the operations; and (iii) either configuring the individual operation applications or writing the rules that express the behaviour to be exhibited. Although there are many data preparation products, and the market for data preparation tools is estimated to be *$2.9* billion [3], most of these products are essentially visual programming platforms, in which users make many, fine-grained decisions. The consequence of this is that data preparation is typically quoted as taking 80% of the time of data scientists, who would prefer to be spending their time on analysing and interpreting results[1].

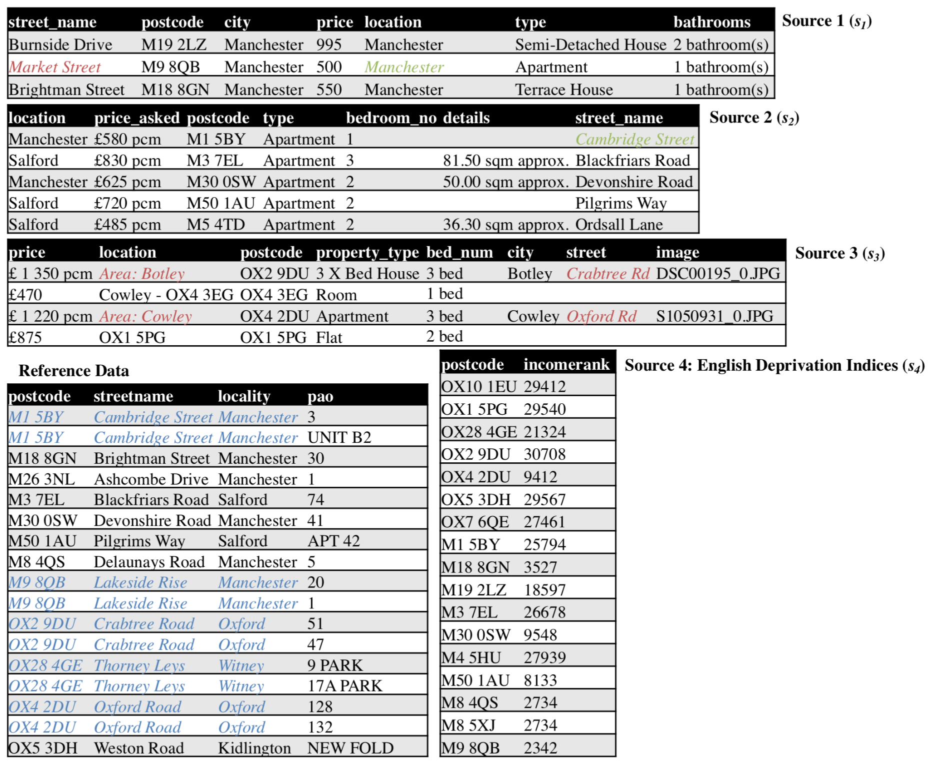
The high cost of data preparation has been recognised for a considerable period. For example, research into dataspaces [4] proposed a pay-as-you-go ap-

[1]https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#33d344256f63

proach to data integration, in which there was an initial and automated bootstrapping phase, which was followed by an incremental improvement phase in which the user provided feedback on the data product. This gave rise to a collection of proposals for pay-as-you-go data integration and cleaning platforms [5], which in turn led to proposals for the use of crowds as a possible source of feedback [6]. This research provided experience with pay-as-you-go data management, without leading to many end-to-end systems; for the most part, feedback was obtained for a particular task (e.g. mapping selection, entity resolution) and used for that task alone. This was alright, but collecting feedback on lots of individual components is itself expensive, and thus not especially helpful for the complete, many-step data preparation process.

This, therefore, leaves open the question as to how to make a multi-step data preparation process much more cost effective, for example through automation and widespread use of feedback on data products. There are now some results on automating comprehensive data preparation pipelines. For example, in Data Tamer [7], machine learning is used to support activities including the alignment of data sets and instance level integration through entity resolution and fusion. In some respects Data Tamer follows a pay-as-you-go approach, as the training data used by the learning components is revised in the light of experience. Furthermore, in VADA [8, 9], a collection of components (for matching, mapping generation, source selection, format transformation and data repair) are orchestrated automatically over data sources, informed by supplementary instance data drawn from the domain of the target schema [10]. However, to date, feedback has only been applied in VADA to inform the selection of mappings.

In this paper we investigate how feedback on the data product that results from the multi-component data preparation process in VADA can be used to revise the results of multiple of these wrangling components in a well-informed way. In particular, given feedback on the correctness of tuples in the data product, a feedback assimilation strategy explores a set of hypotheses about the reason for problems with the result. The statistical significance of these hypotheses is then tested, giving rise to the generation of a revised data integration



Source 1 (s_1)

street_name	postcode	city	price	location	type	bathrooms
Burnside Drive	M19 2LZ	Manchester	995	Manchester	Semi-Detached House	2 bathroom(s)
Market Street	M9 8QB	Manchester	500	*Manchester*	Apartment	1 bathroom(s)
Brightman Street	M18 8GN	Manchester	550	Manchester	Terrace House	1 bathroom(s)

Source 2 (s_2)

location	price_asked	postcode	type	bedroom_no	details	street_name
Manchester	£580 pcm	M1 5BY	Apartment	1		*Cambridge Street*
Salford	£830 pcm	M3 7EL	Apartment	3	81.50 sqm approx.	Blackfriars Road
Manchester	£625 pcm	M30 0SW	Apartment	2	50.00 sqm approx.	Devonshire Road
Salford	£720 pcm	M50 1AU	Apartment	2		Pilgrims Way
Salford	£485 pcm	M5 4TD	Apartment	2	36.30 sqm approx.	Ordsall Lane

Source 3 (s_3)

price	location	postcode	property_type	bed_num	city	street	image
£ 1 350 pcm	*Area: Botley*	OX2 9DU	3 X Bed House	3 bed	Botley	*Crabtree Rd*	DSC00195_0.JPG
£470	Cowley - OX4 3EG	OX4 3EG	Room	1 bed			
£ 1 220 pcm	*Area: Cowley*	OX4 2DU	Apartment	3 bed	Cowley	*Oxford Rd*	S1050931_0.JPG
£875	OX1 5PG	OX1 5PG	Flat	2 bed			

Reference Data

postcode	streetname	locality	pao
M1 5BY	*Cambridge Street*	*Manchester*	3
M1 5BY	*Cambridge Street*	*Manchester*	UNIT B2
M18 8GN	Brightman Street	Manchester	30
M26 3NL	Ashcombe Drive	Manchester	1
M3 7EL	Blackfriars Road	Salford	74
M30 0SW	Devonshire Road	Manchester	41
M50 1AU	Pilgrims Way	Salford	APT 42
M8 4QS	Delaunays Road	Manchester	5
M9 8QB	*Lakeside Rise*	*Manchester*	20
M9 8QB	*Lakeside Rise*	*Manchester*	1
OX2 9DU	*Crabtree Road*	*Oxford*	51
OX2 9DU	*Crabtree Road*	*Oxford*	47
OX28 4GE	*Thorney Leys*	*Witney*	9 PARK
OX28 4GE	*Thorney Leys*	*Witney*	17A PARK
OX4 2DU	*Oxford Road*	*Oxford*	128
OX4 2DU	*Oxford Road*	*Oxford*	132
OX5 3DH	Weston Road	Kidlington	NEW FOLD

Source 4: English Deprivation Indices (s_4)

postcode	incomerank
OX10 1EU	29412
OX1 5PG	29540
OX28 4GE	21324
OX2 9DU	30708
OX4 2DU	9412
OX5 3DH	29567
OX7 6QE	27461
M1 5BY	25794
M18 8GN	3527
M19 2LZ	18597
M3 7EL	26678
M30 0SW	9548
M4 5HU	27939
M50 1AU	8133
M8 4QS	2734
M8 5XJ	2734
M9 8QB	2342

Figure 1: Data Sources and Reference Data in a simple data preparation scenario. Specific values are coloured as follows: red font – incorrect value w.r.t. conditional functional dependencies (CFDs); green font: correct value w.r.t. CFDs; blue font – data used in mining CFDs.

process. The proposed approach thus uses the same feedback to inform changes to many different data preparation components, thereby seeking to maximise the return on the investment made in the provision of feedback.

The contributions of the paper are as follows:

1. A technique for applying feedback on a data product across a multi-step data preparation process that both identifies statistically significant issues and provides a mechanism for exploring the actions that may resolve these issues.

2. An approach to feedback targeting that builds on the statistical analysis from (1) to identify the values on which it would be most useful to obtain additional feedback.
3. A realisation of the techniques from (1) and (2) in a specific data preparation platform, where feedback is used to change the matches used in an integration, change which mappings are used, and change which data quality rules are applied.
4. An empirical evaluation of the implementation of the approaches from (3) that investigates the effectiveness of the proposed approaches both for individual data preparation constructs (matches, mappings, and repairs in the form of conditional functional dependencies (CFDs)) and for applying feedback across all these constructs together.

The remainder of the paper is structured as follows. Section 2 outlines the data preparation pipeline on which we build, and provides a running example that will be used in the later sections. Section 3 provides a problem statement and an overview of the approach. Section 4 details the individual components in the realisation of the approach, and presents feedback assimilation algorithms. Section 5 describes how the feedback can be targeted on result values that are relevant to the approach in Section 4. Section 6 evaluates the technique in a real estate application. Section 7 reviews other approaches to increasing the cost-effectiveness of data preparation, and Section 8 concludes. This paper is an extended version of [11], to which extensions include a proposal for acting on feedback only when the action is predicted to provide a benefit, a technique for feedback targeting, and associated comparative evaluations.

2. A Data Preparation Pipeline

This section provides an overview of the aspects of the VADA data preparation architecture that are relevant to the feedback assimilation approach that is the focus of the paper. The VADA architecture is described in more detail in earlier publications [8, 9, 10].

