Schema Mapping Generation in the Wild: A Demonstration with Open Government Data

Lacramioara Mazilu
University of Manchester, United Kingdom
lara.mazilu@manchester.ac.uk

Nikolaos Konstantinou
University of Manchester, United Kingdom
nikolaos.konstantinou@manchester.ac.uk

Norman W. Paton
University of Manchester, United Kingdom
norman.paton@manchester.ac.uk

Alvaro A.A. Fernandes
University of Manchester, United Kingdom
alvaro.a.fernandes@manchester.ac.uk

ABSTRACT

Schema mapping generation identifies how data sets can be combined to create views that are relevant to an application. Where the data sets to be combined lack declared relationships, such as foreign keys, schema mapping generation can be considered to be *in the wild*. In this paper, we describe an approach to schema mapping generation in the context of open government data, in particular, the London Datastore. Mapping generation is informed by inferred profiling data about the data sets and their relationships, where the data sets are made available as csv files. We outline the mapping generation algorithm, and describe a demonstration of the approach, in which the user can: *(i)* specify the target to be populated by the generated mappings over a collection of sources from The London Datastore; *(ii)* browse the generated candidate mappings and the evidence that informed their creation; and *(iii)* steer the mapping generation process, to make use of preferred sources and dependable profiling results.

1 INTRODUCTION

Given a collection of source datasets, some metadata about them, and a target schema, schema mapping generation produces a collection of views that provide ways of populating the target from the sources. Mapping generation is important because the data of relevance to an application or an analysis is often not immediately available in a single, suitable, integrated form.

Most work on mapping generation has assumed that the source and the target benefit from declared constraints, for example in the form of primary and foreign keys (e.g., as in the seminal work on Clio and its descendents, as reviewed in [5]). However, with the growing availability of open data sets, and the emergence of data lakes, mapping generation over independently produced data sets, with minimal explicit metadata, is arguably even more necessary than for well-defined schemas.

We refer to mapping generation over data sets without declared relationships as *in the wild*. Mapping generation must, among other things, take into account relationships between data sources, and, in this paper, we assume that candidate keys and (partial) inclusion dependencies have been inferred through data profiling [1]. Then, to deploy schema mapping generation in the wild, the following are required:

(1) *A way of exploring the space of candidate mappings.* We use a dynamic programming algorithm to identify promising mappings, referred to as *Dynamap* [8].
(2) *A way of displaying to the user these mappings, their properties, and the evidence on which they build.* As (1) builds on necessarily speculative profiling data, the results of mapping generation must be able to be reviewed by users, for example to ensure that joins are building on inclusion dependencies that represent valid real-world relationships.
(3) *A way to enable the user to steer the mapping generation process.* As (2) may identify issues with generated mappings, users must be able to steer the mapping generation process away from unsuitable decisions, for example by ruling out the use of certain inclusion dependencies.

To show (1) to (3) in practice, we demonstrate our mapping generation algorithm, and its associated user interface, in use with data from The London Datastore[1], which provides hundreds of data sets providing diverse information about London.

The remainder of the paper is structured as follows. Section 2 provides some details on The London Datastore. Our mapping generation approach is reviewed in Section 3. The demonstration in Section 4 shows an example of viewing a generated mapping and understanding it based on its properties and the evidence based on which it was created. The user can steer mapping generation based on the presented information. Section 5 concludes.

2 OPEN DATA CASE STUDY: THE LONDON DATASTORE

Open government data is published in a collection of national, regional, city or topic-based portals, with a view to increasing transparency and informing decision making [3]. The London Datastore is a representive example of a city data repository, providing data sets across a range of topic areas, including transport, employment, housing, health and education. These datasets come from a variety of publishers, including local and national government departments, and many of the data sets use consistent, generous licenses. The London Datastore supports both search and browse interfaces, and allows data sets to be downloaded in a variety of formats.

The demonstration uses comma-separated-value file data sets, released under the UK Open Government License[2]. Typically files contain from a few tens of rows (e.g., there are numerous data sets that have one row for each London Borough, of which there are *33*), to a few thousand rows (e.g., there are around 5000 rows in a data set of modelled household income estimates at a particular, rather fine, area granularity). There may be few (e.g., 2) to many columns in each table (e.g., there are hundreds of columns in a ward atlas table, describing different properties of an electoral ward).

