Incorporating Data Context to Cost-Effectively Automate End-to-End Data Wrangling

Citation for published version:
Koehler, M, Abel, E, Bogatu, A, Civili, C, Mazilu, L, Konstantinou, N, Fernandes, AAA, Keane, J, Libkin, L & Paton, NW 2021, 'Incorporating Data Context to Cost-Effectively Automate End-to-End Data Wrangling', *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 169-186. https://doi.org/10.1109/TBDATA.2019.2907588

Digital Object Identifier (DOI):
10.1109/TBDATA.2019.2907588

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
IEEE Transactions on Big Data

Incorporating Data Context to Cost-Effectively Automate End-to-End Data Wrangling

Martin Koehler, *Member, IEEE*, Edward Abel, Alex Bogatu, Cristina Civili, Lacramioara Mazilu, Nikolaos Konstantinou, *Member, IEEE*, Alvaro A.A. Fernandes, John Keane, Leonid Libkin, and Norman W. Paton

Abstract—The process of preparing potentially large and complex data sets for further analysis or manual examination is often called data wrangling. In classical warehousing environments, the steps in such a process are carried out using Extract-Transform-Load platforms, with significant manual involvement in specifying, configuring or tuning many of them. In typical big data applications, we need to ensure that all wrangling steps, including web extraction, selection, integration and cleaning, benefit from automation wherever possible. Towards this goal, in the paper we: (i) introduce a notion of data context, which associates portions of a target schema with extensional data of types that are commonly available; (ii) define a scalable methodology to bootstrap an end-to-end data wrangling process based on data profiling; (iii) describe how data context is used to inform automation in several steps within wrangling, specifically, matching, value format transformation, data repair, and mapping generation and selection to optimise the accuracy, consistency and relevance of the result; and (iv) we evaluate the approach with real estate data and financial data, showing substantial improvements in the results of automated wrangling.

Index Terms—Data wrangling, data matching, mapping generation, data transformation, data cleaning, source selection

1 Introduction

In the past decade managing, processing and analysing data has changed radically towards establishing data-driven organisations. This shift from a world where most data used by companies and organisations was regularly structured, neatly organised in relational databases, and treated as complete, towards data-driven organisations, necessitates novel and principled methodologies for managing multi-modal and potentially uncertain data at scale[1].

Data wrangling is the process by which such potentially large and complex data sets in data-driven organisations are prepared for analysis or manual examination [2], [3]. In the big data era[4], data wrangling must take place in the context of the four Vs of big data that represent challenges (volume, velocity, variety and veracity) if the fifth V that represents the reward for overcoming the challenges (value) is to be obtained. However, there may be quite a few steps involved in data wrangling that necessitate principled methods tackling the above challenges of big data. A common process involves discovering and selecting appropriate data sources, extracting unstructured text and semi-structured data from the deep Web or Web tables, matching the extracted data with a schema or ontology, generating data transformations integrating data from several sources, repairing and cleaning incomplete and inconsistent data, transforming values for attributes from sources into a uniform format, and resolution and fusion of entities.

Such steps can be carried out using traditional Extract-Transform-Load (ETL) or Big Data analytics platforms [5], [6],[7], both requiring significant manual involvement in specifying, configuring, programming or tuning many of the steps [8], [9]. It is widely reported that intense manual involvement in such processes is expensive (e.g., [10]), often representing more than half the time of data scientists. As the numbers of data sources within organisations and in the public domain grows, there is an increasingly pressing need for cost-effective, scalable and principled techniques for addressing the variety and veracity of big data to increase the value of the wrangling result.

We study the problem of cost-effectively automating an end-to-end data wrangling process, that is, to integrate (addressing variety), clean (addressing veracity), select from a large set of input sources (addressing volume) and create a data product that is suitable for downstream analysis by optimising its quality (addressing value). In more detail, we focus on how a wrangling process can be automated and improved by *data context* [11]: data from the domain in which wrangling is taking place [12]. There have been proposals tailored to a specific type of auxiliary information and for automating individual steps in the

- M. Koehler, E. Abel, A. Bogatu, L. Mazilu, N. Konstantinou, A.A.A. Fernandes, and N.W. Paton are with the Information Management Group, School of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom. E-mail: koehler.martin@gmail.com, {edward.abel, alex.bogatu, lara.mazilu, nikolaos.konstantinou, a.fernandes, norman.paton}@manchester.ac.uk.
- C. Civili and L. Libkin are with the School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, United Kingdom. E-mail: {ccivili, libkin}@inf.ed.ac.uk.
- J. Keane is with the Software Systems Group, School of Computer Science, University of Manchester, Manchester, M13 9PL, United Kingdom. E-mail: john.keane@manchester.ac.uk.

Manuscript received 5 Feb. 2018; revised 14 Aug. 2018; accepted 19 Mar. 2019. Date of publication 15 Apr. 2019; date of current version 1 Mar. 2021. (Corresponding author: Martin Koehler.)
Recommended for acceptance by W. Chang.
Digital Object Identifier no. 10.1109/TBDATA.2019.2907588

wrangling process (e.g., [13], [14], [15], [16]), but there is a need to be more systematic and holistic, ensuring that all the steps can be automated, the process can be scaled [17], and that all these steps make use of the available data context. We observe that individual proposals for automating steps within the wrangling process tend not to use many types of auxiliary data. All of these proposals have merit, but are tailored to a specific type of auxiliary information and do not take into account end-to-end wrangling but focus on individual steps in isolation.

Automating the end-to-end process supports the on-demand population of the data product specified by the user of the system, but potentially results in a data product of limited quality. Data context, such as master data, reference data, or example entities from the domain of interest, can serve as a guide to improve the results of many steps within the wrangling process. Specifically, the claim is that a small number of often readily available types of contextual data can substantially improve the quality of the automatically produced data product. An important claim of our approach is that although data context types may incorporate more specific types of input provided by the user (e.g., data examples in [18]), data context mainly relates to auxiliary information that is already available, for instance in the company or via open data portals. Thus, an implied note is that we do not require auxiliary data to be correct or complete, but present an approach that is capable of dealing with potentially erroneous instances. Another claim of the proposed approach is that diverse data context types are capable of improving different dimensions of quality[19] of the wrangling result. For instance, improving the *accuracy*, the closeness of a value to the elements of the corresponding domain, the *consistency* with respect to the given external knowledge, and the *relevance* of the wrangling result with respect to the user provided input, of the wrangling result necessitates the establishment of a holistic methodology taking into account multiple data wrangling steps and data context types.

Our solution adapts and extends a conference paper [11] by refining the whole wrangling methodology, including an additional wrangling stage and substantially enhancing the evaluation. We build upon some of the latest techniques from the data profiling, integration and cleaning communities on dependency discovery [20], [21], [22], user-driven multi-criteria source selection[23], instance-based schema matching [15], [16], mapping generation and validation, value format transformations [24], and rule-based repair [25]. We extend and refine these approaches to use target instances in automation, to provide a comprehensive, end-to-end approach incorporating instance-based evidence from multiple sources in the *data context* that may be partial or spurious. Using this notion we define a domain-independent methodology to apply data context on a potentially large set of steps and specific methods to inform a concrete set of individual steps, with the objective of improving the quality of the wrangling result.

In terms of return on investment, we argue that a modest and fixed up-front cost in the provisioning of *data context* can significantly improve the results of automation, and thereby reduce the marginal cost involved in data scientists manually refining the results of automated processes or obtaining feedback.

Our contributions in this paper are as follows:

1) A definition of the notion of *data context*, and of its specific types.
2) A methodology to fully and cost-effectively bootstrap an end-to-end data wrangling process based on data profiling that enables further tuning of control parameters for components.
3) A description of how data context can inform multiple steps within an end-to-end wrangling process, specifically matching, mapping validation, value format transformation, rule-based data cleaning and mapping selection to generate and validate candidates with the objective of improving the accuracy, consistency, and relevance of the wrangling result.
4) An evaluation of the approach in two application domains (real-estate and open government data) that shows: (i) significant improvements in the results of automated processes (e.g., the f-measure of the result increased from *0.51* to *0.81* and from *0.59* to *0.8*); (ii) the impact of data context on the individual steps; and (iii) the scalability of the process in terms of number of sources (*25k*) and tuples (*150M*).

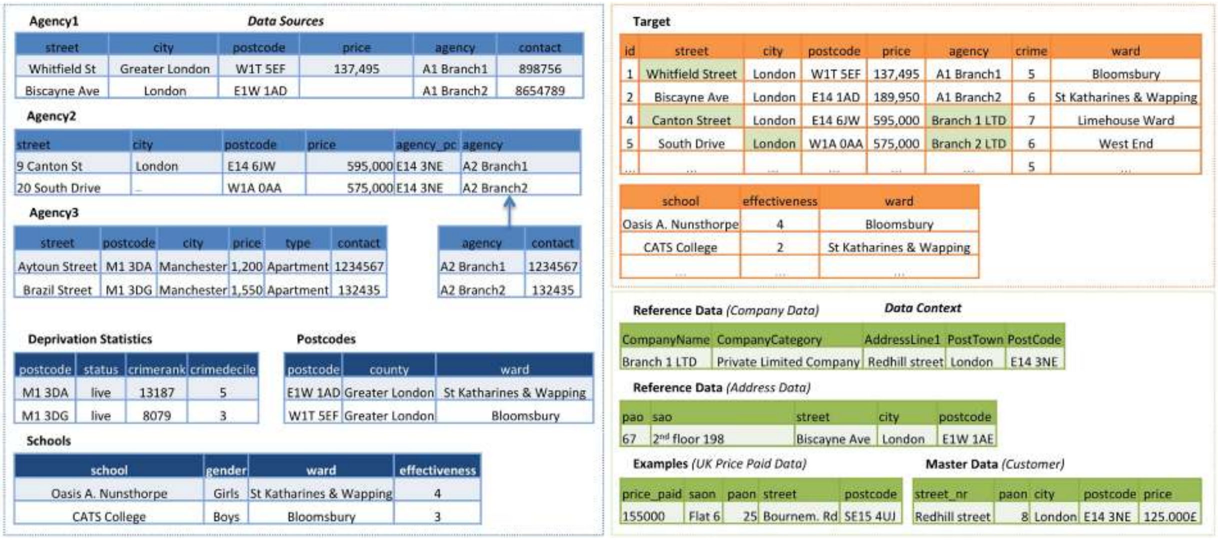
The paper is structured as follows: In Section 2 we define the problem, outline the end-to-end data wrangling process and define data context. Section 3 describes the methodology to automate an end-to-end data wrangling process, and details how the individual steps are automated in general and informed by data context. The process is thoroughly evaluated in Section 4. The paper concludes with related work, conclusions and future work in Sections 5 and 6.

2 Problem Statement

Although data wrangling processes may include different steps, in this paper we demonstrate its automation by applying data context using a specific domain-independent data wrangling process consisting of a series of representative steps. We assume that the end user is a data scientist, who is familiar with the domain within which the data is to be wrangled, and thus who can provide a *target* schema T consisting of multiple tables $t \in T$ without support of target constraints that is to be populated by the wrangling process. Given some data sources S each possibly described by a *source* schema, it is the role of the *wrangling process* to populate the target schema with as much as possible accurate, consistent and relevant values. Thus we have the following baseline *problem statement*:

> *Given a collection of data sources S and a target schema T, automatically populate the target schema with data from the source data sets that best "fits" the target.*

In the process in Fig. 2, this involves matching the source and target schemas, reformatting values that may be represented in different ways, completing or correcting inconsistent values, generating mappings from the matches, and selecting mappings and tuples. Carrying out all these tasks automatically is not straightforward and the data that ultimately populates the target schema is likely to be error-prone. For automating the process we initially assume that the sources S are populated with instances (e.g., extracted



Data Sources

Agency1

street	city	postcode	price	agency	contact
Whitfield St	Greater London	W1T 5EF	137,495	A1 Branch1	898756
Biscayne Ave	London	E1W 1AD		A1 Branch2	8654789

Agency2

street	city	postcode	price	agency_pc	agency
9 Canton St	London	E14 6JW	595,000	E14 3NE	A2 Branch1
20 South Drive	–	W1A 0AA	575,000	E14 3NE	A2 Branch2

Agency3

street	postcode	city	price	type	contact
Aytoun Street	M1 3DA	Manchester	1,200	Apartment	1234567
Brazil Street	M1 3DG	Manchester	1,550	Apartment	132435

agency	contact
A2 Branch1	1234567
A2 Branch2	132435

Deprivation Statistics

postcode	status	crimerank	crimedecile
M1 3DA	live	13187	5
M1 3DG	live	8079	3

Postcodes

postcode	county	ward
E1W 1AD	Greater London	St Katharines & Wapping
W1T 5EF	Greater London	Bloomsbury

Schools

school	gender	ward	effectiveness
Oasis A. Nunsthorpe	Girls	St Katharines & Wapping	4
CATS College	Boys	Bloomsbury	3

Target

id	street	city	postcode	price	agency	crime	ward
1	Whitfield Street	London	W1T 5EF	137,495	A1 Branch1	5	Bloomsbury
2	Biscayne Ave	London	E14 1AD	189,950	A1 Branch2	6	St Katharines & Wapping
4	Canton Street	London	E14 6JW	595,000	Branch 1 LTD	7	Limehouse Ward
5	South Drive	London	W1A 0AA	575,000	Branch 2 LTD	6	West End
...	...	...	...	...	...	5	...

school	effectiveness	ward
Oasis A. Nunsthorpe	4	Bloomsbury
CATS College	2	St Katharines & Wapping
...	...	...