2.1. Defining a Data Preparation Task

In VADA, instead of handcrafting a data preparation workflow, the user focuses on expressing their requirements, and then the system automatically populates the end data product. In particular, the user provides:

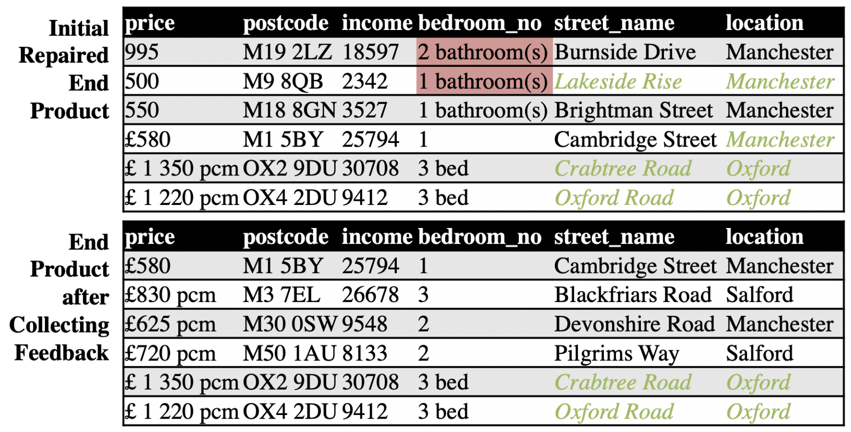
Input Data Sources: A collection of data sources that can be used to populate the result. Figure 1, illustrates the data sources in our running scenario. These sources include real estate property data (sources s_1 to s_3) and open government data (source s_4).

Target Schema: A schema definition for the end data product. In the running example, the target schema consists of one table, *property*, with six attributes, namely *price*, *postcode*, *income*, *bedroom_no*, *street_name* and *location*.

User Context: The desired characteristics of the end product; as the end data product is obtained by an automated process, many candidate solutions can be generated. The user context allows the system to select solutions that meet the specific requirements of the user [12]. The user's requirements are modelled as a weighted set of criteria, with the sum of their weights equal to one. Different quality criteria can be used; for example, VADA supports completness w.r.t. empty values, completeness w.r.t. data context, consistency w.r.t. data context, correctness w.r.t. feedback and relevenace w.r.t. feedback). In our running example, we consider 6 criteria, each one on the correctness with respect to feedback of a target schema attribute, and a weight of $\frac{1}{6}$.

Data Context: Supplementary instance data associated with the target schema, which can be used as additional evidence to inform the automated data preparation process [10]. For example, in Figure 1, reference data is provided that provides definitive address data.

Given the above information, and a user-defined targeted size for the end product of 6 tuples from the data sources in Figure 1, the system can auto-



Initial Repaired End Product

price	postcode	income	bedroom_no	street_name	location
995	M19 2LZ	18597	2 bathroom(s)	Burnside Drive	Manchester
500	M9 8QB	2342	1 bathroom(s)	*Lakeside Rise*	*Manchester*
550	M18 8GN	3527	1 bathroom(s)	Brightman Street	Manchester
£580	M1 5BY	25794	1	Cambridge Street	*Manchester*
£ 1 350 pcm	OX2 9DU	30708	3 bed	*Crabtree Road*	*Oxford*
£ 1 220 pcm	OX4 2DU	9412	3 bed	*Oxford Road*	*Oxford*

End Product after Collecting Feedback

price	postcode	income	bedroom_no	street_name	location
£580	M1 5BY	25794	1	Cambridge Street	Manchester
£830 pcm	M3 7EL	26678	3	Blackfriars Road	Salford
£625 pcm	M30 0SW	9548	2	Devonshire Road	Manchester
£720 pcm	M50 1AU	8133	2	Pilgrims Way	Salford
£ 1 350 pcm	OX2 9DU	30708	3 bed	*Crabtree Road*	*Oxford*
£ 1 220 pcm	OX4 2DU	9412	3 bed	*Oxford Road*	*Oxford*

Figure 2: Using feedback to improve the end product. The shaded red background denotes false positive feedback obtained on the initial end product, in the light of which the system is able to refine the data preparation process to yield the revised data product without the problematic values.

matically produce the first end data product in Figure 2. In the absence of feedback or any other user-defined characteristics, the system will select tuples that are as complete as possible to populate the end data product. However, as illustrated in Figure 2, feedback on the correctness of the result can be used to revise how the data product is produced, and thus improve its overall quality.

2.2. Data Preparation Process and Components

Figure 3 illustrates the basic flow of events for the data processing pipeline in this paper, where the plus precedes and follows parallel tasks. First, the system is initialised using the sources and data context that the user has provided. Then, *CFD Miner*, *Data Profiler* and *Matching* components are run on the sources and data context. Given the matches and profiling data, the *Mapping* component generates a set of candidate mappings, over which *Mapping Selection* evaluates the user criteria to select the most suitable mappings for contributing to the end product. Subsequently, the *Data Repair* component repairs constraint violations that are detected on the end product. The components are now outlined in more detail:

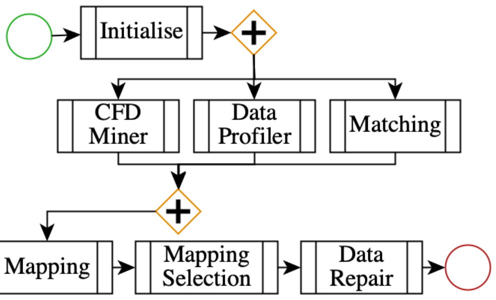


Figure 3: A Data Preparation Pipeline

Initialise: This component sets up the system Knowledge Base with metadata about the data available in the system, the target schema, user preferences and component configurations.

Matching: Given a set of sources and the target schema T, the matching component produces a set of matches between their attributes and the target schema attributes. Each match has a similarity *score*, with $0 \leq score \leq 1$.

Profiling: This component analyses the available sources and produces a set of candidate keys and a set of inclusion dependencies among the sources.

Mapping Generation: Using the results of the two previous components as inputs, mapping generation uses dynamic programming to explore the space of possible ways of combining the sources using unions or joins, producing a set of candidate mappings [13].

Mapping Selection: Given a set of user criteria, a set of candidate mappings, the target schema and a targeted size[2], mapping selection establishes how

[2]In automated data preparation, the system can generate many different ways of populating the target, which may be increasingly unable to meet the user's requirements. As such, the criteria allow the user to make explicit some priorities to the system, and the *size* threshold allows the user to say *enough*. In practice, the size threshold is likely to be set in the light of

many tuples to obtain from each of the candidate mappings to populate a target schema instance, thus creating an end data product [12].

Data Repair: Repair takes place in the presence of reference data; reference data is considered to provide complete coverage of the domains for certain target columns. Data repair involves the cooperation of 2 components. First, CFD Miner is trained on the data context [14]. The component is configured by a *support size* parameter. In the example illustrated in Figure 2, with a support size of 2, it will produce a number of CFDs, including the following:

property([postcode] $\rightarrow$ [streetname],(M1 5BY || Cambridge Street))
property([postcode] $\rightarrow$ [locality],(M9 8QB || Manchester))
property([postcode] $\rightarrow$ [streetname],(OX2 9DU || Crabtree Road))
property([postcode] $\rightarrow$ [locality],(OX28 4GE || Witney))
property([postcode] $\rightarrow$ [streetname],(OX4 2 DU || Oxford Road))

Given a set of CFDs and a dataset, the *Data Repair* component will identify violations of these CFDs, which can then be repaired, in the sense that violations are removed [14]. In Figure 2, rules are learnt from the repeated presence of the italicised tuples highlighted in blue in the reference data in Figure 2. Thus, the incorrect values *Market Street*, *Crabtree Rd*, etc. have been corrected to *Lakeside Rise*, *Crabtree Road*, etc. resulting in an end product that is consistent with respect to the reference data.

3. Problem Statement

This section provides more details on the problem to be solved, along with an overview of the approach to be followed. The problem can be described as follows.

the results obtained.

> Assume we have a data preparation pipeline P, that orchestrates a collection of data preparation steps $\{s_1, ..., s_n\}$, to produce an end data product E that consists of a set of tuples. The problem is, given a set of feedback instances F on tuples from E, to re-orchestrate some or all of the data preparation steps s_i, revised in the light of the feedback, in a way that produces an improved end data product.

In the remainder of this section, we pin down the nature of the feedback, and the way in which the feedback is used to change how data preparation is carried out.

3.1. The Feedback

In this paper, we assume that the feedback takes the form of true positive (*TP*) or false positive (*FP*) annotations on tuples or attribute values from E. Where a tuple is labelled as a *TP* this means that it is considered to belong in the result; and where a tuple is labelled as an *FP*, this means that it should not have been included in the result. When an attribute value is labelled as a *TP*, this means that the given value is a legitimate member of the domain of the column, and when it is labelled as an *FP* this value should not have appeared in that column.

These annotations lend themselves to propagation as follows. If a tuple is marked as *TP*, all of its attribute values are marked as *TP*. If an attribute value is marked as *FP*, all tuples containing that value for the same attribute are marked as *FP*. For example, in Figure 7, as the second tuple has been identified as a *TP*, this means that we know that its attribute values are also legitimate, and thus can be annotated as true positives (e.g. *Bayswater Road* can be considered to be a legitimate value for *street*). Furthermore, as the *bedrooms* value *5* has been annotated as *FP*, we can infer that its tuple and all other tuples with a *bedrooms* value equal to *5* should also be annotated as false positives. These annotations are then used to derive properties of matches, mappings and repair rules, as detailed in Section 4.

3.2. Acting on the Feedback

Given such annotations, the approach consists of the following steps:

1. Given feedback F, identify a collection of hypotheses H that could explain the feedback. For example, if an attribute value is incorrect, the following are possible hypotheses: (i) a match that was used to associate that value in a source with this attribute in the target is incorrect; (ii) a mapping that was used to populate that value in the target is incorrect, for example joining two tables that should not have been joined. Furthermore, if a *tuple* is incorrect, the following are possible hypotheses: (i) the mapping that was used to produce the tuple is incorrect, for example by projecting out inappropriate columns from sources; and (ii) a data repair has introduced an error into the tuple.
2. Given a hypothesis $h \in H$, review all the evidence pertaining to h from the feedback to establish if the confidence in the hypothesis is sufficient to suggest that it should be investigated further. For example, if the hypothesis is that a match is incorrect, all the feedback on data derived from that match should be considered together, with a view to determining whether the match should be considered problematic.
3. Given a hypothesis $h \in H$ in which there is some confidence from feedback, identify actions that could be taken in the pipeline P. For example, if the hypothesis is that a match is incorrect, possible actions would be to drop that match or to drop all the mappings that use the match.
4. Given that evidence from feedback F may support several different hypotheses, there is a space of possible actions that could be carried out, each leading to a different collection of data preparation steps. As a result, we must explore the space of candidate integrations that implement the different actions.

Given the feedback, these steps can be followed by the system without further user involvement, as detailed in Section 4.