[1] https://data.london.gov.uk
[2] http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/

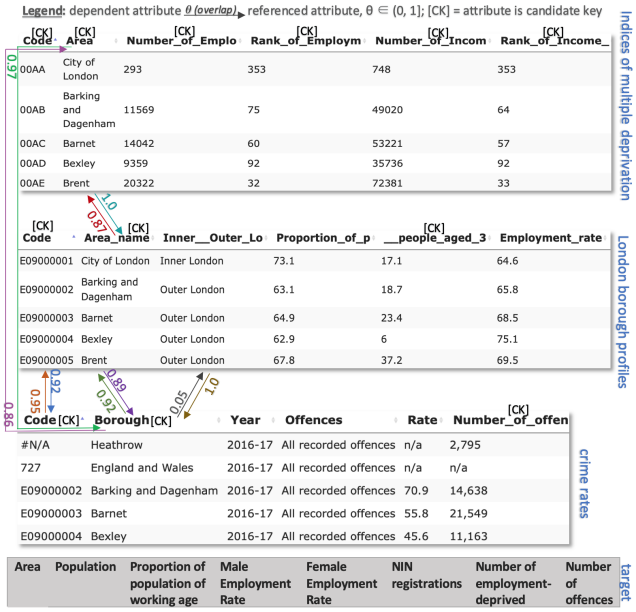

Figure 1: London datasets example

Figure 1 contains fragments (i.e., shows subsets of both tuples and attributes of the tables) from three tables in The London Datastore that are used in our running example. The tables in Figure 1 show population statistics for different London areas and (possibly their corresponding) crime rates[3].

3 MAPPING GENERATION IN DYNAMAP

In this section, we illustrate mapping generation in Dynamap with an example over The London Datastore.

The problem and some evidence. The *problem* is that of generating a set of mappings that populate a *target* table from a set of *source* tables. The target table definition provides only the names of the columns, and the source table definitions contain only column names and their associated values. Mapping generation techniques have been developed that use different types of evidence, including declared constraints [5], data examples [2] and feedback [4]. In Dynamap, the *evidence* used to inform mapping generation is *matches* between *source* and *target* attributes, and (partial) *inclusion dependencies*, both of which can be inferred using standard profiling algorithms [1]. In Figure 1, the *target* requires information about population and employment statistics per area (first 7 attributes), together with the corresponding number of offences in each area (last attribute). Also, Figure 1 shows the detected profile data on the represented sources: 10 inclusion dependencies (represented by arrows, where their direction is from the dependent to the referenced source) and 12 candidate keys (annotated with *[CK]*). The *overlap* (θ) is the fraction of the distinct values in the dependent attribute that are found in the referenced attribute. In the example, eight inclusion dependencies are partial ($\theta \in (0, 1)$) and two are full ($\theta = 1.0$).

Exploring the search space. Mapping generation is typically a search problem, either using generic (e.g., [6]) or bespoke (e.g., [5]) algorithms. In Dynamap, the space of candidate mappings is every way of combining the sources using *union* or *join* operations, where union compatibility is detected by comparing matches with the target and join operations are based on full or partial inclusion dependencies. We explore the search space using dynamic programming, but with pruning to ensure that only promising parts of the space are actually visited. As a result, mappings are constructed bottom-up, from pairs of tables, then from tables with intermediate mappings containing two tables, etc. For example, in Figure 1, a first step could be to merge *Indices of deprivation* with *London borough profiles* by joining on *Area* attributes, and then, their result intermediate mapping could be merged further with the *crime rates* table through join on either *Code* or *Area/Borough* attributes.

Annotating candidate mappings. As the search must combine pairs of intermediate mappings, we need to know the candidate keys of candidate mappings and their (partial) inclusion dependencies. These need to be derived during the search process, as it is not computationally viable to evaluate (i.e., materialize) every candidate mapping and run a profiler on it. The derived profile information is also used to compute a fitness, which represents the predicted fraction of complete rows. Formulas have been developed for propagating profiling data through unions and joins [8]. For example, in Figure 1, after joining *Indices of deprivation* with *London borough profiles* on *Area*, their corresponding inclusion dependencies and candidate keys are propagated such that it shows the relationship between the newly created intermediate mapping and the *crime rates* table. Based on this propagated profiling data, the mapping generation system can detect that the new mapping will produce *Area* and *Code* values that are overlapping with *Borough* and *Code* in *crime rates*, thus, it concludes that they can be joined on one of the overlapping attribute pairs.

