# Feedback Driven Improvement of Data Preparation Pipelines

Nikolaos Konstantinou
School of Computer Science
University of Manchester
Manchester, UK
nikolaos.konstantinou@manchester.ac.uk

Norman W. Paton
School of Computer Science
University of Manchester
Manchester, UK
norman.paton@manchester.ac.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. Furthermore, the approach takes into account all the available feedback in determining which actions should be applied to refine the data preparation process. The approach has been implemented to refine the results of 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.

## 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 [34], and for more *ad hoc* analyses carried out by data scientists may be referred to as *data wrangling* [27]. 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 [24], most of these products are essentially visual programming platforms, in which users make many, fine-grained decisions. The consequences 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].

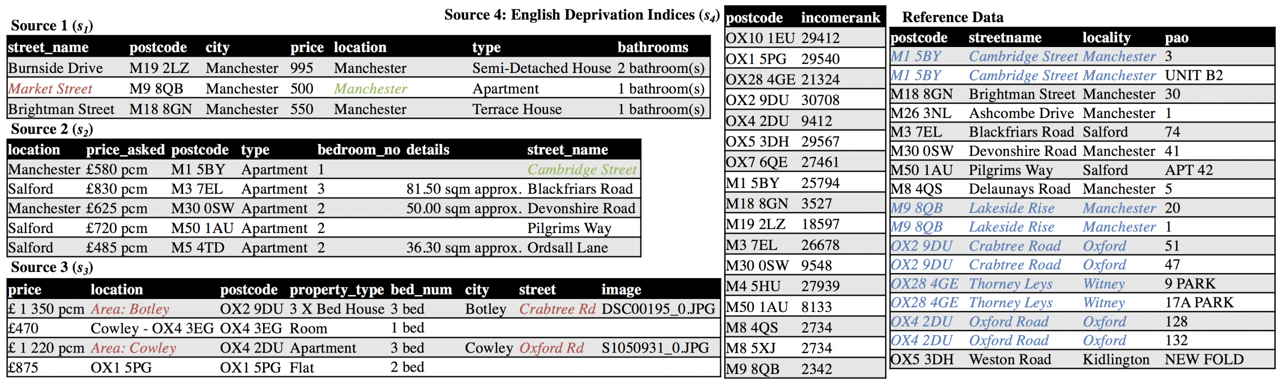
The high cost of data preparation has been recognised for a considerable period. For example, research into dataspaces [12] proposed a pay-as-you-go approach to data integration, in which there was an initial and automated bootstrapping phase, which was followed by an incremental improvement phase in 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 [17], which in turn led to proposals for the use of crowds as a possible source of feedback [9]. 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 [29], 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 [21], 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 [20]. 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 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]https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/33d344256f63



**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 | | | |

**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 |

**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 |

**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.**

(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) A realisation of the technique from (1) 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.

(3) An empirical evaluation of the implementation of the approach from (2) that investigates the effectiveness of the proposed approach both for individual data preparation constructs (matches, mappings and 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 the feedback assimilation algorithm. Section 5 evaluates the technique in a real estate application. Section 6 reviews other approaches to increasing the cost-effectiveness of data preparation, and Section 7 concludes.

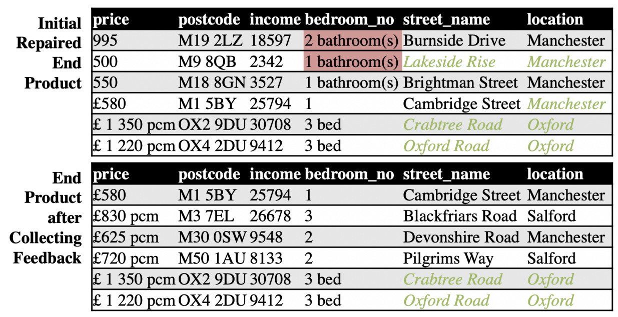
## 2 A DATA PREPARATION PIPELINE

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 [20, 21].