4. Solution

This section provides additional details on how the steps from Section 3 are carried out in practice, and includes details on how the feedback can be used to inform actions on matches, mappings and repairs. In particular, Section 4.1 identifies hypotheses that may be suggested by the available feedback; Section 4.2 describes how the statistical significance of such hypotheses can be ascertained; Section 4.3 identifies some actions that may be taken in response to a hypothesis; Section 4.4 describes how the different aspects of the solution are brought together in an algorithm that acts on the hypotheses; and Section 4.5 provides a revised version of the algorithm that acts on the hypotheses only when the results of the actions are predicted to improve the quality of the integration.

4.1. Hypotheses

Given an *end data product* on which some *feedback* has been collected, an obvious question is *what can the feedback tell us about the way in which the data product has been produced.* More specifically, given an *end data product* and some *feedback* that identifies problems with the data product, an obvious question is *what went wrong in the production of this data product.* For example, in Figure 1, the *street_name Market Street* is not consistent with the *postcode M9 8QB*, and thus could have been labelled as an *FP* by the user. It is straightforward to identify possible reasons for this problem, which we term *hypotheses.* In this case, possible reasons include:

Matching: Incorrect match used to relate the source to the target.

Mapping Generation: Incorrect mapping combined sources in an inappropriate way.

Data Repair: Incorrect data repair replaced the correct value with the wrong one.

In this paper, *all* these hypotheses are considered as possible explanations for the identified FP. However, the question then is *when do we provide credence to a hypothesis*? The following section uses statistical techniques to test when the feedback indicates that there is statistically significant evidence to support such hypotheses.

4.2. Hypothesis Testing

This section describes how hypotheses are tested with respect to the evidence from feedback.

As stated in Section 2, the user can define the desired characteristics of the end product in the form of a set of criteria. Some of these criteria can be determined without feedback (such as the completeness with respect to the data context of the values in a column), but some can be informed by feedback (such as relevance and correctness)[3]. In the examples and experiments in this paper, feedback is on correctness. Equation 1 is used to estimate the value of a criterion based on feedback; $\hat{c_s}$ is the estimated value for the criterion on a source s, tp (resp. fp) the numbers of tuples marked by the user as true (resp. false) positives, and $|s|$ is the source size in number of tuples.

$$\hat{c_s} = \frac{1}{2}(1 + \frac{tp - fp}{|s|}) \tag{1}$$

Thus, $\hat{c_s}$ takes a value between 0 and 1, and, in the absence of feedback, the default value for a criterion is *0.5*. For example, consider that $\hat{c_s}$ is an estimate of the correctness of an attribute. If the feedback on the correctness of the attribute has *6 tp* values and *3 fp* values, then if there are *100* tuples, the estimated correctness for the attribute in s is now $\frac{1}{2}(1 + \frac{6-3}{100}) = 0.515$.

[3]In relation to feedback, by *relevance*, we mean that the results are pertinent to an application; for example, in the real estate example, the user could only be interested in properties in Oxford. By *correctness*, we mean that the values conform with the ground truth as seen by the user; for example, the user may know that the number of bedrooms in a property is different from that stated.

We now proceed to establish when criteria estimates are significantly different using standard statistical techniques [15][4]. The estimated values of a criterion $\hat{c}$ on two sources s_2 and s_1 are considered to be *significantly different* when Equation 2 holds:

$$\hat{c}_{s_2} - \hat{c}_{s_1} > z\sqrt{se_{s_2}^2 - se_{s_1}^2} \tag{2}$$

where $\hat{c}_{s_2}$ (resp. $\hat{c}_{s_1}$) is the evaluation of criterion $\hat{c}$ on s_2 (resp. s_1), se_{s_2} and se_{s_1} are the standard errors for sources s_2 and s_1 respectively, calculated by Equation 3, and z is a statistical term measuring the relationship between a value and the mean of a group of values. The standard error is calculated by Equation 3 below.

$$se_s = \sqrt{\frac{\hat{c_s}(1-\hat{c_s})}{L_s}} \tag{3}$$

where s is a source, $\hat{c_s}$ is the evaluated feedback-based criterion on source s, and L_s is the amount of feedback collected on source s.

Given the above, then our estimation of a data criterion $\hat{c_s}$ on a source s is $\hat{c_s} \pm e_s$ where e_s is the margin of error for the data criterion evaluation on s, and $0 \leq \hat{c_s} \pm e_s \leq 1$. A source s can be either a set of values, or a set of tuples. The formula for the margin of error is given in Equation 4.

$$e_s = z \cdot se_s \cdot \sqrt{\frac{|s| - L_s}{|s| - 1}} \tag{4}$$

where $|s|$ is the number of elements in s (source size), and L_s the amount of feedback collected for source s. This is feedback either provided directly on the data product (if it is the end data product) or propagated to it. We only consider attribute-level feedback instances when evaluating L_s on a set of

[4]Such statistical techniques have been used before to compare mappings on the basis of feedback, with a view to targeting feedback collection [16]. In that work, where there is uncertainty as to which mappings should be preferred in mapping selection, additional feedback is obtained to reduce the uncertainty.

values, and we only consider tuple-level feedback instances when evaluating L_s on a set of tuples. In our experiments we use the z-score that corresponds to a confidence level of 80%, i.e. 1.282, so there is an 80% chance that the true value lies within the margin of error of the estimate.

In the absence of feedback on a set of tuples or values, given that $\hat{c_s}$ is unknown, hence $\hat{c_s} = \frac{1}{2} \pm \frac{1}{2}$, we can solve Equation 4 by setting $e_s = \frac{1}{2}$ and $L_s = 0$, and evaluate the standard error se_s for the source s (needed for evaluating significant difference using Equation 2) in the absence of feedback as:

$$se_s = \frac{1}{2z}\sqrt{\frac{|s|-1}{|s|}} \tag{5}$$

Note that where feedback is sparse, the standard error is likely to be large, and Equation 2 will tend not to be satisfied. This provides a means of determining when to respond to feedback without the use of arbitrary thresholds.

Next, we discuss the comparisons we used when testing the different components in the system. The focus is on the value sets c_1 and c_2 being considered as s_1 and s_2 respectively, when evaluating Equation 2.

Testing matches for significant difference. If the data that results from a match is significantly worse than that produced from other matches for the same target schema attribute, then the match is suspicious. The statistical significance of the difference between match results can be determined using Equation 2. Figure 4 provides an example of the target schema T and a source s, with a match between attribute $s.bed\_num$ and the target schema attribute $T.bedroom\_no$: $m_1 : s.bed\_num \sim T.bedroom\_no$.

When testing match m_1 for significant difference, Equation 2 is evaluated on the projections on the matching attributes. Specifically, to test the significance of match m_1, for the evaluation of Equation 2, we use feedback on the value sets c_1 and c_2 (i.e., the values populating $s.bed\_num$ and $T.bedroom\_no$), as illustrated in Figure 4, in estimating criterion values and associated statistical properties.

Note that the set c_1 is the complete set of values for attribute $s.bed\_num$,

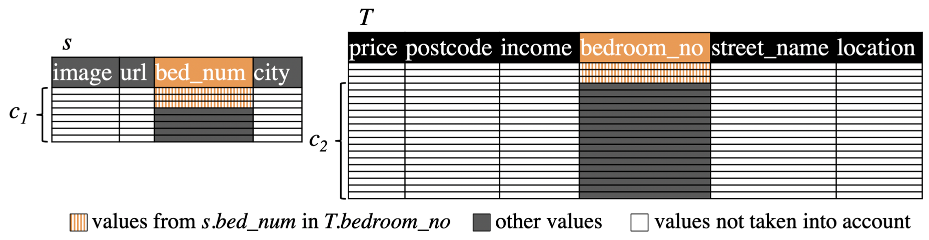


Figure 4: Detecting significantly different matches

regardless of whether the values are selected to populate the end product or not, while c_2 is the greyed part of attribute $T.bedroom\_no$.

Consider the example in Figure 1, for which some feedback has been obtained on the initial end product in Figure 2, to the effect that the first two tuples have *fp* annotations for their *bedroom_no*. In Figure 2, the first 3 tuples have been obtained from *Source 1*, but involve an incorrect match of *Source1.bathrooms* with *Target.bedroom_no*. The correctness estimated for this match using Equation 1 is $\frac{1}{2}(1+\frac{0-2}{3}) = 0.167$. Assume that the correctness for the values obtained from other matches on the same column is estimated on the basis of feedback collected using Equation 1 to be *0.5*. Although with this small illustrative example statistical significance is difficult to establish, we can see that the correctness estimate associated with the match from *Source1.bathrooms* with *Target.bedroom_no* is lower than that for the other values in the column (obtained using matches involving source attributes *Source 2.bedroom_no* and *Source 3.bed_num*), and thus the match involving *Source1.bathrooms* with *Target.bedroom_no* may be identified as being suspicious.

Testing mappings for significant difference. If the result of a mapping is significantly worse than the results from the other mappings, then the mapping is suspicious. The statistical significance of the difference between mapping results can be determined using Equation 2.

Figure 5 shows an example of a target schema T populated with tuples selected from 5 candidate mappings. Candidate mappings m_1 to m_4 contribute

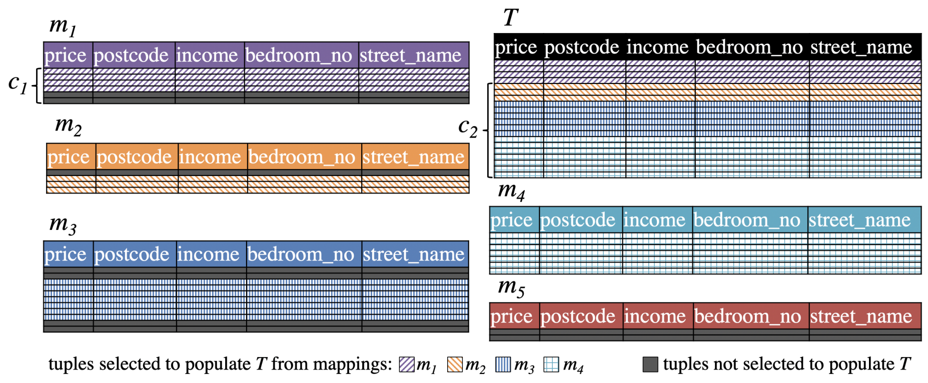

m_1
price postcode income bedroom_no street_name
c_1
m_2
price postcode income bedroom_no street_name
m_3
price postcode income bedroom_no street_name
T
price postcode income bedroom_no street_name
c_2
m_4
price postcode income bedroom_no street_name
m_5
price postcode income bedroom_no street_name
tuples selected to populate T from mappings: ▨ m_1 ▨ m_2 ▥ m_3 ⊞ m_4
■ tuples not selected to populate T

Figure 5: Detecting significantly different mappings

to the solution, while m_5 does not. For each of the candidate mappings that contributes to the solution, we evaluate the participating tuples against the rest of the tuples of the end product, using Equation 2. For instance, to evaluate whether mapping m_1 is significantly different, we use feedback on the tuples in c_1 and c_2, as illustrated in Figure 5.