Data Context

Reference Data *(Company Data)*

CompanyName	CompanyCategory	AddressLine1	PostTown	PostCode
Branch 1 LTD	Private Limited Company	Redhill street	London	E14 3NE

Reference Data *(Address Data)*

pao	sao	street	city	postcode
67	2nd floor 198	Biscayne Ave	London	E1W 1AE

Examples *(UK Price Paid Data)*

price_paid	saon	paon	street	postcode
155000	Flat 6	25	Bournem. Rd	SE15 4UJ

Master Data *(Customer)*

street_nr	paon	city	postcode	price
Redhill street	8	London	E14 3NE	125.000£

Fig. 1. Data wrangling scenario: A collection of raw Web extracted real-estate sources and deprivation data (left-hand side) is wrangled into the target (right-hand side). Data context, i.e., reference data, master data and examples, informs the process of schema matching, value format transformation, data repair, and mapping generation, validation and selection.

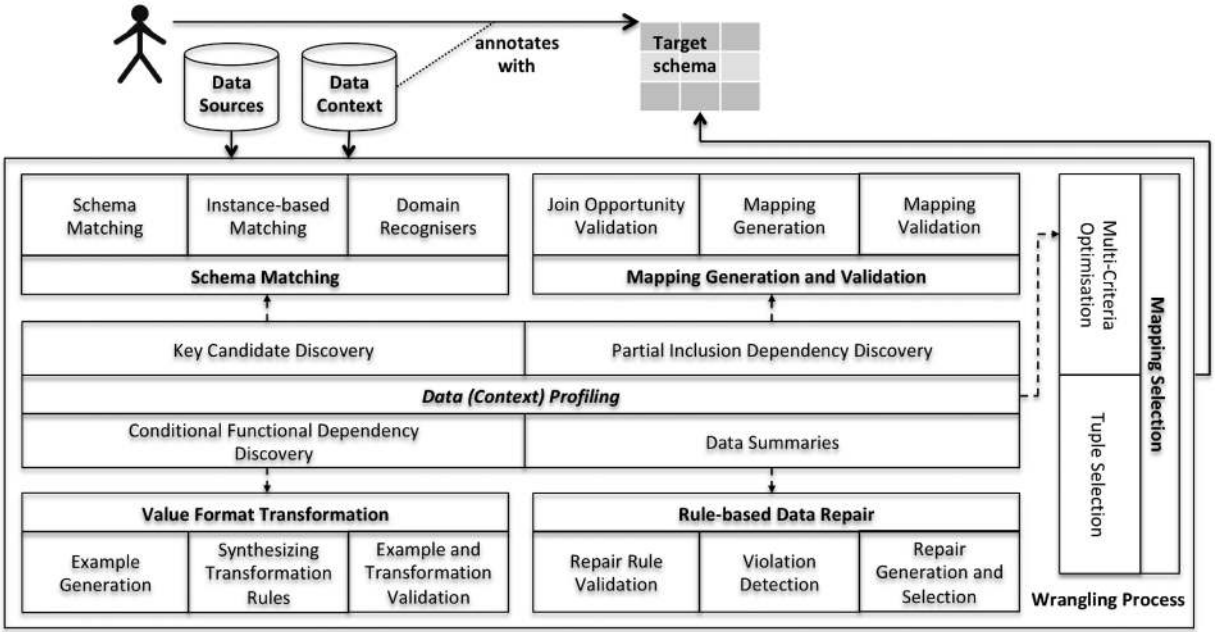


Fig. 2. Data wrangling process: User annotates the target schema with data context and starts the wrangling process. The process involves several stages including schema matching, format transformations, data repair, mapping generation and validation, and mapping selection. All stages can be automated by means of data profiling and informed by data context.

from the deep Web, from Web tables or mapping tables), but the target T is defined by its schema only.[1] Before defining data context in detail, we illustrate the process and the problem description by way of a *running example* detailing how a user might be using the wrangling process.

EXAMPLE 1. Consider the data wrangling scenario depicted in Fig. 1. The data sources are extracted from the deep Web and contain information about properties for sale that have been advertised by various real-estate agencies. Additional data sources holding information about deprivation statistics, postcodes and districts, and schools have been downloaded from an open government portal or are available in the data lake. Note that the sources are assumed to be automatically extracted from the Web and thus attribute headers might not be available or are most likely to be uncertain and often not meaningful.

Suppose the user wants to gain information about houses that are for sale, and specifies a target schema including attributes that describe streets, post codes, prices, agencies, schools and their effectiveness and crime statistics. We further assume that the user can associate this target schema with data of relevance – the data context. For example, in the UK, open data sets provide

[1] It is a complementary problem to infer a possible target schema for the collection of sources S, that could be tackled by applying a clustering technique such as in [26].

complete lists of street names, post codes and company registries (reference data), as well as details of historical property sales (examples). Furthermore, the user might have access to master data available in the company the user is working for. Such data collections either define the extent of certain attributes in the target, or provide indicative values. The user can relate several attributes of different data context categories with the target. For instance, the attribute "price_paid" in examples can be associated with the attribute price in the target, and the attributes "postcode, street, city" can be aligned between address reference data and the target. Note that not all attributes need to be associated, and that some values in the data context might be missing or erroneous.

A data wrangling process can infer schematic correspondences between the sources and the target. In addition, a wrangling process can synthesise consistent value formats for attributes in different sources and repair missing or erroneous values. For instance, street and agency names can be transformed ("Whitfield St" → "Whitfield Street", "Branch 1" → "Branch 1 LTD") by means of master or reference data, and missing "city" values can be repaired. Transformed or repaired values in the target are highlighted in Fig. 1. Furthermore, the process discovers schema mappings, integrating potentially transformed and repaired sources in different ways into the target. The wrangling process maps the data context to the target to better understand different quality dimensions, i.e., accuracy, consistency and relevance of the results of different wrangling stages, and to inform all the steps in the process. For instance, schematic correspondences and schema mappings can be validated, and a subset of the data that best fits the target can be selected.

2.1 Data Context Types

In this paper, we describe how automation can be informed by the data context, which consists of data sources D that can be aligned with the target schema, thereby providing partial, potentially erroneous and contradicting instance-based evidence about the target. Data context data can be:

Reference Data. A collection of values that stipulate the valid domain of a set of specific attributes of the target $\boldsymbol{T}$: correctly repaired and transformed instances I_t of $\boldsymbol{T}$ are a subset of instances I_d of $d \in D$, for the set of related attributes. Thus reference data is complete, in that there are no missing values, and accurate, in that it provides correct values that may be expected to occur in the product. Reference data can be utilised to estimate and improve the *accuracy* (the closeness of a value, v, to the elements of the corresponding definition domain [19]) of sources, intermediate results or the wrangling result.

Master Data. Master data can be defined as constituting a consistent view on the core entities in an organisation. Thus, master data are correct and accurate values stipulating a set of target attributes. In contrast to reference data, the set relation between the sets I_t of $\boldsymbol{T}$ and I_d of $d \in D$ is not known *a priori*. As a result, for example, on their similar attributes I_t and I_d may overlap, be disjoint, or one may be contained in the other. Master data supports the estimation and improvement of the *consistency* of sources, intermediate results and the wrangling result.

Example Data. A collection of data items that (partly) express the domain of the target. Examples may include empty and erroneous values and stipulate a set of target attributes. Again, the relation between the sets I_t and I_d is not known *a priori*. Examples support the estimation and improvement of the *relevance* of the sources, intermediate results, and the wrangling result, i.e., how close they are to the provided input.

The different data context types are illustrated in Fig. 1 for the running example.

2.2 Data Context Relationships

Data context sources (D) can be related to the tables in the target schema $\boldsymbol{T}$ using *data context relationships* $R(d, t, c)$, where $d \in D$, $t \in \boldsymbol{T}$ and $c \in \{reference, master, example\}$. For expressing them we use the notion of a *tuple generating dependency* (*tgd*) of the form $\forall x(\phi_D(x) \rightarrow \exists y \psi_T(x, y))$ where $\phi_D(x)$ is a conjunction of atoms over the data context and $\psi_T(x, y)$ is a conjunction of atoms over the target schema.

The target is not directly populated from the data context, but rather the data context is used to inform the steps that populate the target. We assume the data scientist has sufficient knowledge of the domain to identify suitable data sets for the data context, and to envisage their precise relationship to the target schema (as exemplified by the *tgd* above). Data context relationships might then be made available as demonstrated via the VADA user interface [27].

2.3 Refined Problem Statement

We can now explicitly define the *problem statement* of how to inform an end-to-end data wrangling process consisting of multiple steps with data context and data profiling:

> *Given a set of sources $\boldsymbol{S}$, source instances I_s for each source $s \in \boldsymbol{S}$, a target $\boldsymbol{T}$, data context sources D with instances I_d, and data context relationships $R(d, t, c)$, automatically populate the target $\boldsymbol{T}$ with (potentially transformed and repaired) instances that best "fit" the target, i.e., that optimise the accuracy, consistency and relevance of the wrangling result.*

3 Data Context Informed Wrangling

This section describes how data context is used to inform the automation of the stages in the wrangling process to improve the accuracy, consistency and relevance of the wrangling result. Fig. 2 provides an overview of the wrangling process populating a target schema potentially consisting of multiple tables, and acting on top of a set of semi-structured data sources stored in different formats (e.g., CSV, XML, relational). The target schema is to be annotated with data context and the wrangling process is invoked by the user. The wrangling process includes five consecutive stages, as described in Algorithm 1, namely schema matching (line 2), value format transformation (line 3), rule-based data repair (line 4), mapping generation and validation (line 5), and mapping selection (line 6) to be informed by data context and data profiling that is executed upfront.

Selecting or discovering sources based on their relevance, consistency, and accuracy could potentially be done early in a wrangling pipeline, which would lower the computational costs because there would not be the need to execute repair, transformation, matching and mapping generation for all sources. In our approach, we show that the wrangling results benefit from selecting sources late in the process, as the results of repairs and transformations, as well as the integrability of sources can be incorporated in the decision process.

Each wrangling step involves two types of functionality: (i) a *concrete* technique that takes the form of one or more calls to the operations of an existing state-of-the-art approach to that wrangling stage; and (ii) an *abstract* algorithm, building on evidence that includes data profiling and data context, to initialise the inputs required by the concrete technique, tune configuration parameters or thresholds, and validate generated programs. In all cases, we have adopted state-of-the-art methods for the concrete techniques, and we highlight calls of *existing techniques* in *italic* in the abstract algorithms. The presented algorithms are *abstract* in the sense that they could make use of alternative concrete techniques. In general, we utilise data context in a twofold manner to (1) generate program candidates (e.g., matches, repair rules or data examples) and (2) to validate and select candidates (e.g., mappings, specific values and their format).