### 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 $s_4$).



| | price | postcode | income | bedroom_no | street_name | location |
|---|---|---|---|---|---|---|
| **Initial Repaired End Product** | 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* |

| | price | postcode | income | bedroom_no | street_name | location |
|---|---|---|---|---|---|---|
| **End Product after Collecting Feedback** | £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.**

**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 [1]. The user's requirements are modelled as a weighted set of criteria, with the sum of their weights equal to one. Although in general different quality criteria can be used (such as completeness, consistency or relevance), in our running example, we consider 6 criteria, each one on the correctness 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 [20]. 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 automatically produce the first end data product

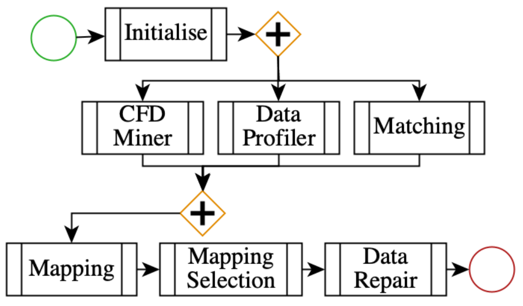


**Figure 3: A Data Preparation Pipeline**

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:

**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 searches the space of possible ways of combining the sources using unions or joins, producing a set of candidate mappings.

**Mapping Selection:** Given a set of user criteria, a set of candidate mappings, the target schema and a targeted size, mapping selection establishes how many tuples to obtain from each of the candidate mappings to populate a target schema instance, thus creating an end data product [1].

**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 [11]. 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] → [streetname],(M1 5BY || Cambridge Street))
property([postcode] → [locality],(M9 8QB || Manchester))
property([postcode] → [streetname],(OX2 9DU || Crabtree Road))
property([postcode] → [locality],(OX28 4GE || Witney))
property([postcode] → [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 [11]. 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.

> 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 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[2].

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; and (iii) a format transformation has introduced an error into the value.
(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

[2]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 any of these attribute values are marked as *FP*.

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.

## 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 CFDs. 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; and Section 4.4 describes how the different aspects of the solution are brought together in an algorithm.

### 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 here 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. The question then is *when do we provide credence to a hypothesis.* Given the evidence available in the form of feedback, hypotheses are tested as follows:

**Matching:** 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.

**Mapping Generation:** If the result of a mapping is significantly worse than the results from the other mappings, then the mapping is suspicious.

**Data Repair:** If the repaired values from a repair rule are significantly worse than the other values for the repaired attribute, then the rule is suspicious.

The following section uses statistical techniques to compare matches, mappings and feedback using evidence from feedback.

### 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). In the examples and experiments in this paper, feedback is on correctness. For criteria that are based on feedback, Equation 1 is used, in which $\hat{c_s}$ is the criterion evaluation 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.

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

Thus, in the absence of feedback, the default value for a criterion is *0.5*. However, if we have feedback on the correctness of an attribute that there are *6 tp* values and *3 fp* values, then if there are *100* tuples, the estimated correctness for the attribute in $s$ is now $\frac{1}{2}(1 + \frac{6-3}{|100|}) = 0.515$.

We now proceed to establish when criteria estimates are significantly different using standard statistical techniques [7][3]. 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. For instance, a z-score of zero indicates that the value and the mean are identical. In our experiments we use the z-score that corresponds to a confidence level of 80%, i.e. 1.282. 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 values, and

[3]Such statistical techniques have been used before to compare mappings on the basis of feedback, with a view to targeting feedback collection [28]. 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.