It is important to mention that for mappings, before evaluation, we propagate to the mapping the feedback that the user has provided on the end product. Furthermore, the Mapping Selection component ensures that the tuples that are marked as true (resp. false) positives are selected (resp. not selected).

Testing repairs for significant difference. If the repaired values from a CFD are significantly worse than the other values for the repaired attribute, then the CFD is suspicious. The statistical significance of the difference between repair results and the underlying data set can be determined using Equation 2.

In Figure 6 we see the result of a CFD, cfd_1, on the end product. We mark as green the *2* attribute values for column *T.postcode* that are found to be correct, and with red the *3* values in column *T.income* found to be violating cfd_1, and that were thus repaired. As before, we use feedback on the tuple sets c_1 and c_2 in Figure 6 to provide the inputs to Equation 2. Note that the values annotated as correct in Figure 6 are not considered in c_1. We only consider the

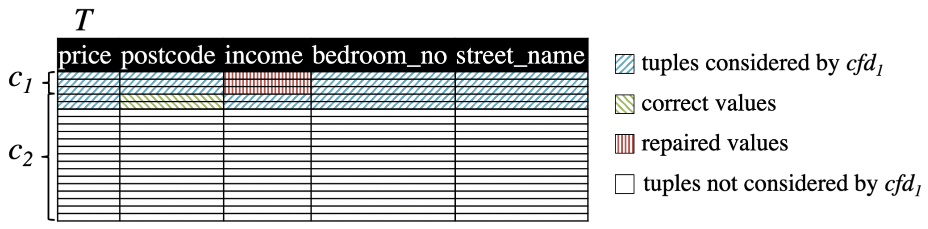


Figure 6: Detecting significantly different CFDs

tuples that were modified by the CFDs.

4.3. Actions

Each of the hypotheses is associated with a respective action, typically set in this paper to ruling out the suspicious item before rerunning the data preparation pipeline. An item here is a match, a candidate mapping, or a repair rule. The hypotheses and respective actions per component are thus summarised below:

- **Matching.** If a match is suspicious, discard the match. This means that mapping generation (or any of the other components down the line) will not take this match into account.
- **Mapping Generation.** If a mapping is suspicious, discard the mapping. This means that Mapping Selection will not take this mapping into account.
- **Data Repair.** If a repair rule is suspicious, discard the CFD.

We need not take actions for items that are found to be significantly better than others, as they are retained by default. As such, we focus on removing suspicious items.

These actions may be felt to be quite blunt. However, we note that the actions are only taken when there is (statistically significant) evidence that the deleted match, mapping or CFD is problematic. The deletion of a match means that the system may choose a different match from a matcher. The deletion of a mapping means that the number of results previously obtained from the deleted mapping will now be obtained from one or more other mappings. The deletion of the CFD most likely result in fewer repairs.

The actions do not guarantee that the result will be improved. As such, we introduce two algorithms for applying actions, one that always applies a proposed action (in Section 4.4), and another that applies a proposed action only if the result is predicted to be better than before (in Section 4.5). These algorithms are then compared in experiments in Section 6.

4.4. Algorithm 1: Acting on the Hypotheses

Algorithm 1 provides the basic sequence of events while assimilating feedback. It is assumed that the feedback assimilation takes place in the context of an integration, in which *Matches* is the set of existing matches, *profileData* contains the existing inclusion dependency and primary key data, and *CFDs* is a set of existing conditional functional dependencies, as produced in the upper part of Figure 3.

Then, the algorithm considers the hypotheses on matches, mappings and CFDs in turn; this order is chosen because changes to matches give rise to changes to mappings, and CFDs are applied to the results of mappings.

First, we iterate over the available matches (lines 2–6) and test whether any of these produces a result that is significantly worse than the results of other matches. $T.a$ (line 3) is a target schema attribute that is the target of a match m with a source attribute $m.a$. Any match that yields a significantly worse result is removed.

The remaining matches are then used for mapping generation (line 7), and any generated mappings that perform significantly worse than the others are removed (lines 8–12).

A similar process is followed with CFDs, removing CDFs that are found to be problematic in the light of the feedback (lines 14–19). The end product is then repaired (line 20) using the reduced set of CFDs. The resulting end data product can then be provided to the user for analysis or further feedback collection. Newly collected feedback instances are added to the existing ones, and propagated to the end product and the candidate mappings. In addition, the feedback can periodically be applied to the whole process using Algorithm 1 to generate a revised end product. The process continues until the user finds the result fit for purpose and terminates the data preparation pipeline.

Function significantlyWorse (lines 3, 9 and 16) is essentially testing Equation 2, and returns true if it holds, or false otherwise. The arguments for this function are the ones illustrated as c_1 and c_2 in Figure 4 when detecting significantly worse matches, Figure 5 when detecting significantly worse mappings, and Figure 6 when detecting significantly worse CFDs. Next, using the same approach we detect and remove suspicious mappings (lines 12–16) and suspicious CFDs (lines 18–23). As s in line 15 we define the set of tuples in T violating a given cfd.

4.5. Algorithm 2: Cautiously Acting on the Hypothesis

The algorithm ApplyFeedback from Section 4.4 is decisive – when a hypothesis is supported by the evidence from feedback, its associated action is carried out. However, there is no guarantee that this will lead to an improved result; there is still a chance that the hypothesis is not correct, and it is also possible that (even if the hypothesis is correct) the action will not lead to an improved result. For example, a *match* could be removed, but the problem could be with the data in the source and not with the match, in which case there may not be a more suitable match. Similarly, a *mapping* could be removed but there may not be a better mapping available to replace it.

With a view to avoiding these risks, Algorithm 2 takes a more cautious approach. In essence, Algorithm 2 calls Algorithm 1, but the end product that results from the actions carried out by Algorithm 1 are only retained if they are predicted (in the light of the feedback) to be better than or equally as good as the existing results.

Thus in, Algorithm 2, the first step is to estimate the precision of the current *endProduct* (line 2); the function *precisionF* uses Equation 1 to estimate the precision of the current end product in the light of the available feedback. Then, ApplyFeedback is used to apply the available feedback to change the integration by removing suspicious *matches*, *mappings* and *CFDs* (line 3). The preferred end product is the one with the highest estimated precision (lines 5-7).

We note that although *Algorithm 2* may be felt to be a simple improvement to *Algorithm 1*, the experiments provide a different story.

5. Targeted Feedback Collection

Feedback collection can follow one of two rather different approaches:

- In *targeted* feedback collection, the *system* identifies what feedback should be collected, and prompts the user for that feedback.
- In *untargeted* feedback collection, the *user* decides what feedback to provide, and quite likely when to provide it.

In the *targeted* approach, there is a specific interface for feedback collection, and feedback collection is separate from the browsing of results. In the *untargeted* approach, typically feedback is collected as a side-effect of some other activity, such as the viewing of the result of the data preparation process. For example, Figure 7 shows an interface for annotating *cells* or *tuples* as *relevant* (shaded in green) or *not relevant* (shaded in purple) within a user interface whose primary purpose is to support the browsing of the end product. For example, the feedback indicates that houses with *5 bedrooms* are not relevant.

An advantage of the *targeted* approach is that the system can seek to obtain the feedback that will be most valuable for informing decision-making. However, there is a risk that the system may seek feedback that requires knowledge the users do not have. An advantage of the *untargeted* approach is that it is less obtrusive for the user, but a disadvantage is that the user may provide feedback

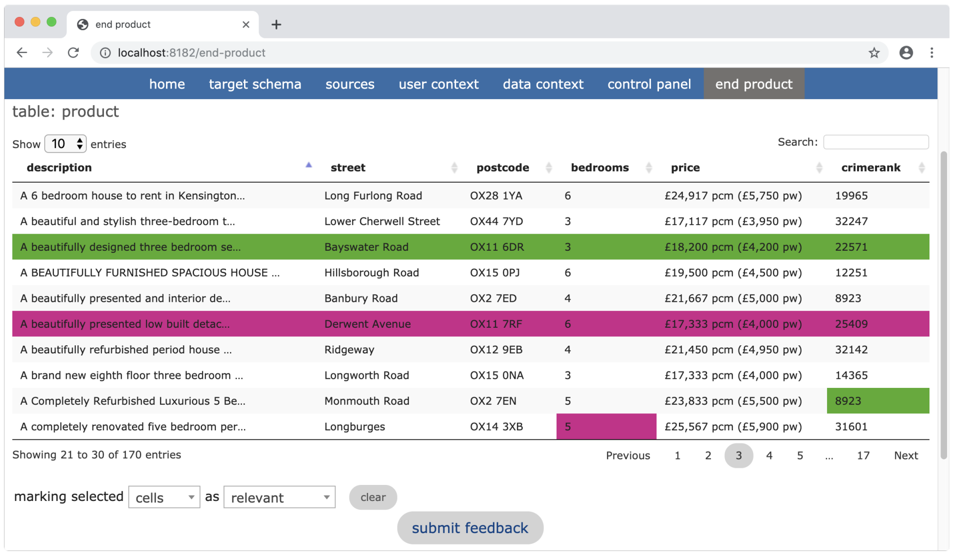


table: product

Show 10 entries

description	street	postcode	bedrooms	price	crimerank
A 6 bedroom house to rent in Kensington...	Long Furlong Road	OX28 1YA	6	£24,917 pcm (£5,750 pw)	19965
A beautiful and stylish three-bedroom t...	Lower Cherwell Street	OX44 7YD	3	£17,117 pcm (£3,950 pw)	32247
A beautifully designed three bedroom se...	Bayswater Road	OX11 6DR	3	£18,200 pcm (£4,200 pw)	22571
A BEAUTIFULLY FURNISHED SPACIOUS HOUSE ...	Hillsborough Road	OX15 0PJ	6	£19,500 pcm (£4,500 pw)	12251
A beautifully presented and interior de...	Banbury Road	OX2 7ED	4	£21,667 pcm (£5,000 pw)	8923
A beautifully presented low built detac...	Derwent Avenue	OX11 7RF	6	£17,333 pcm (£4,000 pw)	25409
A beautifully refurbished period house ...	Ridgeway	OX12 9EB	4	£21,450 pcm (£4,950 pw)	32142
A brand new eighth floor three bedroom ...	Longworth Road	OX15 0NA	3	£17,333 pcm (£4,000 pw)	14365
A Completely Refurbished Luxurious 5 Be...	Monmouth Road	OX2 7EN	5	£23,833 pcm (£5,500 pw)	8923
A completely renovated five bedroom per...	Longburges	OX14 3XB	5	£25,567 pcm (£5,900 pw)	31601

Showing 21 to 30 of 170 entries

marking selected cells as relevant

Figure 7: Feedback collection interface.

that is not especially useful to the system. In any event, we can only adopt a targeted approach if a suitable targeting scheme can be proposed.