In what follows, the emphasis in this paper is on the abstract algorithms that enable each wrangling step to make use of data context, as it is these that enable the automation of the complete wrangling process.

Algorithm 1. Data Context Informed Wrangling

```
Require: sources $S$ and instances $I_S$, target schema $T$, Set of data context sources $D$ and instances $I_D$, Source and data context profiling results $P_S, P_D$, configuration $C$
Ensure: set of selected mappings and number of selected tuples per mapping $G_{best}$
1: procedure WRANGLING
2:     $M \leftarrow$ matching$(S, I_S, T, D, I_D, C)$    $\triangleright$ $M$ includes matches from the sources $S$ to the target $T$
3:     $S', I'_S \leftarrow$ transform$(S, I_S, D, I_D, M, P_D, P_S, C)$ $\triangleright$ $S'$ includes sources $S$ with transformed value formats according to $D$
4:     $S'', I''_S \leftarrow$ repair$(S', I'_S, D, I_D, M, P_D, C)$ $\triangleright$ $S''$ includes sources $S$ with repaired values according to $D$
5:     $G, I_G \leftarrow$ mapping$(S'', I''_S, T, D, I_D, M, P_D, P_S, C)$    $\triangleright$ $G$ includes the set of many-to-one mappings between sources $S$ and the target $T$
6:     $G_{best} \leftarrow$ mappingselect$(G, I'_G, T, D, I_D, C)$    $\triangleright$ $G_{best}$ includes the set of selected mappings and number of selected tuples per mapping
7:     return $G_{best}$
8: end procedure
```

3.1 Source and Data Context Profiling

Data profiling can be defined as the problem of efficiently analysing a given data set[28] and is utilised in various domains from data cleansing, through data integration to data analytics. Data profiling algorithms give rise to a means of informing data wrangling as a whole and of individual wrangling stages. Tools such as Metanome[2] support a wide variety of profiling tasks including detection of unique column combinations, partial inclusion dependencies or functional dependencies. Herein, we convey the idea that a data wrangling process can be cost-effectively bootstrapped by means of the results of such data profiling methods executed on top of the input sources as well as on the data context (input to Algorithm 1). We detail the data profiling information used to automate the data wrangling process while describing the individual wrangling steps.

3.2 Schema Matching

Schema matching can be defined as the problem of detecting schematic correspondences between schema elements of data sources S and the target T. Schematic correspondences identify potentially equivalent pairs of schema elements, along with a confidence measure that is most often expressed as a similarity score. Especially in the context of raw data sources extracted from the deep web or via (open government) data portals, sources usually do not comply with a single schema, and detecting schematic correspondences remains an open research challenge. Correctly identifying schematic correspondences is usually not possible in a completely automated way.

3.2.1 Automating Schema Matching

We approach the challenge of automating schema matching by applying the generate-and-test methodology. Generating and testing candidate schematic correspondences involves different types of evidence.

Metadata evidence captures characteristics of schema elements such as their names, data types, and structural properties, and supports comparison of the source and the target schema for finding correspondences.

Target instances provide additional evidence on the values that are part of the target, which can be exploited by instance-based matchers. String similarities and word frequencies can be used to improve correctness of the generated candidates, for instance based on indexing or signature based approaches [29].

Domain-specific evidence explains additive knowledge of parts of a data source. Usually, domain evidence is created and maintained by domain experts and can be exploited by domain recognisers or gazetteers. Generic domain recognisers can be used in a system to raise or lower similarity scores of matches based on additional evidence[13].

3.2.2 Context Informed Automation

Algorithm 2 is used to automate schema matching, using data context information when it is available. The algorithm is invoked for each source S and the target T. It uses metadata and data context based evidence in the two phases, generate, and test.

In the absence of data context, the abstract algorithm applies schema-based matchers (line 2) to generate candidate schematic correspondences, if source schemas are available. Our concrete implementation combines different metadata-based match heuristics, i.e., column names, column name tokens, data types, statistics such as set averages, schema paths, and parent and leaf relationships in the schema, to calculate schematic correspondences between the source and the target. Specifically, we execute the Coma community edition,[3] with the workflow configuration 7001.

[2] Metanome: http://metanome.de

[3] Coma CE: https://sourceforge.net/projects/coma-ce/

When data context is provided in D, each such data set is used as a partial extensional representation of the target to carry out instance-based matching with the source (line 6). Any instance-based matchers such as signature or index-based approaches can be utilised. In our approach, the term frequency inverse document frequency (tf-idf) is calculated in addition to the schema-based matchers (Coma++ instance matchers are used).

Algorithm 2. Matching

```
Require: source schema $S$ and instances $I_S$, target schema $T$,
    Set of data context schemas $D$ and instances $I_D$,
    configuration $C$ (including lower and upper bound $lb, up$)
Ensure: Set of matches $M$
 1: procedure MATCH
 2:     $M \leftarrow gen\_schema\_match(S, T)$
 3:     $M_d \leftarrow \{\}$
 4:     for all $d \in D$ do
 5:         $M_d \leftarrow$ combine($M\_d, gen\_inst\_match(S, d, C)$)
 6:     end for
 7:     $M \leftarrow$ combine_match($M, M_d$)
 8:     for all $d \in D$ do
 9:         $M \leftarrow$ test_match($S, T, d, C$)
10:     end for
11:     return $M$
12: end procedure
```

Match testing takes further advantage of data context through utilisation of domain-specific validation (line 10). In our system, we have implemented generic recognisers, exploiting information gained from data context. The generic recognisers combine inference of basic types, characteristics such as the distribution and length of the values, as well as string similarity measures. For type inference we utilise a set of predefined patterns to match the extracted data. Examples include numbers, dates, URLs, phone numbers and postcodes, similarly to [30]. We exploit regular expressions defined in the regular expression library.[4] An example regular expression used for UK postcodes, examples of which are given in Fig. 1, is shown here:

```
^([A-PR-UWYZ0-9][A-HK-Y0-9]
[AEHMNPRTVXY0-9]?[ABEHMNPRVWXY0-9]?
{1,2}[0-9][ABD-HJLN-UW-Z]{2}|GIR 0AA)$
```

Recognisers are used to refine schematic correspondences to target attributes aligned with data context by increasing or decreasing their similarity scores, and to detect new correspondences undetected by schema- and instance-based matchers.

In general, we execute schema and instance-based matchers with a low threshold (0.2) to avoid missing correct matches and we normalise match scores to hide differences in the computed ranges (scores depend not only on source and target instances, but also on the utilised similarity measures). In addition, we restrict the search space by choosing max n matches per target attribute. After testing correspondences with domain recognisers, we generate all potentially correct match sets between the sources and the target and compute a normalised score for each match set supporting the ranking of match sets. For execution of the next steps (transformations, repair and mapping generation and validation), we restrict the search space to the top 10 match sets.

3.3 Value Format Transformation

Value format transformation [24], [31] is defined as the problem of changing the textual representation of values for an attribute from sources into a uniform format represented in the target. This task is carried out by applying a range of syntactic string manipulations on the source values, such as string concatenation, sub-string extraction or sub-string permutation. For example, a source might abbreviate recurring parts of an address (e.g., *Canton St*), when the full representation is required (*Canton Street)*. Correctly inferring and applying transformation rules is a hard problem and usually involves some form of user involvement [10], [31]. In our approach, we seek to automatically identify data examples that can be used to synthesise transformation rules using FlashFill [31].

3.3.1 Automating Value Format Transformations

We approach the challenge of automating value format transformations by applying the generate and test methodology based on different types of evidence.

Metadata evidence describing schema elements combined with schematic correspondences between the source and the target are used to identify potentially equivalent attributes whose value representations have to be aligned.

Profiling evidence, such as functional dependencies, gives rise to a means of generating data examples relating attributes from a data set with a target [24]. In this paper, data context provides the extensional data that is required for the target. More specifically, assume that we wish to transform the values in source attribute $\boldsymbol{S}.s_n$ into the format used in $\boldsymbol{T}.t_m$. If functional dependencies (FD) in $\boldsymbol{S}$ and in $\boldsymbol{T}$, i.e., $fd_1 : [\boldsymbol{S}.s_i] \rightarrow [\boldsymbol{S}.s_n]$ and $fd_2 : [\boldsymbol{T}.t_j] \rightarrow [\boldsymbol{T}.t_m]$, and schematic correspondences exist between the determinants $(\boldsymbol{S}.s_i, \boldsymbol{T}.t_j)$ and the dependents $(\boldsymbol{S}.s_n, \boldsymbol{T}.t_m)$ of the functional dependencies, and if the values in the determinants correspond, data examples can be generated automatically. However, the challenge of selecting the correct representation for specific attributes, i.e., the direction of the transformation, remains open and thus fully automating transformations needs to stem from additional types of evidence such as data context.

3.3.2 Context Informed Automation

Algorithm 3 is used to automate value format transformations when data context is available. To automate value format transformation, we make use of data context as a partial representation of the target. The algorithm is invoked for each source S separately and returns a transformed source $\boldsymbol{S'}$. First, source and target data examples between the source and each data context source $d \in D$ are generated (line 4). Our concrete implementation of generating data examples is based on functional dependencies within the source and all data context types. Transformation rules are generated and validated based on the accumulated data examples (line 5). The concrete algorithm used to synthesise transformation rules from the identified source and target data examples applies a programming-by-example (PBE) approach. A full description of the operations used in

[4] Regular expression library: http://regexlib.com/

transformation rules, e.g., string concatenation, sub-string extraction or sub-string permutation, together with a detailed description of the domain-specific language to perform them, is available in [31].

Algorithm 3. Value Format Transformation

```
Require: source $S$ and instances $I_S$, Set of data context schemas
    $D$ and instances $I_D$, set of matches $M$, set of source and
    data context profiling $P_S, P_D$, configuration $C$
Ensure: Transformed sources $S'$ and instances $I'_S$
 1: procedure TRANSFORM
 2:     $R \leftarrow \{\}$          ▷ $R$ are selected transformation rules
 3:     for all $d \in D$ do
 4:         $E_d \leftarrow generate\_examples(s, d, M, P_S, P_d)$
 5:         $R_d \leftarrow generate\_test\_transform\_rules(E)$
 6:         $R \leftarrow$ accumulate_validated$(R_d)$
 7:     end for
 8:     $T' \leftarrow compute\_transformations(S, R)$
 9:     $S' \leftarrow$ fuse_transformations$(S, T')$
10:     return $S', I'_S$
11: end procedure
```

We apply k-fold validation to select a set of transformation rules. In this process, the set of examples is randomly partitioned into k equally sized subsets. Then, transformations are synthesised, in k rounds, using the examples from the other $k - 1$ partitions, and the synthesised transformation is tested on the remaining partition.

Algorithm 3 uses different data context types consecutively to generate and test data examples and transformation rules. If source columns can be transformed by means of multiple data context items, we can improve the accuracy, consistency and relevance of the sources with respect to the different data context sources.

Synthesizing transformation rules for a single source attribute based on different possibly contradicting and disjoint sets of data examples, as well as applying different sets of data examples/transformation rules consecutively could lead to inconsistent transformation results, which results in the problem of truth discovery[32]. Thus, we propose two additional steps in the abstract algorithm generating the candidate transformed values (line 9) and fusing the candidate values (line 10). There exist multiple approaches, e.g., majority voting, incorporating trustworthiness of sources, or Bayesian methods, to address the data fusion problem, often applied on a large set of Web sources. In [33] the authors present a scalable and robust truth discovery scheme for many sources. Herein, we have to decide the true value for a large set of tuples, but for each, from a small set of values generated by rules that have been learned from data context. In our approach, we apply majority voting to fuse potentially contradicting values that have been generated by means of different data context types (reference data, master data, examples).

3.4 Rule-Based Data Repair

The data repair problem involves detecting and repairing certain classes of data errors, e.g., violations of integrity constraints. Integrity constraints can be given in a range of languages, varying from user-defined functions to database constraints like conditional functional dependencies or inclusion dependencies [34]. Here we adopt conditional functional dependencies (CFD). A CFD

$$\phi = (R : X \rightarrow Y, t_p),$$

where X and Y are disjoint subsets of the set of attributes, extends standard functional dependencies (FDs) by enforcing pattern tuples t_p of semantically related constants [25]. To increase the consistency and accuracy of data, violations have to be detected based on the given constraints, and suitable repairs have to be chosen for the detected violations.