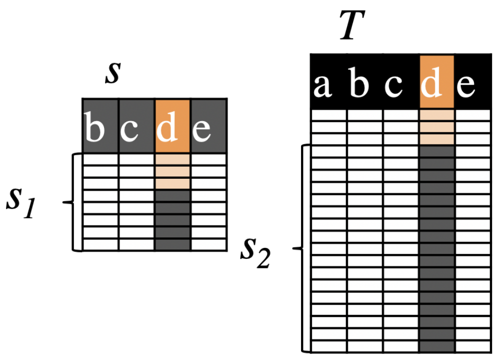


**Figure 4: Detecting significantly different matches**

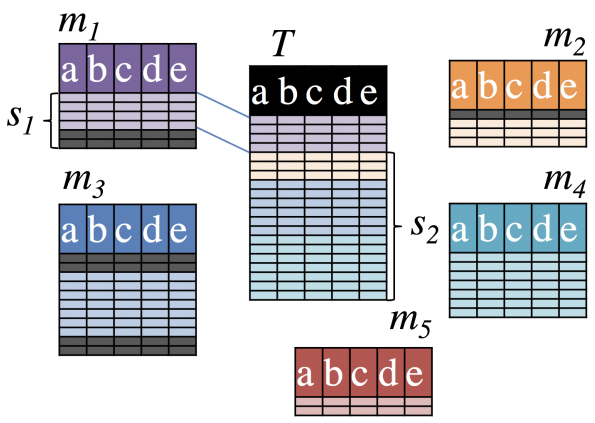


**Figure 5: Detecting significantly different mappings**

we only consider tuple-level feedback instances when evaluating $L_s$ on a set of tuples.

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:

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

Next, we discuss the comparisons we used when testing the different components in the system. The focus is on what is considered as $s_1$, $s_2$, when evaluating Equation 2.

**Testing matches for significant difference**. Figure 4 shows the target schema $T$ and a source $s$, with a match between attribute $s.d$ and the target schema attribute $T.d$: $m_1 : s.d \sim T.d$.

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 value sets $s_1$ and $s_2$ (i.e., $s.d$ and $T.d$), as illustrated in Figure 4.

Note that the set $s_1$ is the complete set of values for attribute $s.d$, regardless of whether the values are selected to populate the end product or not, while $s_2$ is the greyed part of attribute $T.d$.

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 $\frac{1}{2}(1 + \frac{0-2}{|3|}) = 0.167$. The correctness for the values obtained from other matches on the same column, on which no feedback has been collected, is estimated 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**. 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 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, for to evaluate whether mapping $m_1$ is significantly different, we use $s_1$ and $s_2$ 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, and on the mappings themselves. Furthermore, the Mapping Selection component ensures that the tuples that are marked as true (resp. false) positives are selected (resp. not selected).

**Testing CFDs for significant difference**. In Figure 6 we see the result of a CFD on the end product. We mark as green the *2* attribute values for column $c$ that are found to be correct, and with red the *3* values in column $d$ found to be violating the CFD, and that were thus repaired. As before, we consider tuple sets $s_1$ and $s_2$ in Figure 6 as inputs to Equation 2. Note that the correct values are not considered in $s_1$, as they have no effect on the end product. We only consider the tuples that were modified.

## 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 CFD rule. The hypotheses and respective actions per component are thus summarised below:

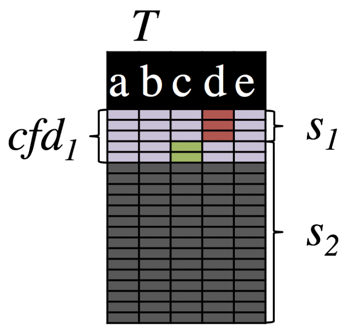


**Figure 6: Detecting significantly different CFDs**

**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: end function
```

- **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 rule.

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 only focus on removing suspicious items.

## 4.4 Algorithm

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 data preparation pipeline.

Function significantlyWorse (lines 3, 9 and 16) is essentially testing Equation 2 with $s_1$ and $s_2$ as arguments, and returns true if it holds, or false otherwise. The arguments for this function are the ones illustrated as $s_1$ and $s_2$ in Figure 4 when detecting significantly worse matches, Figure 5 when detecting significantly worse mappings, and Figure 6 when detecting significantly worse rules. 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 $cfd$.

## 5 EVALUATION

### 5.1 Experimental Setup

For the evaluation, we used as sources the following datasets: (a) forty datasets with real-estate properties extracted from the web using OXpath [13], with contents similar to sources $s_1$ to $s_3$ in Figure 1, (b) English indices of deprivation data, downloaded from www.gov.uk, as shown in $s_4$ in Figure 1. As reference data, we used open address data from openaddressesuk.org.

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 5.

Each of the criteria in the user context was 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 7. 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.

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.

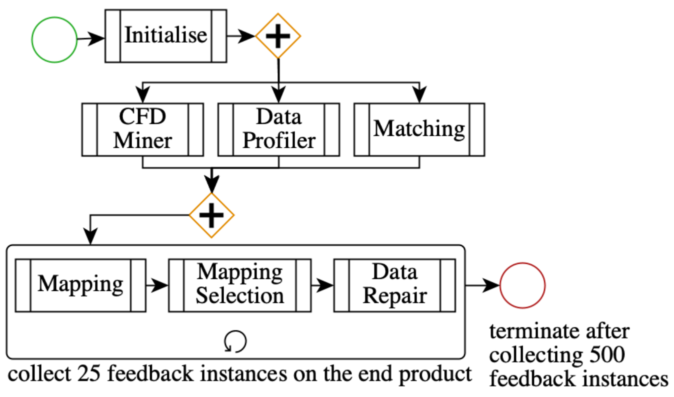

Initialise
CFD Miner
Data Profiler
Matching
Mapping
Mapping Selection
Data Repair
collect 25 feedback instances on the end product
terminate after collecting 500 feedback instances

**Figure 7: Experiments**

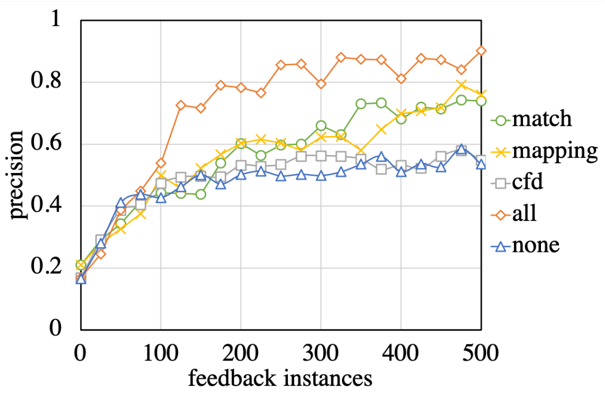


**Figure 8: Precision for different configurations**

- 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.

## 5.2 Results

The precision values obtained for different configurations are illustrated in Figure 8[4]. Each of the lines in Figure 8 is the average of 5 runs from 0 to 500 randomly collected feedback instances, using the configurations from Section 5.1. 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 8.

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.

Notice in Figure 8 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 *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 this experiment: 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.

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 Figure 8. It is unclear from this experiment which one has more impact.

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

- The rules being learnt are numerous, e.g., 3526 rules were learnt with a support size of 5. As a result, each rule 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 rule.
- Each of the rules typically has a small effect to the end product. As such, even when a problematic rule is removed, this tends not to have much of an effect on overall quality.

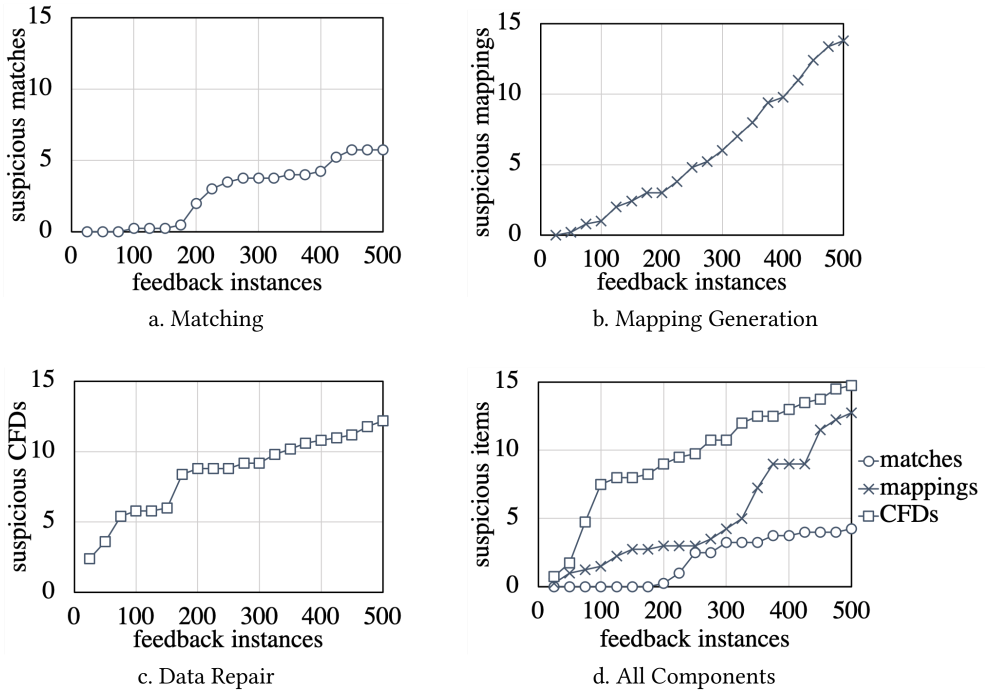
However, it is still clear that identifying CFDs that introduced errors to the data, and discarding them, has the potential to have a positive effect on the resulting quality.

The most important result for the overall proposal, however, is that when the actions across all the components are combined, this provides much the greatest improvement compared with any of the individual means of applying the feedback. Furthermore, much of this improvement is obtained with modest amounts of feedback.