Our approach to feedback targeting is motivated by the way in which feedback is used to test hypotheses in Section 4, where feedback is collected on integration artifacts (matches, mappings and CDFs) that are statistically *significantly worse* than their counterparts. For feedback targeting, our approach aims to obtain feedback in a way that will help to identify these *significantly worse* artifacts. The approach is as described in Algorithm 3.

This algorithm follows quite closely the structure of Algorithm 1, as the idea is to obtain feedback only on the data that may benefit Algorithm 1. As Algorithm 1 collects feedback in cases where the feedback identifies suspicious cases, there is no point in collecting feedback where there is already enough feedback to establish that an artifact is (statistically significantly) better than or worse than its counterparts. Both *significantlyBetter* and *significantlyWorse* are defined in terms of Equation 2. As a result, Algorithm 3, returns in *bfTarget* the data from the end product that is associated with matches, mappings or repairs for which the hypotheses from Section 4.1 cannot yet be established as true or false.

6. Evaluation

6.1. Experimental Setup

For the evaluation, we used the following datasets:

- 40 datasets with real-estate properties extracted from the web using OXpath [17], with contents similar to sources s_1 to s_3 in Figure 1. Each dataset had from 6 to 16 attributes, with an average of 11 attributes per dataset. Their initial total size was 7.8k tuples. These datasets were used as source data.

- English indices of deprivation data, downloaded from www.gov.uk, as shown in s_4 in Figure 1. The complete dataset had 6 attributes and 62.3k tuples with indices for Manchester and Oxford. The dataset was used as a source.

- Open addresses UK data from openaddressesuk.org. The complete dataset had 11 attributes, 41.7k tuples for Manchester and Oxford (postcode sectors M and O). The dataset was used as reference data.

Then, to enable the automatic evaluation of correctness on the end product and throughout the pipeline, we constructed a ground truth based on the available data, which we used for our experiments, as follows: we manually matched, mapped, deduplicated and then repaired the end product. This gave us a dataset consisting of approximately 4.5k tuples.

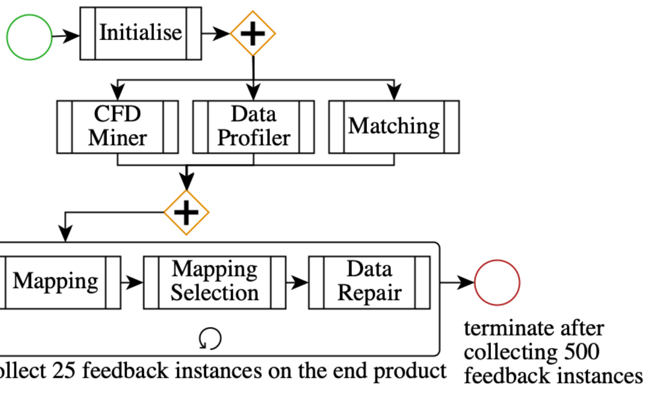
For the experiments, the match *threshold* was set to 0.6, the top 100 mappings are retained from mapping generation, and the *support size* for the data repair component was set to 15. From experience, these are reasonable *match* and *support size* thresholds, allowing through reasonable numbers of matches and CFDs (respectively) without deluging the system with unpromising proposals. The top *100* mappings is fully sufficient to include the most promising candidates over 40 datasets. When making decisions based on feedback, a confidence level of 80%, is used. The results are not especially sensitive to the confidence level. We experimented with different confidence levels; for example when the confidence level is increased, typically the same hypotheses are found to be true, but after a few more items of feedback have been collected to support them.

Each of the criteria in the user context were set to be the correctness of an attribute from the target schema. Thus, the user criteria are: correctness(price), correctness(postcode), correctness(income), correctness(bedroom_no), correctness(street_name) and correctness(location). They all have the same weight, $\frac{1}{6}$. The correctness of each of these properties is estimated based on feedback. Mapping selection is configured to select what are predicted to be the best *1000* tuples from a subset of the generated mappings.

The workflow for the experiments is illustrated in Figure 8. We essentially automatically reran the workflow of Figure 3, collecting 25 feedback instances in each iteration, until 500 feedback instances on the end product had been collected and propagated. Feedback is generated randomly in the experiments, based on the correctness of the respective tuple with respect to the ground truth. In each iteration, half of the feedback is given at tuple level, and half at attribute level. By *randomly generated*, we mean that feedback is obtained by randomly selecting true and false positive annotations from the ground truth.

We then configured the system to test:

- Matching. This means running Algorithm 1 without lines 8–12, and without lines 14–19.
- Mapping generation. This means running Algorithm 1 without lines 2–6, and without lines 14–19.
- Data repair. This means running Algorithm 1 without lines 2–6, and without lines 8–12.



Initialise
CFD Miner
Data Profiler
Matching
Mapping
Mapping Selection
Data Repair
collect 25 feedback instances on the end product
terminate after collecting 500 feedback instances

Figure 8: Experiments

- All of the components. This means running Algorithm 1 as defined.
- None of the components. This means running Algorithm 1 without lines 2–6, without lines 8–12, and without lines 14–19. In this case, although no actions are taken to remove matches, mappings or CFDs, the data product is still improved in the light of feedback, as *MappingSelection* in line 13 is able to benefit from improved estimates of mapping correctness.

The *none of the components* case forms the baseline, as we know of no direct competitor that is seeking to apply feedback on the result of an integration to revise the behaviour of several integration components.

6.2. Results

In the figures in this section, *v0* refers to Algorithm 1 that applies feedback with untargeted feedback collection, *v1* refers to Algorithm 2 that applies feedback cautiously, and *v2* refers to collecting targeted feedback with Algorithm 3, then applying it using Algorithm 1. Each of the lines in the figures in this section is the average of 20 runs from 0 to 500 randomly collected feedback instances, using the configurations from Section 6.1, while only reporting the measurements that fall between the 5th and 95th percentile so as to further reduce randomness effects.

As usual, the precision of the end product is evaluated as: $precision = \frac{tp}{tp+fp}$, where a tuple is considered a true positive (tp) if all of its values appear in the ground truth, and as a false positive (fp) otherwise.

The precisions obtained when testing each of the three cases while considering all of the components for suspicious items are illustrated in Figure 9. In a setting where mapping selection retains the top *1000* tuples from a ground truth of around *4500* tuples, the recall increases in line with the precision. Note that, as the collection of different feedback instances leads to different possible actions, which leads to different results being made available for feedback collection, there is no realistic way to provide identical feedback instances to different runs of the experiment. This is reflected in the uneven curves in Figure 9.

Note that differences between results with small numbers of feedback instances are a consequence of the random sampling of feedback, and its impact on mapping selection, and not a consequence of the actions taken based on hypotheses. For small numbers of feedback instances there are few or no actions taken based on the feedback (see Figures 13 to 15) because there is not enough evidence to support the hypotheses.

We notice in Figure 9 that targeted feedback collection (v2) is an improvement on untargeted feedback collection (v0), as with less feedback collected we can typically gain higher precision on the end product. Applying actions cautiously (v1) did not perform as well. One explanation for this could be as follows. With cautious adaptation, there are fewer changes to the result that is the basis for feedback collection, and thus feedback keeps accumulating on the same portion of the result. Such feedback will tend to confirm what is already known, rather than enabling insights to be gained on other parts of the result.

Notice in Figures 10 and 12 that the *none* case (i.e., when there is no discarding of matches, mappings or rules along the way) still leads to an increase of the precision of the end product. This is because, throughout the experiments, mapping selection (as outlined in Section 2.2) is informed by correctness estimates, and feedback is used to improve the correctness estimates. So, before any feedback is collected, all mappings have the same correctness (estimated at

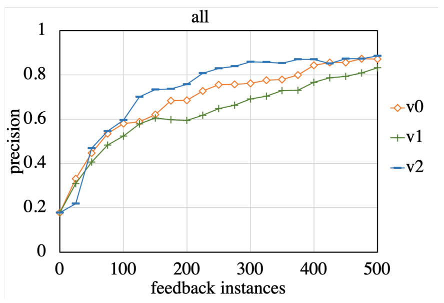


Figure 9: Precision for different algorithms

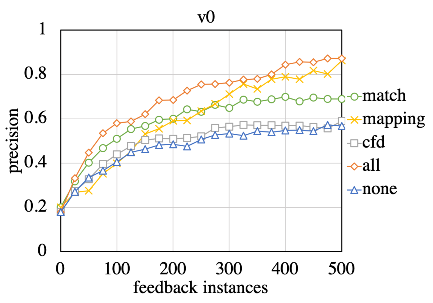


Figure 10: Precision for Algorithm 1

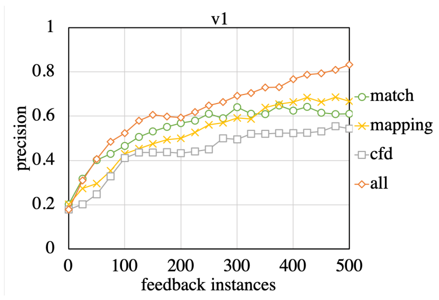


Figure 11: Precision for Algorithm 2

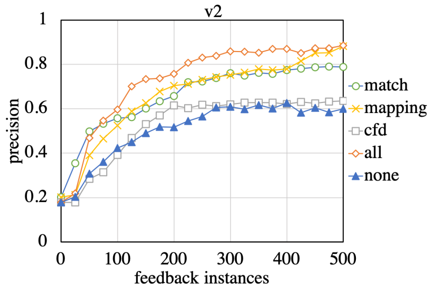


Figure 12: Precision for targeted feedback collection

0.5) and thus mapping selection considers all mappings to be of equal quality. However, as feedback is gathered, different mappings will come to have different levels of correctness, and mapping selection will make increasingly well informed selection decisions. As these selection decisions relate to correctness, this in turn leads to more true positives in the results and a higher precision.

As such, there might be considered to be 2 baselines in these experiments: the case in which there is no feedback, and the precision is around *0.2*, and *none* in which the feedback informs mapping selection, but the approach from Section 4 is not applied. Note that we did not run tests for a *none* case in Figure 11 as it is the same as in Figure 10.