4 DEMONSTRATION

Mapping generation in the wild will be demonstrated through hands-on experience generating and examining mappings over datasets from The London Datastore, in the context of the Data Preparer[4] data integration and cleaning platform, a descendent of the VADA Architecture [7]. Users interact with the system via a web interface; our goals are to demonstrate: *(i)* how data profiling can be used to provide information about the properties of the data sets; *(ii)* how this evidence can be used by Dynamap to combine sources in plausible ways; *(iii)* how users can review the candidate mappings, and the evidence on which they are based; and *(iv)* how users can steer the mapping generation process, for example by excluding schema elements or profiling data that are not relevant to the result.

Figure 2 shows an example that extends the one in Figure 1 by adding to it three other sources containing data about house and population density, national insurance registrations, and statistics on the population from the local authorities. Although the demonstration will support different ways of using the system, one approach would involve the following steps.

Browse the available sources. The interface supports search and browse over the sources that are input to the system.

Define a target. The user can interactively name a target table and its attributes, or edit an existing target to add/remove attributes. The target table used throughout is that from Figure 1.

Generate mappings and view the result. Given some source tables and a target, the user can ask the system to generate a result,

[3]Due to space limitations, throughout the rest of this paper, the figures contain attribute and table names capped at 15 characters.

[4]https://thedatavaluefactory.com

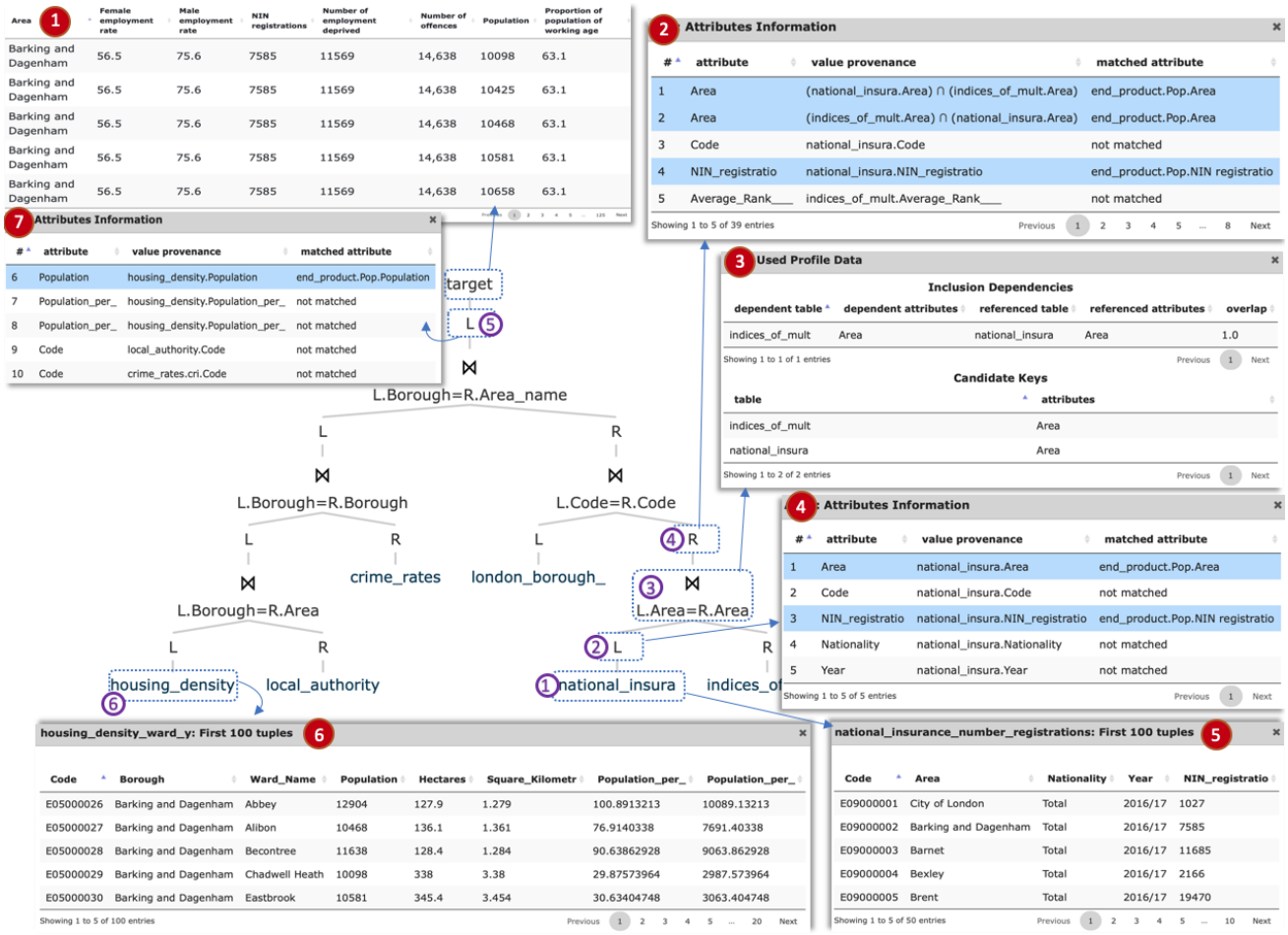

Figure 2: Mapping explanation tree and associated information

which means running a matcher, a data profiler (for detecting candidate keys and partial/full inclusion dependencies), mapping generation and result display.

View a candidate mapping. From the result, the user can ask the system to explain how the generated mappings were produced. Each generated mapping has *(i)* a view that has the schema of the chosen target table and that shows how the target is populated once the mapping is executed (e.g., ❶ in Figure 2), and *(ii)* a *mapping explanation*. Each mapping is *explained* through the help of a tree view where the user can understand in which order the sources were merged and which operators were used to merge them. Figure 2 shows how six sources are merged through five joins by a mapping. The *root* of the tree is the chosen target table and the leaves of the tree are the input sources merged by the mapping. The intermediate nodes represent the incremental build-up of the explained mapping, starting with the merge of the leaves (the sources) and reaching the target when all sources were merged or when no other merge opportunities could be identified. As