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

Next, we report on the number of suspicious items discovered in each of the experiments. Each line in Figure 9, corresponds to the average of the same 5 runs as reported in Figure 8. Figures 9a to c, respectively show the numbers of suspicious matches,

[4]In a setting where mapping selection retains the top *1000* tuples from a ground truth of around *4500* tuples, the recall tends to increase broadly in line with the precision.



a. Matching b. Mapping Generation c. Data Repair d. All Components

**Figure 9: Detecting Suspicious Products**

mappings and CFDs considered, when only the respective component is being tested. Figure 9d shows the overall number of suspicious items discovered and discarded when all components were being tested.

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.

From Figures 8 and 9 it is clear that taking feedback into account and acting on all components increases the return on the investment from feedback. When considering all possible actions together, the overall benefit is greater, and the benefit accrues with smaller amounts of feedback.

## 6 RELATED WORK

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

In *pay-as-you-go* data preparation, following from the vision of dataspaces [12], a best-effort integration is produced through automated bootstrapping, which is refined in the light of user feedback. There have been many proposals for pay-as-you-go data preparation components, for example relating to data extraction [10], matching [18, 36], mapping [5, 6, 30] and entity resolution [14, 25]. Such proposals have addressed issues such as targeting the most appropriate feedback [10, 28, 35], and accommodating unreliable feedback, in particular in the context of crowdsourcing [9, 23]. However, such work has primarily used a single type of feedback to inform a single data preparation step.

In relation to *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 [5], 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 [14], 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 [25] as part of a single optimisation step that configures together blocking, detailed comparison and clustering. This contrasts with the current paper in focusing 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.

In terms of *reducing manual effort in data preparation*, there are a variety of different approaches. 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 [19] can suggest potential transformation programs, and FlashFill [16] can synthesize transformation programs from examples. There are also proposals in which mappings are discovered based on data instances (e.g., [2, 15, 26]). 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 [4, 32] or patterns [31, 33] that support recurring data preparation behaviours, techniques for managing evolution of ETL programs [8], and development of ETL processes that abstract over more concrete implementation details [3, 22]. 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 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 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) *A realisation of the technique from (1) 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 technique 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.

(3) *An empirical evaluation of the implementation of the approach from (2) that investigates the effectiveness of the proposed approach both for individual data preparation constructs (matches, mappings and CFDs) 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.

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.

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