In Figure 9, the results are shown when all components are being considered for suspicious items. Next, in Figures 10, 11 and 12, we illustrate the number of suspicious items per component (i.e., match, mapping and repair), in cases v0, v1 and v2, respectively. The individual component results are similar per case, with the mapping generation component typically being the most impactful. Ruling out suspicious matches also yields good results with few feedback instances, as shown in the respective measurements in Figures 10, 11 and 12, but the measurements show that once the suspicious matches have been identified, more feedback does not yield further improvements when the only actions relate to matches.

The actions that have been taken by the matching and mapping components, i.e., the removal of suspicious matches or mappings, have had a positive impact on the precision of the end product, as shown in Figures 10, 11 and 12. It is unclear from this experiment which one will generally have more impact, although removal of suspicious mappings tends to be the most beneficial action, especially when more feedback has been obtained.

Taking action on the CFDs has had little impact on the end product precision. This can be understood as follows:

- The CFDs being learnt are numerous, e.g., 1260 CFDs were learnt with a support size of 15. As a result, each CFD applies to quite few rows, and it can be difficult to obtain enough feedback to draw statistically significant conclusions as to the quality of an individual CFD.
- Each of the CDFs typically has a small effect to the end product. As such, even when a problematic CFD is removed, this tends not to have much of an effect on overall quality.

However, it is still clear that identifying CFDs that introduced errors to the data, and discarding them, has the potential to have a positive effect on the resulting quality.

We noticed in the experiments that discarding suspicious matches, mappings or CFDs, even in the case of significant issues with the quality of their associated results, did not always guarantee that the precision of the end product would increase. This happens because the contents of the tuples that replace the discarded ones are yet unknown, as feedback on them is not always present. As such, the new tuples are not guaranteed to be of better quality. The trend

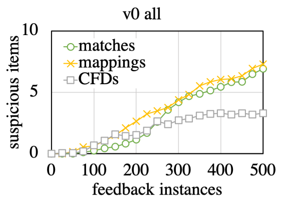


Figure 13: Breakdown of suspicious items for v0, all components considered

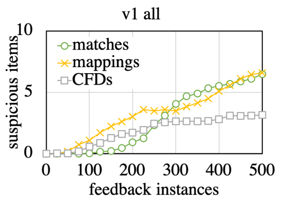


Figure 14: Breakdown of suspicious items for v1, all components considered

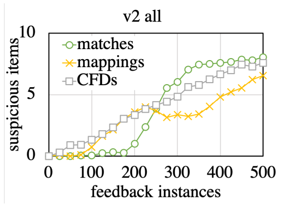


Figure 15: Breakdown of suspicious items for v2, all components considered

is, though, that the overall correctness can be expected to increase, even if not monotonically.

The targeted feedback collection (v2) seemed to present the best results, followed by v0, then by v1, not only in cases when all components were being tested, but also in cases components were being tested individually.

In all cases, actions in the mapping component seem to be delivering good results, eventually reaching a similar impact to considering all components in the cases of v0 (Figure 10) and v2 (Figure 12).

Next, in Figures 13, 14 and 15, we present a breakdown of how different components performed in each of the three approaches tested in the same experiments as in Figures 10, 11 and 12, respectively. We notice that in v0 and in v1 approximately the same number of suspicious items was detected, but in v2, when feedback collection is targeted, more suspicious matches and more suspicious mappings will be discovered with fewer feedback instances. We also notice that mappings can be ruled out with relatively fewer instances than matches. Quite a few feedback instances need to be collected before matches are considered suspicious. This is because FP feedback on *any* of the attributes of a mapping can cause it to be considered to be suspicious, whereas a match can only become suspicious on the basis of evidence about a specific attribute. As a result, statistically significant evidence tends to take longer to accrue for matches than mappings.

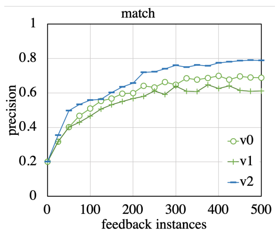


Figure 16: Testing the match component

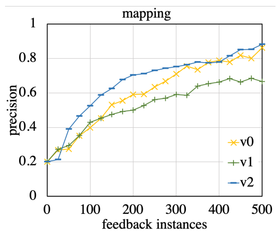


Figure 17: Testing the mapping component

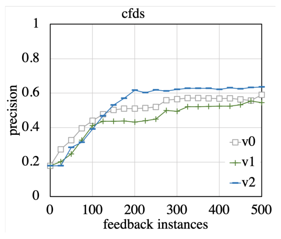


Figure 18: Testing the data repair component

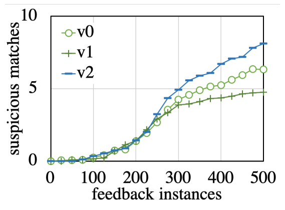


Figure 19: Suspicious matches for different algorithms

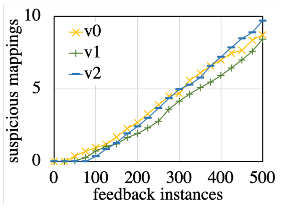


Figure 20: Suspicious mappings for different algorithms

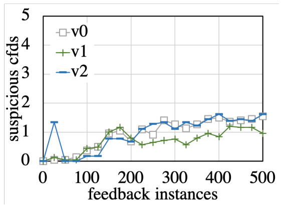


Figure 21: Suspicious CFDs for different algorithms

Next, in Figures 16, 17 and 18, we present for comparison the tests in which only a single component (one of match, mapping, or repair) was being checked, in the various cases (v0, v1, or v2). These are the same lines as in the graphs in Figures 10, 11 and 12, only grouped differently here for comparison, in order to study for each component the different behaviour per algorithm.

It is shown that, setting aside the randomness that is exhibited when few feedback instances (less than 100) are present and few or no actions are taken, precision tends to increase faster in the case of targeted feedback (v2) than in the other cases. It is also consistently higher than in the other cases, though slightly. Cautious application of feedback (v1) did not deliver good quality results in individual components.

We can observe the following: (i) Rather few suspicious matches have been detected, so the benefit obtained from the removal of each such match has been quite substantial. We note that as matches relate to individual columns, obtaining sufficient *FP* feedback on the data deriving from a match can require quite a lot of feedback. (ii) More suspicious mappings are identified, and they are identified from early in the process. (iii) Although quite a few suspicious CFDs are identified, this is a small fraction of the overall number.

Next, we report on the number of suspicious items discovered in each of the experiments. Each line in Figures 19, 20 and 21, corresponds to the average of the same 20 runs as before. These figures, respectively, show the numbers of suspicious matches, mappings and CFDs considered, when only the respective component is being tested.

We notice in Figure 19 that the shape of the curve for the suspicious matches is similar in the three cases, with targeted feedback (v2) leading to the discovery of more suspicious matches. In Figure 20 we notice that targeted feedback collection did not identify more suspicious mappings, so the benefit for precision from targeting must have come from identifying different mappings as suspicious. Regarding the suspicious CFDs discovered and plotted in Figure 21, we notice that even with targeted feedback collection, the large number of CFDs, and the small number of affected tuples per CFD, as described before, did not lead to discovering many of them.

We can see that the suspicious items (matches, mappings and CFDs) when each component was being considered individually, were less than when all components were being considered together. This is an effect of feedback propagation; e.g., when marking a mapping as suspicious, the tuples resulting from respective matches and cfds were also marked as suspicious.

The most important result, however, is that taking feedback into account and acting on all components (using Algorithm 1) increases the return on the investment from feedback (as illustrated in Figure 10). When considering all possible actions together, the overall benefit is greater, and the benefit accrues with smaller amounts of feedback. Furthermore, when feedback collection is targeted on the items for which there is no estimate of significant difference from the rest (using Algorithm 3), the benefits from the actions can be achieved with even less feedback (as illustrated in Figure 12).

7. Related Work

In this section, we consider related work under four headings, *pay-as-you-go data preparation*, *reducing manual effort in data preparation*, *applying feedback to multiple activities* and *targeted feedback collection*.

7.1. Pay-as-you-go Data Preparation

In *pay-as-you-go* data preparation, following from the vision of dataspaces [4], a best-effort integration is produced through automated bootstrapping, which is refined in the light of user feedback. There have been several proposals for complete dataspace systems, with different emphases, for example, in relation to programmable components [18], uncertainty in bootstrapping [19] and personal information management [20].

There have also been many proposals for pay-as-you-go data preparation components, for example relating to data extraction [21], matching [22, 23], mapping [24, 25, 26] and entity resolution [27, 28]. Such proposals have addressed issues such as targeting the most appropriate feedback [21, 16, 29], and accommodating unreliable feedback, in particular in the context of crowdsourcing [6, 30]. However, such work has primarily used a single type of feedback to inform a single data preparation step.

7.2. Reducing Manual Effort in Data Preparation

There are a variety of different approaches to *reducing manual effort in data preparation*. For example, again in the context of individual steps, tools can be used to support data engineers in developing transformations. For example, in relation to format transformation, Wrangler [31] can suggest potential transformation programs, and FlashFill [32] can synthesize transformation programs from examples. There are also proposals in which mappings are discovered based on data instances (e.g., [33, 34, 35]). In ETL, there are also a variety of approaches that seek to reduce the amount of manual labour required. For example, this includes the provision of language features [36, 37] or patterns [38, 39] that support recurring data preparation behaviours, techniques for managing evolution of ETL programs [40], and development of ETL processes that abstract over more concrete implementation details [41, 42]. However, although such work focuses on raising the abstraction levels at which data engineers engage in data preparation tasks, we are not aware of prior results that use feedback on data products to make changes across complete data preparation processes.

7.3. Applying Feedback to Multiple Activities

It has been recognised that the same feedback may be able to inform different aspects of the same step. For example, in [24], feedback on the correctness of results is used to inform both *mapping selection* and *mapping refinement*. In contrast with the work here, these are two essentially independent proposals that happen to use the same feedback, where the feedback has been obtained directly on the mapping results. In Corleone [27], several different aspects of entity resolution are informed by feedback on matching pairs; in particular the feedback is used to inform the learning of comparison rules and to identify and resolve problem cases. This may be the closest work to ours, though the focus is on a single data integration step, for which custom feedback has been obtained. Also for entity resolution, feedback on matching pairs is used in [28] as part of a single optimisation step that configures together blocking, detailed comparison and clustering. This contrasts with the current paper in focusing on different aspects of the same integration step. To the best of our knowledge, there is no prior work in which several distinct steps in the data preparation process are considered together when responding to feedback.