3.4.1 Automating Rule Based Data Repair

We approach the challenge of automating rule-based data repair by applying generate and test phases. Generating and testing involves information gained from target instances and concepts known from applying master data.

Target instances can be used to underpin automatic discovery of integrity constraints from data. For instance, the CFD discovery algorithm [22] is capable of finding a canonical cover of s-frequent minimal constant CFDs based on an input relation R and a support size $support$. The support size is the number of tuples matching the pattern of each CFD learned by the algorithm. *Master data* can be used to generate certain fixes based on editing rules [35]. Certain fixes can be characterised as fixes that are known to be absolutely correct, i.e., do not introduce new errors when repairing the data. Intuitively, this method aligns with our approach.

Algorithm 4. Rule-Based Repair

```
Require: sources $S$ and instances $I_S$ target schema $T$, set of
    relationships $L$, Set of data context schemas $D$ and
    instances $I_D$, data context profiling $P_D$, configuration $C$
Ensure: Repaired sources $S'$ and instances $I'_S$
 1: procedure REPAIR
 2:     $R \leftarrow \{\}$                        ▷ Repair rules
 3:     for all $d \in D$ do
 4:         $R_d \leftarrow$ test_rules$(P_d, d, I_d, C)$
 5:         $R \leftarrow$ accumulate_rules$(R, R_d)$
 6:     end for
 7:     $R \leftarrow$ rewrite_rules$(R, S, L)$
 8:     $V \leftarrow generate\_violations(S, R)$
 9:     $E' \leftarrow generate\_repair(S, I_S, V)$
10:     $E'' \leftarrow test\_repair(S, I_S, E')$
11:     $S' \leftarrow apply\_repair(S, I_S, E'')$
12:     return $S', I'_S$
13: end procedure
```

3.4.2 Context Informed Automation

Algorithm 4 is used to automate rule-based repair when data context is available. The algorithm is invoked for each source S and the wrangling target T. It takes source instances I_S and data context into account. First, the algorithm proposes a method for automatically testing and validating available repair rules, i.e., learned from data context (line 4). This validation step is executed for each data context type $d \in D$. Afterwards, the generated repair rule candidates are accumulated and prepared for the specific source S (line 5 and line 7). After violations of the validated repair rules have been detected in the source (line 8), a concrete repair algorithm is executed on the source and cleans the source according to the automatically generated and validated rules (line 9 - 11).

In our concrete implementation, we use domain-independent CFDs as repair rules and generalise an approach that utilises master data to discover certain fixes [35] towards selecting CFDs for all available data context sources D. This relaxes the notion of certain fixes, as data context might provide spurious evidence. To fully automate the process, there is a need to automatically validate the learned repair rules (line 8). A strict validation method (see Algorithm 5) is necessary as discovered CFDs might be erroneous (e.g., the $cfd_1 : ([town] \rightarrow [street], (London || KingsCross)$ could be read in the way that all tuples with the town *London* should have the street *Kings Cross* or they are erroneous) or the data context might provide spurious evidence (e.g., examples might not include correct values).

The described CFD validation algorithm (Algorithm 5) focuses on selecting a set of CFDs, CFD_{best}, maximising the precision of the repair process, i.e., minimising the number of incorrect repairs. We incrementally increase the support size parameter used for discovering CFDs for each data context type CFD_d (line 4, 5, 11) and apply a validation step (line 7) on the selected ones. The confidence of the CFDs is used to calculate a $score$ for each iteration (line 7) to select the set of CFDs CFD_{best} to be applied in the repair process. The score is the percentage of CFDs with confidence equal to 1, i.e., CFDs not violating any tuple in the training data. Finally, all CFDs violating tuples in the training data are filtered out.

Algorithm 5. Cfd Validation

```
Require: CFDs $CFD_d$ learned from data context source $d$,
    data context schema $d$ and instances $I_d$, configuration $C$
    including (lower bound $lb$, initial support $is$, step size $step$)
Ensure: Validated CFDs $CFD_{best}$
 1: procedure CFD VALIDATION
 2:     $CFD_{best} \leftarrow \{\}$
 3:     $score_{best} = lb$
 4:     $support \leftarrow is$
 5:     while $score > score_{best}$ do
 6:         $CFD_{d'} \leftarrow$ select_cfd$(CFD_d, d, support)$
 7:         $score \leftarrow$ validate$(CFD_{d'}, d, I_d)$
 8:         if $score > score_{best}$ then
 9:             $score_{best} = score$
10:             $CFD_{best} \leftarrow CFD_{d'}$
11:             $support = support + step$
12:         end if
13:     end while
14:     return $CFD_{best}$
15: end procedure
```

To apply rule-based repair, we again use data context to represent target instances. Thus, we assume that we can apply the aggregated CFDs, discovered and validated using different data context types for the target, to detect violations and generate repair operations for sources based on the repair algorithm described in [25] (line 9,10,11). Repair operations are based on attribute value modifications as they are sufficient to resolve CFD violations. In short, and following the notation in [25], if a tuple t violates a CFD $\phi = (R : X \rightarrow Y, t_p)$, composed of a FD plus a pattern tuple t_p, the algorithm either modifies the values of t for the attributes matching the right-hand side of the FD, according to the pattern tuple, or modifies the values of some attributes of t matching the left-hand side of the FD. In case of violations of t with another tuple t', different attribute modifications will be applied. The repair algorithm produces a repair that is as close as possible to the original dataset, by choosing, at each step (testing), to repair the attribute of a tuple t with minimum repair cost. Such a cost model depends on a distance function, which in our case is based on the Damerau-Levenshtein metric.

3.5 Schema Mapping Generation and Validation

Schema mapping generation and validation can be defined as the problem of generating data transformations from data sources $\boldsymbol{S}$ into a target $\boldsymbol{T}$ and validating the resulting candidates for use. Schema mappings can be expressed using source-to-target tuple generating dependencies (st-tgds) of the form

$$\sigma : \forall x(\phi_S(x) \rightarrow \exists y \psi_P(x, y)),$$

where $\phi_S(x)$ is a conjunction of source atoms, and $\psi_P(x, y)$ is a conjunction of target atoms.

3.5.1 Automating Mapping Generation

We approach the challenge of automating schema mapping generation and validation by applying generate and test phases based on different types of evidence.

Metadata evidence describing schema elements, their structure and primary/foreign key relationships, combined with schematic correspondences between the sources and the target, supports the application of mapping generation approaches such as ++Spicy [36], S4 [37] or [38].

Profiling Data profiling [20], [21] infers descriptive information about sources that can be exploited by mapping generation and selection, though automatically detected candidate keys and partial inclusion dependencies can provide misleading evidence.

Target instances can be exploited by mapping validation approaches using instance-based similarity measures between mapping results and target instances. For example, tree similarity measures taking into account the topology and the information content support target instance evidence [36]. In [37], row and column-based containment scores are used to validate project-join queries, and in [39] feedback on tuples is used to inform selection.

3.5.2 Context Informed Automation

Algorithm 6 is used to automate mapping generation and validation without data context, and improves the result if data context is available. The algorithm is invoked for the set of input sources $\boldsymbol{S}$ and the target $\boldsymbol{T}$ and takes metadata, profiling and data context evidence into account. In the generation phase (line 2), we validate join opportunities between the sources $\boldsymbol{S}$ based on key candidates and partial inclusion dependencies. In general, our approach supports the use of the full set of detected partial inclusion dependencies and allows relaxation of the assumptions on key attributes.

To enable the search for candidate mappings and to focus on promising groups of source tables, we divide the search space into smaller problems by applying a table union search and cluster each unionable input source (e.g., real-estate source) with all additional sources (e.g., schools or deprivation statistics) to be integrated (line 3), and by executing mapping generation and validation for each source cluster. In addition, we restrict the search space by applying a threshold of 0.5 on the overlap of detected inclusion dependencies. To detect key candidates, a threshold of 0.9 is applied to accommodate potential outliers, i.e., non-distinct, null or empty values.

To generate candidate mappings, we search the space of candidate mappings involving join operations based on validated join opportunities for each source cluster (line 8) by aligning possible source join paths with the target. We generate mappings for the top-k match sets as described in Section 3.2 (*Schema Matching*). Executing the mapping generation algorithm results in a set of candidate mappings to be validated by data context. Generated mappings are expressed in VADALOG[40], a member of the Datalog± family of logic languages whose core language contains rules known as *existential rules* or *source-to-target dependencies*. A simplified candidate mapping (see running example depicted in Fig. 1) populating both target tables by joining *Agency 1 (A)*, *Deprivation statistics (D)* and *Postcodes (P)* as well as selecting schools from *Schools (S)* is shown in (1).

$$\begin{aligned}\sigma_1 : \forall x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}(A(x_1, x_2, \\ x_3, x_4, x_5, x_6) \wedge D(x_3, x_7, x_8, x_9) \wedge P(x_3, x_{10}, x_{11})) \rightarrow \\ \exists y_1 T_1(y_1, x_1, x_2, x_3, x_4, x_5, x_8, x_{11})) \qquad (1) \\ \sigma_2 : \forall x_{12}, x_{13}, x_{14}, x_{15}(S(x_{12}, x_{13}, x_{14}, x_{15})) \rightarrow \\ T_2(x_{12}, x_{15}, x_{14})).\end{aligned}$$

When data context is available, we test mapping candidates G for each cluster by computing mapping validation scores (line 13) for each available data context type. We calculate the Jaccard set containment score, defined in Equation (2), as a measure for similarity between the instances I_{g_k} of attribute k in the set of attributes $dom(g)$ of the mapping g and the instances I_{d_j} of each aligned attribute j in the set of attributes $dom(D)$ in the data context type D.

$$score_{g_{d_k}} = \max_{j \in dom(d)} \frac{|I_{d_j} \cap I_{g_k}|}{|I_{d_j}|}. \qquad (2)$$

As the different attributes in the sources and the data context might not be statistically independent and to detect correct join paths, we calculate a row-row similarity metric (see Equation (3)). We adapt the metrics used in S4[37] according to data context to estimate the accuracy (reference data), consistency (master data) and relevance (examples) for each mapping candidate. The combined metric calculates the mean of the validation scores for all aligned mapping attributes.

$$score_{g_d} = \frac{\sum_{k \in dom(g)} score_{g_{d_k}}}{|dom(g)|}. \qquad (3)$$

Finally, we calculate a combined score for each candidate mapping as defined in Equation (4) to rank the candidates according to their similarity with the data context.

$$score_g = \frac{\sum_{d \in D} score_{g_d}}{|d|}. \qquad (4)$$

The algorithm repeats the generate and test phases for each cluster (line 5), retaining the candidate mappings with the best scores (line 21). In the final experiments we estimated the Jaccard containment scores with Locality Sensitive Hashing Ensemble[41]. This enables us to scale the approach without significantly reducing result quality. In general, the algorithm is capable of ranking mappings according to their accumulated scores. This enables a top-k approach and the incorporation of user feedback to refine mappings which is left open for future work. In the absence of data context, we choose the mapping candidate satisfying the most schematic correspondences for each group of sources.