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

## REFERENCES

[1] Edward Abel, John Keane, Norman W. Paton, Alvaro A.A. Fernandes, Martin Koehler, Nikolaos Konstantinou, Julio Cesar Cortes Rios, Nurzety A. Azuan, and Suzanne M. Embury. 2018. User driven multi-criteria source selection. *Information Sciences* 430-431 (2018), 179–199. https://doi.org/10.1016/j.ins.2017.11.019

[2] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. 2011. Designing and refining schema mappings via data examples. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*. 133–144. https://doi.org/10.1145/1989323.1989338

[3] Syed Muhammad Fawad Ali and Robert Wrembel. 2017. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. *VLDB J.* 26, 6 (2017), 777–801. https://doi.org/10.1007/s00778-017-0477-2

[4] Ove Andersen, Christian Thomsen, and Kristian Torp. 2018. 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.* http://ceur-ws.org/Vol-2062/paper10.pdf

[5] Khalid Belhajjame, Norman W. Paton, Suzanne M. Embury, Alvaro A. A. Fernandes, and Cornelia Hedeler. 2010. Feedback-based annotation, selection and refinement of schema mappings for dataspaces. In *EDBT*. 573–584. https://doi.org/10.1145/1739041.1739110

[6] Angela Bonifati, Radu Ciucanu, and Slawek Staworko. 2014. Interactive Inference of Join Queries. In *17th International Conference on Extending Database Technology, EDBT*. 451–462. https://doi.org/10.5441/002/edbt.2014.41

[7] Michael G. Bulmer. 1979. *Principles of Statistics.* Dover Publications.

[8] Darius Butkevicius, Philipp D. Freiberger, and Frederik M. Halberg. 2017. 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.* http://ceur-ws.org/Vol-1810/DOLAP_paper_08.pdf

[9] Valter Crescenzi, Alvaro A. A. Fernandes, Paolo Merialdo, and Norman W. Paton. 2017. Crowdsourcing for data management. *Knowl. Inf. Syst.* 53, 1 (2017), 1–41. https://doi.org/10.1007/s10115-017-1057-x

[10] Valter Crescenzi, Paolo Merialdo, and Disheng Qiu. 2015. Crowdsourcing large scale wrapper inference. *Distributed and Parallel Databases* 33, 1 (2015), 95–122. https://doi.org/10.1007/s10619-014-7163-9

[11] Wenfei Fan, Floris Geerts, Laks V S Lakshmanan, and Ming Xiong. 2011. Discovering conditional functional dependencies. *Proceedings - International Conference on Data Engineering* 23, 5 (2011). https://doi.org/10.1109/ICDE.2009.208

[12] Michael J. Franklin, Alon Y. Halevy, and David Maier. 2005. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record* 34, 4 (2005), 27–33. https://doi.org/10.1145/1107499.1107502

[13] Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart, and Andrew Sellers. 2013. OXPath: A language for scalable data extraction, automation, and crawling on the deep web. *The VLDB Journal* 22, 1 (01 Feb 2013), 47–72. https://doi.org/10.1007/s00778-012-0286-6

[14] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude W. Shavlik, and Xiaojin Zhu. 2014. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*. 601–612. https://doi.org/10.1145/2588555.2588576

[15] Georg Gottlob and Pierre Senellart. 2010. Schema Mapping Discovery from Data Instances. *JACM* 57, 2 (2010), 6:1–6:37. https://doi.org/10.1145/1667053.1667055

[16] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (2012), 97–105. https://doi.org/10.1145/2240236.2240260

[17] Cornelia Hedeler, Khalid Belhajjame, Norman W. Paton, Alessandro Campi, AlvaroA.A. Fernandes, and SuzanneM. Embury. 2010. Dataspaces. In *Search Computing*. LNCS, Vol. 5950. Springer Berlin Heidelberg, 114–134. http://dx.doi.org/10.1007/978-3-642-12310-8_7