Although not relating to feedback as such, AlphaClean [43] configures parameters for several data cleaning components in a data cleaning pipeline, and thus explores how data preparation outcomes can be improved by considering several steps together. It is assumed that there is a common representation for repairs that change attribute values, and that the quality of a data set can be assessed using user defined functions. AlphaClean then searches for threshold values and parameters that configure the different components. AlphaClean shares with this paper the goal of improving the behaviour of several data preparation components, though these components have a more narrowly focused purpose (revising attribute values). In our case it could also be desirable to systematically search the space of possible preparation plans in the light of feedback, but this space can become large, and evaluating candidate plans takes some time (e.g. as was the case for [28]), so significant challenges would have to be overcome to realise this in practice.

7.4. Targeted Feedback Collection

There are two fundamentally different approaches to feedback collection, *untargeted* and *targeted*. In *untargeted* feedback collection, the user can provide feedback on whatever data items they choose. In terms of the user interface, one approach could allow feedback to be entered when browsing the result of a data preparation process. In contrast, in *targeted* feedback collection, an algorithm identifies data items on which it would be most useful to obtain feedback, and requests this feedback explicitly. In such an approach, the feedback may be obtained from a user or from a third party, such as a crowd worker [6, 30].

Specific proposals for feedback targeting can use *generic* or *bespoke* techniques. Generic techniques build on a methodology that is applicable to a range of problems, such as one of the flavours of active learning [44], which has been applied to several different problems relating to data preparation, such as data extraction [21], mapping generation [29] and entity resolution [45]. Active learning needs to be configured for application to a particular problem, for example through a model that predicts the impact of additional evidence on the uncertainty of a model of the problem.

While active learning has been shown to be effective in a range of settings, several authors have developed bespoke techniques for feedback targeting, which build directly on features of the problem; examples include the use of probabilistic models for record linkage [46] and entity resolution [47], and custom techniques for organising entity resolution feedback collection [48].

In this paper, all decision making, including that relating to feedback targeting, builds on estimates of the quality of matches, mappings and CFDs, taking into account the errors in these estimates that result from sampling. In this setting, feedback can be targeted on the result tuples for which additional evidence may be useful for clarifying hypotheses relating to matches, mappings or repair rules. The same statistical techniques have been applied to target feedback for source selection based on estimates of precision and recall [16] and on multi-criteria decision analysis [49]. In all cases, candidates for additional feedback are those data items that have a statistically significant prospect of impacting on the result.

8. Conclusions

The development of data preparation processes is laborious, requiring sustained attention to detail from data engineers across a variety of tasks. Many of these tasks involve activities that are amenable to automation. However, automated approaches have partial knowledge, and thus cannot necessarily be relied upon to make the best integration decisions. When automation falls short, one way to improve the situation is through feedback on the candidate end data product. The successful combination of automation and feedback provides a route to data preparation without programming, which was considered to be important by 90% of participants in a survey on end user data preparation[5].

Towards this goal of reducing the burden of data preparation, we now revisit and elaborate on the contributions from the introduction.

1. *A technique for applying feedback on a data product across a multi-step data preparation process that both identifies statistically significant issues*

[5]https://www.datawatch.com/2017-end-user-data-preparation-market-study-2/

and provides a mechanism for exploring the actions that may resolve these issues. We have described an approach in which hypotheses about the problems with an integration are tested for statistical significance with respect to user feedback on the candidate end data product, giving rise to actions that seek to resolve issues with the feedback. The same approach is potentially applicable to different types of feedback, diverse data preparation components, and a variety of actions.

2. *An approach to feedback targeting that builds on the statistical analysis from (1) to identify the values on which it would be most useful to obtain additional feedback.* We have described an approach that targets feedback on the results that have the potential to confirm or refute the hypotheses from (1).

3. *A realisation of the techniques from (1) and (2) in a specific data preparation platform, where feedback is used to change the matches used in an integration, change which mappings are selected, and change which data quality rules are applied.* We have indicated how the techniques can be applied to matching, mapping and repair steps, on the basis of true/false positive annotations on data product tuples, in the VADA data preparation system.

4. *An empirical evaluation of the implementation of the approach from (3) that investigates the effectiveness of the proposed approach both for individual data preparation constructs (matches, mappings and repair rules) and for applying feedback across all these constructs together.* An experimental evaluation with real estate data has shown how the approach can identify actions that can improve data product quality on the basis of changes to individual data preparation steps, and can coordinate changes across multiple such steps, with particularly significant benefits from the combined approach. Furthermore, it has been shown how the proposal for feedback targeting enables improvements to precision to be obtained with less feedback.

There are several promising directions for further investigation, which include: (a) extending the data preparation components to which the approach is applied, for example to include source selection or data format transformation; (b) considering alternative actions, such as changing match scores, mapping algorithm thresholds, or rule learning parameters; and (c) more selective or incremental application of actions, with a view to identifying the subset of the candidate actions that together are the most effective.

Acknowledgement: This work is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) through the VADA Programme Grant.

References

[1] P. Vassiliadis, A survey of extract-transform-load technology, IJDWM 5 (3) (2011) 1–27.

[2] T. Rattenbury, J. Hellerstein, J. Heer, S. Kandel, C. Carreras, Principles of Data Wrangling, O'Reilly, 2017.

[3] E. Z. Mark A. Beyer, Eric Thoo, Magic quadrant for data integration tools, Tech. rep., Gartner, g00340493 (July 2018).

[4] M. J. Franklin, A. Y. Halevy, D. Maier, From databases to dataspaces: a new abstraction for information management, SIGMOD Record 34 (4) (2005) 27–33. doi:10.1145/1107499.1107502.
URL http://doi.acm.org/10.1145/1107499.1107502

[5] C. Hedeler, K. Belhajjame, N. W. Paton, A. Campi, A. Fernandes, S. Embury, Dataspaces, in: Search Computing, Vol. 5950 of LNCS, Springer Berlin Heidelberg, 2010, pp. 114–134.
URL http://dx.doi.org/10.1007/978-3-642-12310-8_7

[6] V. Crescenzi, A. A. A. Fernandes, P. Merialdo, N. W. Paton, Crowdsourcing for data management, Knowl. Inf. Syst. 53 (1) (2017) 1–41. doi:10.1007/s10115-017-1057-x.
URL https://doi.org/10.1007/s10115-017-1057-x

[7] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, 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. URL `http://www.cidrdb.org/cidr2013/Papers/CIDR13_Paper28.pdf`

[8] N. Konstantinou, M. Koehler, E. Abel, C. Civili, B. Neumayr, E. Sallinger, A. A. A. Fernandes, G. Gottlob, J. A. Keane, L. Libkin, N. W. Paton, The VADA architecture for cost-effective data wrangling, in: ACM SIGMOD, 2017, pp. 1599–1602. `doi:10.1145/3035918.3058730`.
URL `http://doi.acm.org/10.1145/3035918.3058730`

[9] N. Konstantinou, E. Abel, L. Bellomarini, A. Bogatu, C. Civili, E. Irfanie, M. Koehler, L. Mazilu, E. Sallinger, A. A. A. Fernandes, G. Gottlob, J. A. Keane, N. W. Paton, VADA: an architecture for end user informed data preparation, J. Big Data 6 (2019) 74. `doi:10.1186/s40537-019-0237-9`.
URL `https://doi.org/10.1186/s40537-019-0237-9`

[10] M. Koehler, A. Bogatu, C. Civili, N. Konstantinou, E. Abel, A. A. A. Fernandes, J. A. Keane, L. Libkin, N. W. Paton, Data context informed data wrangling, in: 2017 IEEE International Conference on Big Data, BigData 2017, 2017, pp. 956–963. `doi:10.1109/BigData.2017.8258015`.
URL `https://doi.org/10.1109/BigData.2017.8258015`

[11] N. Konstantinou, N. W. Paton, Feedback driven improvement of data preparation pipelines, in: Proceedings of the 21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, 2019.
URL `http://ceur-ws.org/Vol-2324/Paper06-NKonstantinou.pdf`

[12] E. Abel, J. Keane, N. W. Paton, A. A. Fernandes, M. Koehler, N. Konstantinou, J. C. Cortes Rios, N. A. Azuan, S. M. Embury, User driven multi-criteria source selection, Information Sciences 430-431 (2018) 179–

199. doi:10.1016/j.ins.2017.11.019.
URL https://doi.org/10.1016/j.ins.2017.11.019

[13] L. Mazilu, N. W. Paton, A. A. Fernandes, M. Koehler, Dynamap: Schema mapping generation in the wild, in: Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM, 2019, pp. 37–48. doi:10.1145/3335783.3335785.
URL https://doi.org/10.1145/3335783.3335785

[14] W. Fan, F. Geerts, L. V. S. Lakshmanan, M. Xiong, Discovering conditional functional dependencies, Proceedings - International Conference on Data Engineering 23 (5). doi:10.1109/ICDE.2009.208.

[15] M. G. Bulmer, Principles of Statistics, Dover Publications, 1979.

[16] J. C. C. Ríos, N. W. Paton, A. A. A. Fernandes, K. Belhajjame, Efficient feedback collection for pay-as-you-go source selection, in: Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM 2016, Budapest, Hungary, July 18-20, 2016, 2016, pp. 1:1–1:12. doi:10.1145/2949689.2949690.
URL http://doi.acm.org/10.1145/2949689.2949690

[17] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, A. Sellers, Oxpath: A language for scalable data extraction, automation, and crawling on the deep web, The VLDB Journal 22 (1) (2013) 47–72. doi:10.1007/s00778-012-0286-6.
URL https://doi.org/10.1007/s00778-012-0286-6

[18] C. Hedeler, K. Belhajjame, L. Mao, C. Guo, I. Arundale, B. F. Lóscio, N. W. Paton, A. A. A. Fernandes, S. M. Embury, Dstoolkit: An architecture for flexible dataspace management, TLDKS 5 (2012) 126–157.