Algorithm 6. Mapping Generation and Validation

```
Require: set of source schemas $S$ and instances $I_S$, target
    schema $T$, Set of data context schemas $D$ and instances $I_D$,
    set of relationships $R$, set of matches $M$, source profiling
    $P_S$, configuration $F$
Ensure: Set of mappings $G$
 1: procedure MAPPING
 2:   $P_{SV} \leftarrow$ validate$(S, P_S)$
 3:   $C \leftarrow$ cluster$(S, M, P_{SV})$
 4:   $G_{best} \leftarrow \{\}$
 5:   for all $c \in C$ do
 6:     $score_c \leftarrow 0$
 7:     $g_c \leftarrow \{\}$
 8:     $G \leftarrow gen\_schema\_mapping(c, M, P_{SV})$
 9:     for all $g \in G$ do
10:       for all $d \in D$ do
11:         $score_{g_d} \leftarrow$ calculate$(g, D, R, I_D)$
12:       end for
13:       $score_g \leftarrow$ accumulate$(score_{g_d})$
14:       if $score_g > score_c)$ then
15:         $score_c \leftarrow score_g)$
16:         $g_c \leftarrow g$
17:       end if
18:     end for
19:     $G_{best} \leftarrow g$
20:   end for
21:   return $G_{best}, I_G$
22: end procedure
```

3.6 Multi-Criteria Mapping Selection

Source selection can be defined as the problem of identifying a subset of sources (along with the quantities of data to select from each) from a set of available data sources S, that are most valuable for the wrangling scenario and thus most fit for purpose. Here, we do not apply source selection, but mapping selection after matching, transformations, repairs and structure transformations. This increases the integration costs but supports incorporation of the intermediate results and the integrability of the sources in the selection process. Different approaches to source selection map the problem onto a single criterion for optimisation to rank sources or tackle the problem as multi-criterial and calculate a trade-off solution most aligned with a user's preferences regarding a set of criteria [23]. The objective of our approach to mapping selection is to select a collection of n tuples from across the set of mappings with the number of tuples selected per mapping potentially less than the size of the mapping. The targeted size, n, is a user-provided input to the selection algorithm.

3.6.1 Automating Mapping Selection

We approach the challenge of automating mapping selection by applying generation and test phases. Generating and testing are based on different types of evidence.

Source instances can be exploited by mapping selection approaches to calculate mapping-specific criteria and mapping overlap estimations. In general, mapping statistics such as the number of nulls, the standard deviation of attributes or the number of tuples in the mapping can be used as input for multi-criteria based optimisation techniques[42],[23].

Target instances can be exploited by source or mapping selection approaches to calculate quality criteria[43] to improve the selection process. In [42], feedback instances are used to estimate different quality dimensions such as accuracy and freshness of sources. In general, these estimates can be used as input to source or mapping selection approaches. Further, different similarity measures[41] such as the Jaccard set containment score and string similarity measures can be used to calculate appropriate criteria showing good performance over large data sets.

Algorithm 7. Multi-Criteria Mapping Selection

Require: set of mappings G, set of mapping extents I_G, target schema T, Set of data context schemas D and instances I_D, set of relationships R, configuration F including (set of criteria weights W, targeted size ts)

Ensure: Set of selected mappings and number of selected tuples per mapping G_{best}

```
1: procedure MAPPINGSELECT
2:     $C \leftarrow \{\}$                              ▷ $C$ is the set of all criteria
3:     $G_{best} \leftarrow \{\}$
4:     for all $g \in G$ do
5:         $C_g \leftarrow$ calc_criteria$(g, I_g)$      ▷ $C_g$ are criteria calculated without data context
6:         $C \leftarrow$ add$(C, C_g)$
7:         for all $d \in D$ do
8:             $C_{g_{dc}} \leftarrow$ calc_dc_criteria$(g, I_g, d, R, I_d)$      ▷ $C_{g_{dc}}$ are criteria calculated with data context
9:             $C \leftarrow$ add$(C, C_{g_{dc}})$
10:        end for
11:    end for
12:    $W \leftarrow$calculate_weights$(W)$
13:    $G_{best} \leftarrow optimise(G, C, W, ts)$
14:    return $G_{best}$
15: end procedure
```

3.6.2 Context Informed Automation

Algorithm 7 is used to automate mapping selection, exploiting data context information when it is available. The algorithm is invoked for the set of validated mappings G generated in mapping generation and validation for each source S. It takes the mapping extents I_G and data context-based evidence D into account. The algorithm estimates the quality for each mapping $g \in G$ by generating a set of criteria C_g for when data context is not available (line 5) and a set of criteria $C_{g_{dc}}$ if data context is available (line 8). Criteria that are dependent on the mapping extent I_G to be used in the approach are the *completeness* of sources, i.e., the number of nulls and empty values, and the *size*, i.e., the number of tuples in the sources.

When data context is available, the algorithm generates criteria C for each available data context type estimating the *accuracy* w.r.t. reference data, the *consistency* w.r.t. the given master data, and the *relevance* of the mappings M w.r.t. to the provided examples. In general, different types of similarity measures between the mappings and the data context could be exploited (e.g., [44]). In our approach, we utilise Locality Sensitive Hashing (LSH) Ensemble[41] to estimate the Jaccard set containment score (see mapping generation and validation), defined in Equation (2), as a measure for similarity between attribute k in the source S and each aligned attribute j in the data context D in the same way as for mapping validation (see Section 3.5). Again we argue that the different attributes in the sources and the data context might not be statistically independent, and therefore we calculate a combined metric for each data context type to calculate the accuracy (reference data), consistency (master data) and relevance (examples) of the mappings.

We apply a concrete multi-criteria optimisation approach (line 13) for the problem of selecting a subset of valuable mappings for the data wrangling process [23] based on data context. The approach takes inspiration from Multi-Criteria Decision Analysis (MCDA) and determines a trade-off solution that has high overall weighted utility. The optimisation model is implemented using Multi-Objective Linear Programming (MOLP) in LPSolve.[5]

The algorithm takes as input (configuration F) a number of tuples to be retrieved ts and a set of criteria weights W. The algorithm calculates normalised criteria, the selection utility SU, for the quantity of tuples selected from a mapping for each criterion (see [23]). Afterwards, the method optimises the overall weighted utility (see Equation (5)) based on the selection utility (SU) of the possible solution space and the provided weights W. The selection utility is defined for each criterion C and is a normalised value with respect to the range of possible values a criterion can take, therefore allowing aggregation of different criteria values. In addition, the approach is able to take into account the range of values between the criteria ideal and negative ideal solution for each criterion.

$$WU = \frac{\sum_{i=1}^{C} W_i(\sum_{j=1}^{S}(SU_{ji}/ts))}{|C|}. \tag{5}$$

The result of executing mapping selection is an integer vector of length $|M|$ representing the quantity values of tuples to choose from each mapping (where 0 denotes a mapping that is not selected).

4 Experimental Evaluation

We investigate the effectiveness of the data wrangling process by measuring the gain in data quality in the target by

[5] LPSolve: http://lpsolve.sourceforge.net/

informing multiple data wrangling steps with different types of data context in two real world scenarios. As there is not, to the best of our knowledge, a direct competitor tackling end-to-end data wrangling and considering different types of context information, we evaluate the effect against the base case where data context is not used. The efficiency of the approach is evaluated by analysing the computational complexity and by measuring the scalability and performance of the approach.

4.1 Experiment Setup

4.1.1 Application Domain and Data

We perform our experiments on two real world datasets consisting of web-extracted data from the real-estate domain and of the UK open government data portal. The data of the primary scenario (Real-estate - RE) is an extension of the running scenario described in Section 2, and consists of data from 1k real-estate agencies (54k tuples). All datasets have been extracted with Diadem [13] according to their representation on the web page to be as close as possible to a completely automated extraction.

In addition to the real-estate data, we include freely available open government data sets providing statistical information about the locations,[6] and about schools[7] situated nearby properties for sale and their effectiveness.[8] The objective of the scenario is to wrangle the diverse set of sources representing real-estate properties with additional information and to select the best fitting sources and tuples for the specific target.

The second wrangling scenario (Financial data - FD) is based on freely available data from from the UK open government data portal that is categorised into twelve areas. To assume a realistic scenario we integrate all data on *vendor* specific *expenditures*, related *departments* and their *addresses* (91k tuples) and define the target schema *Address, Directorate, Department, Vendor, Expenditure, Payment Date*. These sources are part of the *governmental spending* category with 12.5k sources in total.

4.1.2 Data Context

To evaluate the effect of data context we searched for available and representative data sets to be used as examples, reference and master data. For reference data in the real-estate domain we utilised a subset of the open address data set for the UK for the area we are interested in (30k tuples) that provides high quality address data. For examples we used the freely available UK price paid data including 160k tuples representing property sales information. To emulate master data of a single agency (customer), we cleaned the set of extracted records from a single real-estate agency.

For the second scenario we scraped the set of organisations and corresponding addresses from the open government data portal, checked it by hand and used it as reference data. As master data we used the cleaned expenditure data from a single organisation (500 tuples) and we created a set of target examples (50 tuples) for the scenario.

4.1.3 Experimental Environment

The experiments have been conducted on an Amazon EC2 cluster consisting of 10 *t2.medium instances* (2 vCPUs, 4 GBs each) and a shared nothing PostgreSQL database setup. The runtime results have been obtained by averaging 10 runs.

4.1.4 Measuring Wrangling Quality

There is no benchmark available for measuring the quality of a data wrangling process. While the methods used in all stages of the wrangling process have been evaluated separately (see [15], [23], [24], [25], [36]), the approach of applying instance-based evidence to them, individually or together, has not. Our focus is on measuring the effectiveness of informing the complete wrangling process with data context. However, to explain the effect of different evidence types in more depth, we also drill down using separate experiments on individual stages.

To measure the quality for single stages and for the whole wrangling process we created a ground truth for both test scenarios by hand. The ground truth provides correctly transformed, cleaned and integrated values and correct schematic correspondences for the set of sources to be selected. We sorted the tuples of all sources according to their quality with respect to the available evidence by using approximate string distances and the number of nulls. We checked the overall tuple sorting by hand and repaired the corresponding values, matches and mappings (for these specific sources) for the best fitting tuples (targeted size = 1000) by hand.

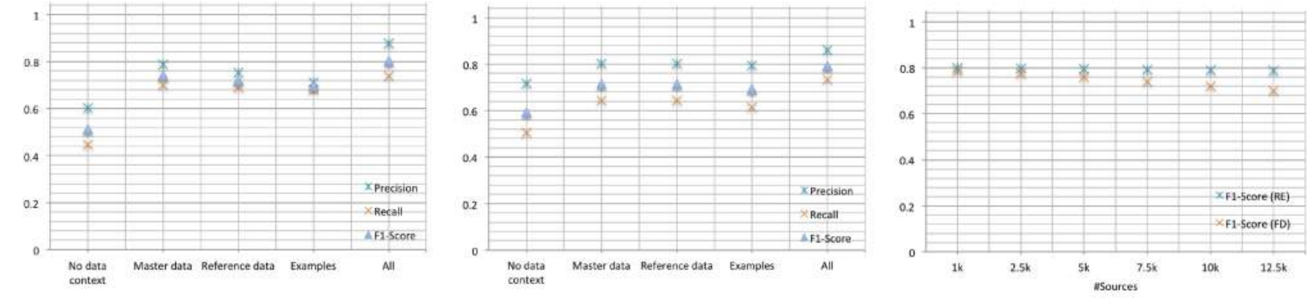
We report on the results obtained in both scenarios and exemplify the gain with the real-estate example. We use the following metrics:

- *Precision*: the fraction of correct items among the retrieved items
$$PPV = \frac{TP}{TP + FP}.$$
- *Recall*: the fraction of correct items that have been retrieved
$$TPR = \frac{TP}{TP + FN}.$$
- *f-measure*: the harmonic mean of precision and recall
$$F_1 = 2 * \frac{PPV * TPR}{PPV + TPR}.$$
- *Accuracy*: the fraction of true items among all items
$$ACC = \frac{TP + TN}{TP + FP + TN + FN}.$$
- *Negative predictive value*: the fraction of negative predictions that are correct
$$NPV = \frac{TN}{TN + FN}.$$

[6] English indices of deprivation: https://www.gov.uk/government/statistics/english-indices-of-deprivation-2015

[7] London Schools Atlas: https://data.london.gov.uk/dataset/london-schools-atlas

[8] School inspections: https://www.gov.uk/government/statistics/maintained-schools-and-academies-inspections-and-outcomes-as-at-31-march-2017



(a) Wrangling process (Real-estate) (b) Wrangling process (Financial data) (c) Scalability (All)

Fig. 3. Experimental results: Data wrangling process.

As we evaluate different stages as well as the whole wrangling process, the specific notions of true and false positives (TP, FP) and true and false negatives (TN, FN) are given in the descriptions of the corresponding experiments.

4.2 Effect of Data Context on the Wrangling Result

The objective of the first experiment is to evaluate the effect of different data context types on the end-to-end data wrangling process. The propositions to be tested are that: 1) the wrangling process in total can benefit from being informed by data context, 2) using multiple data context types together is able to improve the overall wrangling result, 3) each data context type can be used to improve the results of at least a single step, 4) data context can be used to gain combined effects on multiple wrangling stages, and 5) the effect of data context can be achieved with an increasing number of input sources.