[18] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Vinh Tuan Chau, Tri Kurniawan Wijaya, Zoltán Miklós, Karl Aberer, Avigdor Gal, and Matthias Weidlich. 2015. 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.* 1488–1491. https://doi.org/10.1109/ICDE.2015.7113408

[19] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *CHI*. 3363–3372.

[20] Martin Koehler, Alex Bogatu, Cristina Civili, Nikolaos Konstantinou, Edward Abel, Alvaro A. A. Fernandes, John A. Keane, Leonid Libkin, and Norman W. Paton. 2017. Data context informed data wrangling. In *2017 IEEE International Conference on Big Data, BigData 2017.* 956–963. https://doi.org/10.1109/BigData.2017.8258015

[21] Nikolaos Konstantinou, Martin Koehler, Edward Abel, Cristina Civili, Bernd Neumayr, Emanuel Sallinger, Alvaro A. A. Fernandes, Georg Gottlob, John A. Keane, Leonid Libkin, and Norman W. Paton. 2017. The VADA Architecture for Cost-Effective Data Wrangling. In *ACM SIGMOD*. 1599–1602. https://doi.org/10.1145/3035918.3058730

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

[23] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J. Franklin. 2016. Crowdsourced Data Management: A Survey. *IEEE Trans. Knowl. Data Eng.* 28, 9 (2016), 2296–2319. https://doi.org/10.1109/TKDE.2016.2535242

[24] Ehtisham Zaidi Mark A. Beyer, Eric Thoo. 2018. *Magic Quadrant for Data Integration Tools.* Technical Report. Gartner. G00340493.

[25] Ruhaila Maskat, Norman W. Paton, and Suzanne M. Embury. 2016. Pay-as-you-go Configuration of Entity Resolution. *T. Large-Scale Data- and Knowledge-Centered Systems* 29 (2016), 40–65. https://doi.org/10.1007/978-3-662-54037-4_2

[26] Fotis Psallidas, Bolin Ding, Kaushik Chakrabarti, and Surajit Chaudhuri. 2015. 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*. 2001–2016. https://doi.org/10.1145/2723372.2749452

[27] Tye Rattenbury, Joe Hellerstein, Jeffery Heer, Sean Kandel, and Conner Carreras. 2017. *Principles of Data Wrangling*. O'Reilly.

[28] Julio César Cortés Ríos, Norman W. Paton, Alvaro A. A. Fernandes, and Khalid Belhajjame. 2016. 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*. 1:1–1:12. https://doi.org/10.1145/2949689.2949690

[29] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Shan Xu. 2013. 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.* http://www.cidrdb.org/cidr2013/Papers/CIDR13_Paper28.pdf

[30] Partha Pratim Talukdar, Marie Jacob, Muhammad Salman Mehmood, Koby Crammer, Zachary G. Ives, Fernando C. N. Pereira, and Sudipto Guha. 2008. Learning to create data-integrating queries. *PVLDB* 1, 1 (2008), 785–796. https://doi.org/10.14778/1453856.1453941

[31] Vasileios Theodorou, Alberto Abelló, Maik Thiele, and Wolfgang Lehner. 2017. Frequent patterns in ETL workflows: An empirical approach. *Data Knowl. Eng.* 112 (2017), 1–16. https://doi.org/10.1016/j.datak.2017.08.004

[32] Christian Thomsen and Torben Bach Pedersen. 2009. 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.* 49–56. https://doi.org/10.1145/1651291.1651301

[33] Kalle Tomingas, Margus Kliimask, and Tanel Tammet. 2014. Data Integration Patterns for Data Warehouse Automation. In *18th East European Conference on Advances in Databases and Information Systems, ADBIS*. 41–55. https://doi.org/10.1007/978-3-319-10518-5_4

[34] P. Vassiliadis. 2011. A Survey of Extract-Transform-Load Technology. *IJDWM* 5, 3 (2011), 1–27.

[35] Zhepeng Yan, Nan Zheng, Zachary G. Ives, Partha Pratim Talukdar, and Cong Yu. 2015. Active learning in keyword search-based data integration. *VLDB J.* 24, 5 (2015), 611–631. https://doi.org/10.1007/s00778-014-0374-x

[36] Chen Jason Zhang, Lei Chen, H. V. Jagadish, and Caleb Chen Cao. 2013. Reducing Uncertainty of Schema Matching via Crowdsourcing. *PVLDB* 6, 9 (2013), 757–768. https://doi.org/10.14778/2536360.2536374