[19] X. L. Dong, A. Y. Halevy, C. Yu, Data integration with uncertainty, VLDB J. 18 (2) (2009) 469–500. doi:10.1007/s00778-008-0119-9.
URL http://dx.doi.org/10.1007/s00778-008-0119-9

[20] L. Blunschi, J. Dittrich, O. R. Girard, S. K. Karakashian, M. A. V. Salles, A dataspace odyssey: The imemex personal dataspace management system (demo), in: CIDR, 2007, pp. 114–119.
URL `http://cidrdb.org/cidr2007/papers/cidr07p13.pdf`

[21] V. Crescenzi, P. Merialdo, D. Qiu, Croudsourcing large scale wrapper inference, Distributed and Parallel Databases 33 (1) (2015) 95–122. `doi:10.1007/s10619-014-7163-9`.
URL `https://doi.org/10.1007/s10619-014-7163-9`

[22] N. Q. V. Hung, N. T. Tam, V. T. Chau, T. K. Wijaya, Z. Miklós, K. Aberer, A. Gal, M. Weidlich, SMART: A tool for analyzing and reconciling schema matching networks, in: 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015, 2015, pp. 1488–1491. `doi:10.1109/ICDE.2015.7113408`.
URL `https://doi.org/10.1109/ICDE.2015.7113408`

[23] C. J. Zhang, L. Chen, H. V. Jagadish, C. C. Cao, Reducing uncertainty of schema matching via crowdsourcing, PVLDB 6 (9) (2013) 757–768. `doi:10.14778/2536360.2536374`.
URL `http://www.vldb.org/pvldb/vol6/p757-zhang.pdf`

[24] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, C. Hedeler, Feedback-based annotation, selection and refinement of schema mappings for dataspaces, in: EDBT, 2010, pp. 573–584. `doi:10.1145/1739041.1739110`.
URL `http://doi.acm.org/10.1145/1739041.1739110`

[25] A. Bonifati, R. Ciucanu, S. Staworko, Interactive inference of join queries, in: 17th International Conference on Extending Database Technology, EDBT, 2014, pp. 451–462. `doi:10.5441/002/edbt.2014.41`.
URL `https://doi.org/10.5441/002/edbt.2014.41`

[26] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. C. N. Pereira, S. Guha, Learning to create data-integrating queries, PVLDB 1 (1)

(2008) 785–796. doi:10.14778/1453856.1453941.
URL http://www.vldb.org/pvldb/1/1453941.pdf

[27] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, X. Zhu, Corleone: hands-off crowdsourcing for entity matching, in: SIGMOD, 2014, pp. 601–612. doi:10.1145/2588555.2588576.
URL http://doi.acm.org/10.1145/2588555.2588576

[28] R. Maskat, N. W. Paton, S. M. Embury, Pay-as-you-go configuration of entity resolution, T. Large-Scale Data- and Knowledge-Centered Systems 29 (2016) 40–65. doi:10.1007/978-3-662-54037-4_2.
URL https://doi.org/10.1007/978-3-662-54037-4_2

[29] Z. Yan, N. Zheng, Z. G. Ives, P. P. Talukdar, C. Yu, Active learning in keyword search-based data integration, VLDB J. 24 (5) (2015) 611–631. doi:10.1007/s00778-014-0374-x.
URL https://doi.org/10.1007/s00778-014-0374-x

[30] G. Li, J. Wang, Y. Zheng, M. J. Franklin, Crowdsourced data management: A survey, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2296–2319. doi:10.1109/TKDE.2016.2535242.
URL https://doi.org/10.1109/TKDE.2016.2535242

[31] S. Kandel, A. Paepcke, J. M. Hellerstein, J. Heer, Wrangler: Interactive visual specification of data transformation scripts, in: CHI, 2011, pp. 3363–3372.

[32] S. Gulwani, W. R. Harris, R. Singh, Spreadsheet data manipulation using examples, Commun. ACM 55 (8) (2012) 97–105. doi:10.1145/2240236.2240260.
URL http://doi.acm.org/10.1145/2240236.2240260

[33] B. Alexe, B. ten Cate, P. G. Kolaitis, W. C. Tan, Designing and refining schema mappings via data examples, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD, 2011, pp.

133–144. doi:10.1145/1989323.1989338.
URL http://doi.acm.org/10.1145/1989323.1989338

[34] G. Gottlob, P. Senellart, Schema mapping discovery from data instances, JACM 57 (2) (2010) 6:1–6:37. doi:10.1145/1667053.1667055.
URL http://doi.acm.org/10.1145/1667053.1667055

[35] F. Psallidas, B. Ding, K. Chakrabarti, S. Chaudhuri, S4: top-k spreadsheet-style search for query discovery, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015, 2015, pp. 2001–2016. doi:10.1145/2723372.2749452.
URL http://doi.acm.org/10.1145/2723372.2749452

[36] O. Andersen, C. Thomsen, K. Torp, Simpleetl: ETL processing by simple specifications, in: Proceedings of the 20th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data co-located with 10th EDBT/ICDT Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26-29, 2018., 2018.
URL http://ceur-ws.org/Vol-2062/paper10.pdf

[37] C. Thomsen, T. B. Pedersen, pygrametl: a powerful programming framework for extract-transform-load programmers, in: DOLAP 2009, ACM 12th International Workshop on Data Warehousing and OLAP, Hong Kong, China, November 6, 2009, Proceedings, 2009, pp. 49–56. doi:10.1145/1651291.1651301.
URL http://doi.acm.org/10.1145/1651291.1651301

[38] V. Theodorou, A. Abelló, M. Thiele, W. Lehner, Frequent patterns in ETL workflows: An empirical approach, Data Knowl. Eng. 112 (2017) 1–16. doi:10.1016/j.datak.2017.08.004.
URL https://doi.org/10.1016/j.datak.2017.08.004

[39] K. Tomingas, M. Kliimask, T. Tammet, Data integration patterns for data warehouse automation, in: 18th East European Conference on Advances

in Databases and Information Systems, ADBIS, 2014, pp. 41–55. doi:10.1007/978-3-319-10518-5_4.
URL https://doi.org/10.1007/978-3-319-10518-5_4

[40] D. Butkevicius, P. D. Freiberger, F. M. Halberg, MAIME: A maintenance manager for ETL processes, in: Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference (EDBT/ICDT 2017), Venice, Italy, March 21-24, 2017., 2017.
URL http://ceur-ws.org/Vol-1810/DOLAP_paper_08.pdf

[41] S. M. F. Ali, R. Wrembel, From conceptual design to performance optimization of ETL workflows: current state of research and open problems, VLDB J. 26 (6) (2017) 777–801. doi:10.1007/s00778-017-0477-2.
URL https://doi.org/10.1007/s00778-017-0477-2

[42] G. Kougka, A. Gounaris, A. Simitsis, The many faces of data-centric workflow optimization: a survey, I. J. Data Science and Analytics 6 (2) (2018) 81–107. doi:10.1007/s41060-018-0107-0.
URL https://doi.org/10.1007/s41060-018-0107-0

[43] S. Krishnan, E. Wu, Alphaclean: Automatic generation of data cleaning pipelines, CoRR abs/1904.11827. arXiv:1904.11827.
URL http://arxiv.org/abs/1904.11827

[44] B. Settles, Active Learning, Morgan & Claypool, 2012.

[45] K. Bellare, S. Iyengar, A. G. Parameswaran, V. Rastogi, Active sampling for entity matching, in: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, 2012, pp. 1131–1139. doi:10.1145/2339530.2339707.
URL https://doi.org/10.1145/2339530.2339707

[46] G. Demartini, D. E. Difallah, P. Cudré-Mauroux, Large-scale linked data integration using probabilistic reasoning and crowdsourcing, VLDBJ 22 (5)

(2013) 665–687. doi:10.1007/s00778-013-0324-z.
URL http://dx.doi.org/10.1007/s00778-013-0324-z

[47] S. E. Whang, P. Lofgren, H. Garcia-Molina, Question selection for crowd entity resolution, PVLDB 6 (6) (2013) 349–360.
URL http://www.vldb.org/pvldb/vol6/p349-whang.pdf

[48] J. Wang, T. Kraska, M. J. Franklin, J. Feng, Crowder: Crowdsourcing entity resolution, PVLDB 5 (11) (2012) 1483–1494. doi:10.14778/2350229.2350263.
URL http://vldb.org/pvldb/vol5/p1483_jiannanwang_vldb2012.pdf

[49] J. C. C. Ríos, N. W. Paton, A. A. A. Fernandes, E. Abel, J. A. Keane, Crowdsourced targeted feedback collection for multicriteria data source selection, J. Data and Information Quality 11 (1) (2019) 2:1–2:27. doi:10.1145/3284934.
URL https://doi.org/10.1145/3284934

Algorithm 1 Apply collected feedback

```
1: function ApplyFeedback
2:     for all $m \in Matches$ do
3:         if significantlyWorse($m.a$, $T.a \setminus m.a$) then
4:             $Matches$.remove($m$)
5:         end if
6:     end for
7:     $Mappings \leftarrow$ MappingGeneration($Matches, profileData$)
8:     for all $map \in Mappings$ do
9:         if significantlyWorse($map$, $T \setminus map$) then
10:            $Mappings$.remove($map$)
11:        end if
12:    end for
13:    $endProduct \leftarrow$ MappingSelection($Mappings$)
14:    for all $cfd \in CFDs$ do
15:        $s \leftarrow$ ViolatingTuples($T$, $cfd$)
16:        if significantlyWorse($s$, $T \setminus s$) then
17:            $CFDs$.remove($cfd$)
18:        end if
19:    end for
20:    $endProduct \leftarrow$ DataRepair($CFDs$)
21:    return $endProduct$
22: end function
```

Algorithm 2 Apply collected feedback

```
function ApplyFeedbackCautiously(endProduct)
    $maxPrecisionF \leftarrow$ precisionF($endProduct$)
    $newEndProduct \leftarrow$ ApplyFeedback
    $newPrecisionF \leftarrow$ precisionF($newEndProduct$)
    if $newPrecisionF \geq maxPrecisionF$ then
        $endProduct \leftarrow newEndProduct$
    end if
    return $endProduct$
end function
```

Algorithm 3 Identify data on which to collect feedback.

```
    function TargetedFeedback
        $fbTarget = \{\}$
3:      for all $m \in Matches$ do
            if not(significantlyWorse($m.a$, $T.a \setminus m.a$) or significantlyBetter($m.a$, $T.a \setminus m.a$)) then
                $fbTarget = fbTarget \cup$ values originating from $m.a$
6:          end if
        end for
        for all $map \in Mappings$ do
9:          if not(significantlyWorse($map$, $T \setminus map$) or significantlyBetter($map$, $T \setminus map$)) then
                $fbTarget = fbTarget \cup$ values originating from $map$
            end if
12:     end for
        for all $cfd \in CFDs$ do
            $s \leftarrow$ ViolatingTuples($T$, $cfd$)
15:         if not(significantlyWorse($s$, $T \setminus s$) or significantlyBetter($s$, $T \setminus s$)) then
                $fbTarget = fbTarget \cup$ s
            end if
18:     end for
        return $fbTarget$
    end function
```