To evaluate the results, we report on precision, recall, f-measure, and accuracy according to the ground truth. To calculate the metrics, we added a provenance id for each tuple that is internally preserved through all wrangling stages. We compare all values of the resulting tuples with the ground truth by applying the following definitions: **TP** a value in the result that corresponds to a value in the ground truth; **FP** a value in the result that does not correspond to a value in the ground truth; **FN** a value in the ground truth that should be but is not in the result.

We conducted experiments informing all steps of data wrangling with each type of evidence separately, and applying all types at each stage. An overview of the results is depicted in Figs. 3a (RE) and 3b (FD). In the case of *No data context* the process executes Coma schema-based matchers, and mappings are validated and selected based on two criteria, i.e., the completeness (number of nulls) and the size of the sources.

4.2.1 Effect of Multiple Data Context Types

The results in Figs. 3a (RE) and 3b (FD) show that applying all data context types at once in the whole wrangling process results in better target values than both not informing the process and applying single ones. There is a gain of 0.29 (RE) and 0.20 (FD) in f-measure by applying all evidence types at once, with precision improved by 0.27 (RE) and 0.14 (FD) and recall by 0.29 (RE) and 0.23 (FD) respectively.

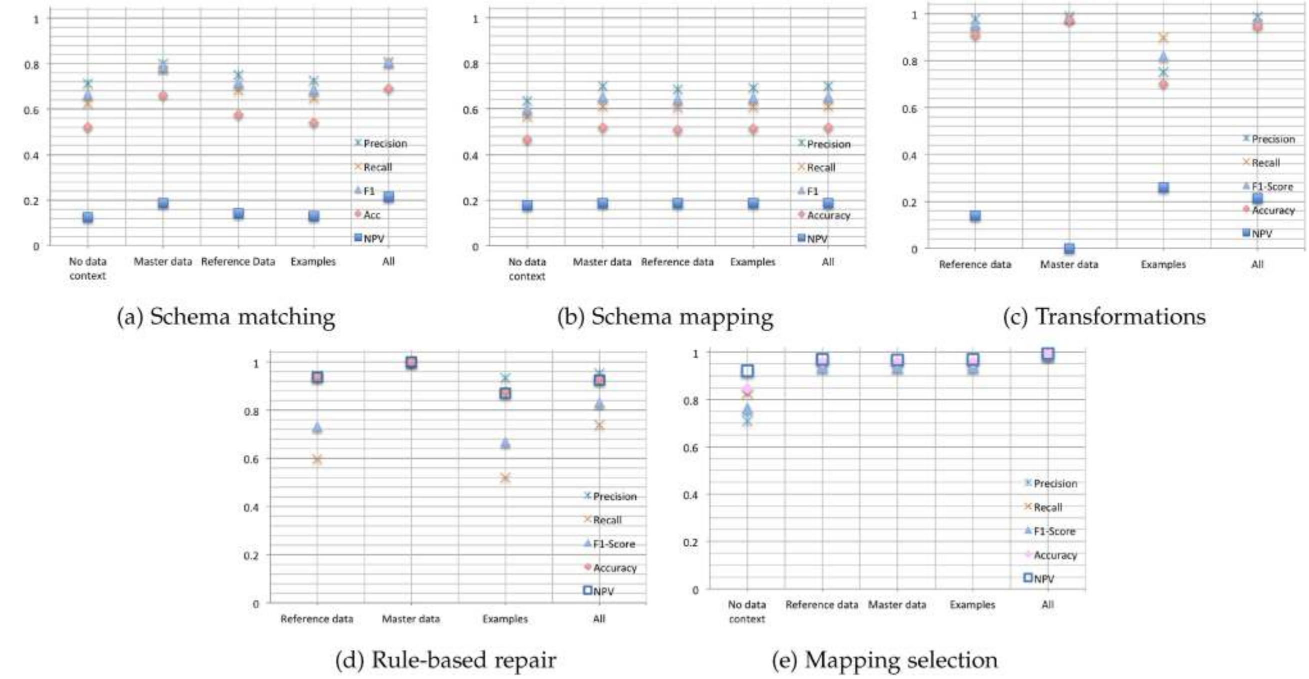
The quality of the result is greatest when all the data context types are available because each type holds different information to be exploited. For instance, master data and reference data enable the process to find different schematic correspondences. Reference data also supports format transformation of $street$ attributes into their desired representation. The format of the attribute $type$ can only be transformed by having examples at hand. Rule-based repair can correct values for attribute $agency$ based on master data, while reference data corrects $city$ and $street$ values. All data context types contribute to the selection of the best sources and tuples.

4.2.2 Effect of Different Data Context Types

The results for individual data context types show that the f-measure of the target improved by 0.23, 0.21 and 0.18 (RE) and 0.12, 0.12 and 0.10 (FD) for master data, reference data and examples, respectively. We report a gain in precision of 0.19, 0.15 and 0.11 (RE) and recall is increased by 0.25, 0.24 and 0.23 (RE). In the financial data scenario precision is increased by 0.08, 0.08 and 0.07 and recall is increased by 0.14, 0.14 and 0.11. Summarising, this shows that each type has had a positive effect, and that there is a combined effect by applying them together. In addition, the improvements do not sum as the gains in the result quality partly overlap, e.g., the same match can be corrected with master data and reference data.

4.2.3 Effect of Number of Input Sources

To evaluate the effectiveness of the approach if the number of non-related sources is increased, we scale the financial data scenario from the initial set of 180 sources up to all sources in the category *governmental spending* (12.5k). In the real-estate scenario, we scaled the number of sources from 1k to 12.5k to test the claim that the wrangling process is capable of obtaining (almost) the same result quality if the same value (input tuples) is available in a larger set of sources. To achieve this we randomly distributed the set of input tuples to the sources created by reusing the source schemes at random. The results (see Fig. 3c) show that informing the data wrangling process with data context is robust if (1) relevant data is available in many sources (RE - 0.81 to 0.79), and if (2) relevant data is wrangled from a large set of diverse sources (FD - 0.79 to 0.7). The expected decrease in the f-score is explained by the availability of more similar data domains which increases the error rates in schema matching and mapping selection (data context-based criteria).



(a) Schema matching

(b) Schema mapping

(c) Transformations

(d) Rule-based repair

(e) Mapping selection

Fig. 4. Experimental results: Effect of data context types on individual stages.

4.3 Effect of Data Context on Wrangling Steps

The objective of the second experiment is to investigate how, and to what extent, each wrangling step has benefited from available data context. We report on the result obtained from the real-estate scenario.

4.3.1 Schema Matching

We report on metrics of detected schematic correspondences in Fig. 4a by using the following definitions relating to one-to-one schematic correspondences: TP - the two elements represent the same concept; FP - the two elements do not represent the same concept; FN - a correct correspondence that is not returned; TN - an incorrect correspondence that is not returned.

We compare the results of using Coma++ with a metadata match workflow with that of a match workflow informed by evidence, i.e., instance-based matchers and domain recognisers. In the first case Coma++ is executed with workflow type 7001. With evidence, workflow 7008 is used, including instance-based matchers and domain recognisers (see Section 3.2). In both cases, the workflow is configured to select multiple matches and to match in both directions.

The results in Fig. 4a show that all metrics can be improved by all data context types, but the effect is different. The differences in the match process results are partly cumulative as instances from different types can inform the detection of different schematic correspondences, which is reflected in the gain of 0.13 in the f-measure in the case of applying all data context types.

Schema-based matchers result in a relatively high number of false positives and negatives. Applying master data results in the highest improvement of precision and recall. The reason is that master data is related to more attributes of the target, which increases the potential gain. In general, most improvements are achieved with domain recognisers as they work well to reduce the similarity score for spurious candidate correspondences with aligned data context and to increase the score for correct ones.

The number of false negatives can be slightly decreased using all kinds of evidence (see the gain in NPV). Examples show the smallest increase in the result quality. The reason is that values in the example set are less diverse than in reference data or master data. This makes it harder to make a well informed decision regarding a correct or incorrect match, as hits occur less often.

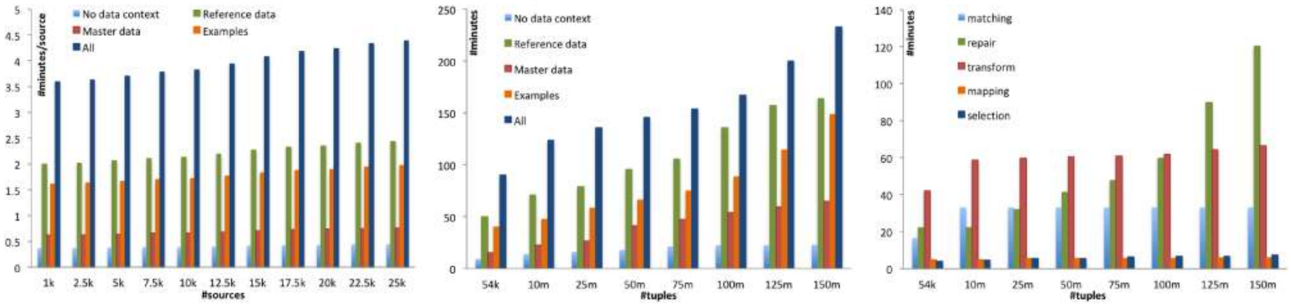
4.3.2 Value Format Transformations

We report on results for validated transformations in Fig. 4c by using the following definitions: TP - the transformation produces the correct output; FP - the transformation produces an incorrect output; FN - a source value that should be transformed is not transformed; TN - a source value that cannot be transformed is not transformed.

The value transformation algorithm detects columns that can potentially be transformed and computes source-target example pairs for them. In our experimental setting, the algorithm finds partially overlapping sets of columns to be transformed, when using reference data, master data and examples showing their possible incremental effect.

We obtained high precision, recall and accuracy but low negative predictive value for all data context types. The low NPV in all cases shows that there are far more false negatives than true negatives. However, high accuracy reports on a low rate of (true and false) negatives compared to the positive values in all cases which slightly diminishes this result.

In case of master data, the results show precision and recall of over 0.98. In this case, no TN occurred, resulting in an NPV of zero. Applying reference data results in a precision of 0.98 and a recall of 0.92. The low NPV results from null values and wrong values (e.g., a wrongly extracted city name in a street attribute) in the sources. The worst case



(a) Effect of number of sources (b) Effect of evidence types (c) Effect of wrangling steps

Fig. 5. Experimental results: Scalability of data wrangling process.

with respect to the quality metrics (precision of 0.75, accuracy of 0.7) occurred when applying examples. It appears that the combination of source and examples is, in some cases, not expressive enough to enable the required transformation to be synthesised.

4.3.3 Rule Based Data Repair

To evaluate rule-based data repair we report on precision, recall, f-measure, accuracy, and NPV of repairs executed (see Fig. 4d) and apply the same metrics as used for data format transformations.

The system discovers CFDs from data context by automatically finding the configuration settings and validating the CFDs. For the experiments we configured a minimal support size of 5 for CFDs. The CFD discovery algorithm and validation steps (lines 6 to 15 in Algorithm 4) set support sizes of 17, 11, and 8 for examples, reference data and master data. The CFDs covered 13 different repair rule patterns with two being incorrect (79 rules) (e.g., $cfd_1 : ([street, city] \rightarrow [postcode], T_p)$). Experimental results show that automatic CFD validation is able to configure CFD discovery effectively with respect to correct and incorrect CFDs.

We report high precision, accuracy, negative predictive value and average recall for all data context types except master data (recall). In the case of master data, all selected repairs are correct and all potential repairs are found. We report a precision of 0.95 for reference data. However, in this case some opportunities have been missed, resulting in a recall of 0.6. Results for examples are in a similar range: precision of 0.93 and recall of 0.52. Negative predictive value is in all cases above 0.87. Combined with the high recall, this shows that our approach resulted in a low number of false negatives.

4.3.4 Schema Mapping Validation

We report on mapping validation according to the ground truth (see Fig. 4b) by using the same notions of TP, FP, FN, and TN as for target quality (see Section 4.2).

Schema mapping generation and validation uses as input schema matches and profile data, such as candidate keys and partial inclusion dependencies. We calculate data context-based validation scores for each mapping and rank them accordingly. The additional benefit is not as substantial as in the other stages, but in our experimental setting this process is often able to select a mapping with a good result. In general, utilising data context for mapping validation leads to an increase of 0.05 to 0.06 in f-measure due to an improvement in both precision and recall.

4.3.5 Mapping Selection

We report on precision, recall, f-measure, accuracy, and NPV of the selected sources and tuples according to the defined ground truth (see Section 4.1.4) for mapping selection in Fig. 4e. In general, we use the same ground truth as for the overall wrangling result validation, but measure the correctly selected tuples as follows: TP - the algorithm selects a correct tuple; FP - a tuple that is selected but should not be; FN - a tuple that should be selected but is not; TN - a tuple that is not selected and should not be.

To calculate data context-based criteria for mappings, we use the Jaccard containment score with the actual value as the query and the data context as the source and we combine the attribute metrics in a single criterion for each data context type (see Section 3.6). The algorithm optimises the consistency, accuracy, and relevance of the selected mappings and tuples.

The experiments show that utilising each single data context type to inform mapping selection immediately improves precision from 0.7 to approximately 0.93 and recall from 0.83 to 0.94. In case where all data context items are available, precision and recall can be improved to 0.98. This rather good result can be achieved because the sources, as many others, follow a power law distribution (as also reported in [23] or [41]). Therefore, a small amount of information is capable of correctly guiding the search.

4.4 Performance Evaluation

To study the efficiency of informing the wrangling process with data context we report on the scalability of the approach with respect to the number of tuples to be wrangled and the number of sources used as input to the process. We report on the effect of different evidence types and of the individual wrangling stages. The performance of all described algorithms for automating wrangling is linearly dependent on the number of sources and/or the number of evidence types. Our to-be-evaluated claim is that the wrangling process is scalable, though the actual runtime depends on the chosen concrete methods for individual wrangling stages that are executed within the presented algorithms. The reported time (see Fig. 5) is the response time of the algorithms executed on the real-estate scenario, i.e., the time the user has to wait before the wrangling results are available.

The number of sources and the number of tuples per source have been chosen according to three use cases. In the real-estate scenario, there exist 1k sources with on average 57 tuples. From the UK Open Government Data (OGD) portal we scraped 25k csv files with on average 4.4k tuples per file (Business, Defence and Environment). In contrast, Data Civilizer [45] reports on the MIT Data Warehouse with 3k tables, in total 1TB of data. These tables can be assumed to be larger in number of tuples (according to the average tuple size, 2.5kb, in our sources, we estimate 140k tuples per source). In our experiments, we chose to scale up to 25k sources (in accordance to OGD) and we scale the number of tuples in single sources from 54 (RE) on average up to 150k (MIT) per source to emulate these differences.

4.4.1 Effect of Number of Sources

The scalability of the wrangling process has been evaluated by increasing the number of input sources from 1k to 25k. The input sources have been created on the basis of the real-estate scenario by generating new input sources with the same characteristics. The results (see Fig. 5a) show that the process scales, according to the performance analysis, with respect to the number of sources. The average runtime per source increases from 3.6 minutes for 1k sources (all data context) to 4.4 minutes with 25k sources and is executed on 10 multi-threaded virtual machines (see Section 4.1.3). The whole process except mapping selection can be executed in parallel for the available sources (by incorporating their clusters). Our experiments show an average speedup of 9.12 and 18.27 for parallelising on 10 and 20 nodes (average over 1k to 25k input sources).

4.4.2 Effect of Data Source Size

The scalability of the wrangling process with respect to the data size has been evaluated by increasing the number of tuples in a scenario with 1k input sources from 54k to 150M (54 to 150k per source) in total according to our three baseline scenarios (Section 4.4). In addition, 0.03 percent of sources in the OGD dataset have more than 150k tuples, supporting our claim of building a realistic scenario at scale. The empirical results (see Fig. 5b) show that the process is scalable with respect to the number of tuples, with and without using evidence. The baseline run shows that execution time is dominated by the workflow itself in the case of very small sources. An important observation is that the type of evidence affects the runtime. The difference in the effect of evidence types on the runtime of the individual steps is explained in the information gain achieved by the evidence type (e.g., number of columns to be transformed, number of repair rules discovered), and dependent on the concrete implementation of the specific step. As expected, the runtime increases with bigger source sizes but the effect of informing the process with evidence is stable and relative to the input size. For instance, for 50M input tuples, the runtime increases from 18 minutes (no data context) to 96, 41, 66, and 146 minutes if informed with evidence (reference data, master data, examples, all).

4.4.3 Effect of Wrangling Steps

We report on the effect of the input size on the runtime of the individual wrangling steps. Again, we increase the number of tuples in the scenario with 1k sources from 54k to 150M. As shown in Fig. 5c for reference data, the performance of matching, mapping generation and mapping selection is close to constant if the number of input tuples is increased. This result is expected as in all three steps our approach samples a fixed amount of tuples from the sources and the provided data context. Value format transformation and repair have to take all input and data context tuples into account and thus their runtime is directly affected by the size as well as by the concrete implementation of the wrangling method. In general, we demonstrate that the wrangling process and wrangling step algorithms scale with the number of tuples and that the experimental results of the concrete method implementations, that could be exchanged, are in line with their performance analysis (see [24],[25]). For instance, moving to a less expressive (e.g., considering only FDs instead of CFDs) but also less expensive repair algorithm would significantly improve repair performance.

5 Related Work

In this section, we briefly discuss: (i) other proposals that explicitly address data wrangling; (ii) other approaches to automated data integration and cleaning that use auxiliary evidence or address scale; (iii) other end-to-end proposals of relevance to data wrangling in the context of Big data. Additional proposals tackling individual steps have already been discussed in Section 3.

Proposals that address data wrangling explicitly, often focus on supporting the data scientist on specific tasks, such as format transformation. For example, the Wrangler system [10] proposes format transformation rules, which can be selected or revised by the data scientist. In FlashFill [31] and related proposals, users provide examples that underpin transformation program synthesis. Such human-in-the-loop proposals complement this work, as they can be applied for parts of a process where automation has not produced a satisfactory result. In DataXFormer [14], identification of transformations is informed by the contents of web tables, which can be seen as a form of data context. Thus, DataXFormer provides an example of a single step in the wrangling process being informed by data context, whereas the contribution of this paper is to exploit such contextual information throughout the process.

In [38] the authors present an approach for optimising complex queries against a diverse set of sources including NoSQL and relational data sources. In common with this approach, we utilise dynamic programming to optimise integration plans based on an internal cost model. In contrast, we focus on integrating a large set of diverse sources (up to 25k) while their focus seems to be more on large data sets (up to 200 GB). The authors of [44] present an indexing architecture for storing and searching media databases characterised by high-dimensional feature vectors. While this approach focuses on scaling with respect to the number of dimensions, in mapping selection, we utilise a scalable indexing approach to compare integration plans with the data context that is tailored towards the power-law distribution of many large scale Web-extracted data sets [41].

A range of proposals have been made for data integration and cleaning components that use auxiliary data (e.g., [13], [14], [15], [16]). Additional evidence can also be obtained, though at further cost, through crowdsourcing [43]. We describe an approach that can build upon and make more effective use of such results, maximising the potential benefit from contextual data by using the same data in multiple steps.

An end-to-end approach to data wrangling is often obtained by data scientists hand crafting integration and cleaning programs, in a manner that is analogous to the writing of ETL workflows for warehouses. Our objective is to reduce the associated manual effort through increased use of automation. This requirement has been recognised by others, and the nearest proposal in terms of its goals is Data Civilizer [46], which uses profiling information to inform join path generation. However, Data Civilizer does not benefit from the systematic use of data context that is described here; it could be extended to do so. On the contrary, Big data analytics platforms such as [7] and [6] focus on optimising the execution of composable data anlytics workflows according to data locality and data flow. Our platform focuses on the design and implementation of a scalable and modularised data wrangling workflow in a domain-independent manner. The execution of this workflow could benefit from the features provided by such underlying Big data analytics platforms. The NIST Big Data Public Working Group presents a big data interoperability framework [4] that includes use cases and requirements for Big data systems and applications. The specification includes use cases (e.g., Web search and text-based data) dealing with finding and integrating relevant data, including data preprocessing to identify what is for instance searchable. Our approach supports preparing and selecting the relevant data entities from a vast amount of semi-structured data that has been extracted from the deep Web by exploiting instance-based evidence.

Different and complementary notions of data context have been given in previous work on context aware systems (e.g., [47]). In such proposals, the focus is to identify the subset of an extent that is most appropriate for a user in a given situation. In contrast, our notion of data context emphasises data from the domain within which data wrangling occurs. The term data context is also used in the proposal for the Ground data context service [48], which is used to capture metadata and annotations relating to diverse data sets. Our notion of data context would seem to be suitable for capturing and sharing using a platform such as Ground.

6 Conclusions

Data scientists spend as much as 80 percent of their time on data wrangling,[9] so cost-effective data wrangling tackling integration (addressing variety), cleaning (addressing veracity), and selection of sources (addressing volume) is crucial to the successful use of big data. In this paper, using two representative real world examples, we show an improvement in f-score from 0.51 to 0.81 and from 0.59 to 0.8 by automating and optimising a five step wrangling process to use data context throughout. As such this paper has presented a cost-effective and scalable methodology for enhanced automation that provides a significant return on investment and supports automatic integration, repairing and selection of data by optimising the accuracy, consistency and relevance of the wrangling result. As illustrated in a demo paper, data scientists can associate data context with a target schema with modest effort via the VADA user interface [27].

Experiments show that data context is able to inform all stages of the wrangling process in different ways: in *matching* by extending the collection of matchers that can be applied from schema-based to instance-based matchers and domain recognisers; in both *format transformation* and *repair* by enabling rules to be learned and validated; in *mapping generation and validation* by allowing mapping validation to be informed by the results of instance-based similarity measures; and in *mapping selection* by calculating data context-specific criteria (i.e., Jaccard set containment scores) corresponding to quality dimensions to guide the optimisation and selection process.

We show that applying multiple data context types on several wrangling stages results in a combined gain in the quality of the final wrangling result; first, by summing the effect of different data context types within a stage, and second, by accumulating results from different stages. For instance, data format transformation and data repair can benefit from additional correspondences detected by data context-informed schema matchers.

We evaluate scalability of the wrangling process in terms of number of sources, number of tuples and the effect of individual stages. We report that the proposed wrangling process is scalable from 1k to 25k input sources and from 54k to 150M tuples. Experiments show that overall runtime depends on the efficiency of the concrete wrangling methods used and that it is affected by the type of data context. In our approach, runtime is dominated by value format transformations and rule-based repair.

Several extensions are targeted for future work. We will investigate the effect of additional data context types on the wrangling pipeline, and on other wrangling stages such as Web data extraction[49]. To further address time-varying variety and veracity problems in data wrangling, we will investigate feedback-based learning and model refinement techniques such as presented in [42] or [50]. Furthermore, we are exploring how to combine evidence gained from data context with user preferences, as shown in [23], to elaborate the possibilities in tailoring a data product for users with different requirements. Complementing our methodological work, we aim towards a systematic architectural model [27] supporting dynamic and incremental orchestration of diverse data wrangling methods and datasets as described.

Acknowledgments

This research is supported by the VADA[10] grant from the UK Engineering and Physical Sciences Research Council EPSRC, grant number EP/M025268/1.

References

[1] S. Abiteboul, M. Arenas, P. Barceló, M. Bienvenu, D. Calvanese, C. David, R. Hull, E. Hüllermeier, B. Kimelfeld, L. Libkin, W. Martens, T. Milo, F. Murlak, F. Neven, M. Ortiz, T. Schwentick, J. Stoyanovich, J. Su, D. Suciu, V. Vianu, and K. Yi, “Research directions for principles of data management (dagstuhl perspectives workshop),” *Dagstuhl Manifestos*, vol. 7, no. 1, pp. 1–29, 2018.

[9] New York Times://http://nyti.ms/1Aqif2X

[10] VADA: http://vada.org.uk

[2] I. Terrizzano, P. M. Schwarz, M. Roth, and J. E. Colino, “Data wrangling: The challenging yourney from the wild to the lake,” in *Proc. Conf. Innovative Data Syst. Res.*, 2015.

[3] F. Endel and H. Piringer, “Data wrangling: Making data useful again,” *IFAC-PapersOnLine*, vol. 48, no. 1, pp. 111–112, 2015.

[4] NBD-PWG, “NIST big data interoperability framework: Volume 3, use cases and general requirements,” vol. 2, 2018.

[5] J. Chen and H. Wang, “Guest editorial: Big data infrastructure,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 148–149, Jun. 2018.

[6] D. Wu, L. Zhu, Q. Lu, and S. Sakr, “HDM: A composable framework for big data processing,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 150–163, Jun. 2018.

[7] Y. Kaniovskyi, M. Koehler, and S. Benkner, “A containerized analytics framework for data and compute-intensive pipeline applications,” in *Proc. 4th ACM SIGMOD Workshop Algorithms Syst. MapReduce Beyond*, 2017, pp. 6:1–6:10.

[8] M. Koehler, “An adaptive framework for utility-based optimization of scientific applications in the cloud,” *J. Cloud Comput.*, vol. 3, no. 1, May 2014, Art. no. 4.

[9] A. R. Hummaida, N. W. Paton, and R. Sakellariou, “Adaptation in cloud resource configuration: A survey,” *J. Cloud Comput.*, vol. 5, no. 1, May 2016, Art. no. 7.

[10] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, “Wrangler: Interactive visual specification of data transformation scripts,” in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 3363–3372.

[11] M. Koehler, A. Bogatu, C. Civili, N. Konstantinou, E. Abel, A. A. Fernandes, J. A. Keane, L. Libkin, and N. W. Paton, “Data context informed data wrangling,” in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 956–963.

[12] T. Furche, G. Gottlob, L. Libkin, G. Orsi, and N. W. Paton, “Data wrangling for big data: Challenges and opportunities,” in *Proc. 19th Int. Conf. Extending Database Technol.*, 2016, pp. 473–478.

[13] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang, “DIADEM: Thousands of websites to a single database,” *Proc. VLDB Endow.*, vol. 7, no. 14, pp. 1845–1856, Oct. 2014.

[14] Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker, “DataXFormer: A robust transformation discovery system,” in *Proc. IEEE 32nd Int. Conf. Data Eng.*, May 2016, pp. 1134–1145.

[15] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, “Schema and ontology matching with coma++,” in *Proc. SIGMOD Int. Conf. Manage. Data*, 2005, pp. 906–908.

[16] O. Lehmberg and C. Bizer, “Stitching web tables for improving matching quality,” *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1502–1513, Aug. 2017.

[17] M. Stonebraker, “The seven tenets of scalable data unification,” Tamr Inc., 2017.

[18] S. Gulwani, “Programming by examples - and its applications in data wrangling,” in *Dependable Software Systems Engineering*. Amsterdam, Netherlands: IOS Press, 2016.

[19] C. Batini and M. Scannapieco, *Data and Information Quality: Dimensions, Principles and Techniques*, 1st ed. Berlin, Germany: Springer Publishing Company, 2016.

[20] A. Rostin, O. Albrecht, J. Bauckmann, F. Naumann, and U. Leser, “A machine learning approach to foreign key discovery,” in *Proc. 12th Int. Workshop Web Databases*, 2009.

[21] T. Papenbrock and F. Naumann, “A hybrid approach to functional dependency discovery,” in *Proc. Int. Conf. Manage. Data*, 2016, pp. 821–833.

[22] W. Fan, F. Geerts, J. Li, and M. Xiong, “Discovering conditional functional dependencies,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 5, pp. 683–698, May 2011.

[23] E. Abel, J. Keane, N. W. Paton, A. A. Fernandes, M. Koehler, N. Konstantinou, J. C. C. Rios, N. A. Azuan, and S. M. Embury, “User driven multi-criteria source selection,” *Inf. Sci.*, vol. 430-431, pp. 179–199, 2018.

[24] A. Bogatu, N. W. Paton, and A. A. A. Fernandes, “Towards automatic data format transformations: Data wrangling at scale,” in *Proc. Brit. Int. Conf. Databases*, 2017, pp. 36–48.

[25] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, “Improving data quality: Consistency and accuracy,” in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 315–326.

[26] K. Christodoulou, N. W. Paton, and A. A. A. Fernandes, “Structure inference for linked data sources using clustering,” in *Transactions on Large-Scale Data- and Knowledge-Centered Systems*. Berlin, Germany: Springer, 2015, pp. 1–25.

[27] N. Konstantinou, M. Koehler, E. Abel, C. Civili, B. Neumayr, E. Sallinger, A. A. Fernandes, G. Gottlob, J. A. Keane, L. Libkin, and N. W. Paton, “The VADA architecture for cost-effective data wrangling,” in *Proc. ACM Int. Conf. Manag. Data*, 2017, pp. 1599–1602.

[28] F. Naumann, “Data profiling revisited,” *ACM SIGMOD Record*, vol. 42, no. 4, pp. 40–49, 2014.

[29] D. Zhao, K. Qiao, Z. Zhou, T. Li, Z. Lu, and X. Xu, “Toward efficient and flexible metadata indexing of big data systems,” *IEEE Trans. Big Data*, vol. 3, no. 1, pp. 107–117, Mar. 2017.

[30] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti, “Extraction and integration of partially overlapping web sources,” *Proc. VLDB Endow.*, vol. 6, no. 10, pp. 805–816, Aug. 2013.

[31] S. Gulwani, “Automating string processing in spreadsheets using input-output examples,” in *Proc. 38th Annu. ACM SIGPLAN-SIGACT Symp. Principles Programm. Lang.*, 2011, pp. 317–330.

[32] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, “Truth finding on the deep web: Is the problem solved?” *Proc. VLDB Endow.*, vol. 6, pp. 97–108, 2013.

[33] D. Zhang, D. Wang, N. Vance, Y. Zhang, and S. Mike, “On scalable and robust truth discovery in big data social media sensing applications,” *IEEE Trans. Big Data*, 2018, doi: 10.1109/TBDATA.2018.2824812

[34] Z. Abedjan, X. Chu, D. Deng, R. Fernandez, I. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang, “Detecting data errors: Where are we and what needs to be done?” *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 993–1004, Aug. 2016.

[35] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Towards certain fixes with editing rules and master data,” *Proc. VLDB Endow.*, vol. 3, no. 1/2, pp. 173–184, Sep. 2010.

[36] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa, “Schema mapping verification: The spicy way,” in *Proc. 11th Int. Conf. Extending Database Technol.*, 2008, pp. 85–96.

[37] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri, “S4: Top-k spreadsheet-style search for query discovery,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 2001–2016.

[38] R. Sellami and B. Defude, “Complex queries optimization and evaluation over relational and NoSQL data stores in cloud environments,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 217–230, Jun. 2018.

[39] K. Belhajjame, N. Paton, S. Embury, A. Fernandes, and C. Hedeler, “Incrementally improving dataspaces based on user feedback,” *Inf. Syst.*, vol. 38, no. 5, pp. 656–687, 2013.

[40] L. Bellomarini, G. Gottlob, A. Pieris, and E. Sallinger, “Swift logic for big data and knowledge graphs,” in *Proc. Int. Conf. Current Trends Theory Practice Informat.*, 2018, pp. 3–16.

[41] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, “LSH ensemble: Internet-scale domain search,” *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1185–1196, Aug. 2016.

[42] J. C. Cortés Ríos, N. W. Paton, A. A. A. Fernandes, E. Abel, and J. A. Keane, “Targeted feedback collection applied to multi-criteria source selection,” in *Proc. 21st Conf. Adv. Databases Inf. Syst.*, 2017, pp. 136–150.

[43] V. Crescenzi, A. A. A. Fernandes, P. Merialdo, and N. W. Paton, “Crowdsourcing for data management,” *Knowl. Inf. Syst.*, vol. 53, no. 1, pp. 1–41, Oct. 2017.

[44] A. Iscen, T. Furon, V. Gripon, M. Rabbat, and H. Jegou, “Memory vectors for similarity search in high-dimensional spaces,” *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 65–77, Mar. 2018.

[45] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang, “The data civilizer system,” in *Proc. 8th Biennial Conf. Innovative Data Syst. Res.*, 2017.

[46] R. Castro Fernandez , D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, “A demo of the data civilizer system,” in *Proc. Int. Conf. Manage. Data*, 2017, pp. 1639–1642.

[47] C. Bolchini, C. A. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca, “And what can context do for data?” *Commun. ACM*, vol. 52, no. 11, pp. 136–140, Nov. 2009.

[48] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, S. Chang, C. Steinbach, V. Subramanian, and E. Sun, “Ground: A data context service,” in *Proc. Conf. Innovative Data Syst. Res.*, 2017.

[49] D. Qiu, L. Barbosa, V. Crescenzi, P. Merialdo, and D. Srivastava, "Big data linkage for product specification pages," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 67–81.

[50] Y. Cong, J. Liu, B. Fan, P. Zeng, H. Yu, and J. Luo, "Online similarity learning for big data with overfitting," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 78–89, Mar. 2018.

Martin Koehler received the PhD degree in computer science from University of Vienna, Austria, in 2012. He is a research associate with the Information Management Group, University of Manchester and a lecturer with the University of Vienna. Until 2015, he worked as a scientist on big data research at the Austrian Institute of Technology. His main research interests include data integration and management at scale, autonomic and cloud computing, and big data. He is a member of the IEEE.

Edward Abel received the PhD degree in the School of Computer Science. He is a research associate with the University of Manchester, School of Computer Science. He is part of the Decision and Cognitive Sciences Research Group, Information Management Group and the Data Science Institute. He is currently working upon the VADA (Value Added DAta systems) project.

Alex Bogatu is working toward the PhD degree in the Information Management Group, University of Manchester. His research interests are focused on data wrangling in general, with data format transportations and data discovery in particular. He previously worked as a system administrator in the telecommunications industry.

Cristina Civili received the PhD degree in engineering in Computer Science from Sapienza, University of Rome, Italy, in 2016. She is a research associate with the School of Informatics, University of Edinburgh. She is part of the Database Research Group, where she works in the VADA project on incomplete information and data quality. Her research interests include incomplete information, data quality, ontology-based data access and knowledge representation.

Lacramioara Mazilu received the master's degree at University Politehnica of Bucharest. She is working toward the PhD degree in the Information Management Group, University of Manchester. Previously, she has worked in industry for the 112 Emergency System and for the Special Telecommunications Service as a software engineer, in Romania. Her research interests lie in the field of data integration, big data, and text mining.

Nikolaos Konstantinou is a research fellow with the Information Management Group, University of Manchester, working in the VADA project. Before that, he has held a variety of technical management and research roles in several research and development projects. His research interests are in data preparation and information management. He is a member of the IEEE.

Alvaro A. A. Fernandes is a senior lecturer with the University of Manchester, where he has been since 1998. He was awarded a PhD in Computer Science by Heriot-Watt University, in 1995. He has done research on novel data models and systems and on query processing. Currently, his focus is on value added data systems.

John Keane is professor of data engineering with the School of Computer Science, University of Manchester. He has previous commercial experience with Phillips, Fujitsu and the TSB Bank. He is an associate editor of the *IEEE Transactions on Fuzzy Systems*. His research focuses on data analytics and decision science.

Leonid Libkin is a professor with the University of Edinburgh, where he holds the chair of Foundations of Data Management with the School of Informatics. He previously worked at the University of Toronto and Bell Labs. His current interests include incomplete information, data wrangling, and graph databases. He is the author of textbooks on finite model theory and data exchange. He is an ACM and a Royal Society of Edinburgh fellow, and a member of Academia Europaea.

Norman W. Paton is a professor of computer science at the University of Manchester. He has had a range of roles at Manchester, including as Head of School. Current research interests include pay-as-you-go data integration, data wrangling and adaptive systems. He is a member of Academia Europaea, President of the EDBT Association, and a member of the Editorial Board of Distributed and Parallel Databases.