mentioned in Section 3, the mappings are constructed in a bottom-up fashion, and this is shown by the tree representation, where each intermediate mapping has two corresponding nodes in the tree: *(i)* an *operation* node (e.g., ③ in Figure 2), and *(ii)* a *L/R* node (e.g., ②, ④, ⑤). The *operation* node is used to show which was the chosen operation between the two operands, and (if present) the join condition. As explained in [8], the operations considered by Dynamap are union, (equi) join or full outer join; different combinations of profiling data lead to different operations, e.g., a candidate key whose attributes share a partial inclusion dependency can lead to a full outer join, while a full inclusion dependency can lead to a join. The *L/R* node shows whether the newly created intermediate mapping is the **l**eft or **r**ight operand in the next operation (if any). For example, in Figure 2, the *national insurance* table (①) is represented as the left operand (②) for the join operation represented in node ③.

Understand a mapping. For understanding mappings, the user can interact with the *mapping explanation tree* and drill down on the evidence that informed its creation. The interaction is done by clicking on the three types of nodes, i.e., *leaf*, *L/R*, and *operation* nodes, each revealing evidence for that merge step:
Leaf node. Clicking on a leaf node leads to a snippet of the initial source that it represents. This helps the user understand the dataset information and whether it was semantically correct to merge it with the other sources in the mapping. This helps to decide whether to keep or discard a source from the search. For instance, after clicking on *national insurance* (①) a snippet of the source appears (❺). It can be observed from it that the source contains data about national insurance registrations, which is relevant to the chosen target that requires employment data.
L/R node. Clicking on a *L/R* node shows information about each attribute of that intermediate mapping: *(i)* the target attribute

that it matches (if any), and *(ii)* the provenance of the distinct values in the attribute. The information about the value provenance is shown as set operations based on how the parent attributes were merged, e.g., the result of joined attributes contains the intersection of their values. This helps to understand if a match is inappropriate or if source attributes were incorrectly merged. In Figure 2, to understand how the *national insurance* source contributes to the target, the user needs to click on its corresponding *L* node (②), for which the information about the attributes appears (❹) so they learn that this source contributes with both *Area* and *NIN registration* values to the target (highlighted in blue in ❹). Node ④ is another example of *L/R* node that corresponds to the result of the join on *Area* between *national insurance* and *indices of deprivation* datasets (③). The attribute information for this intermediate mapping shows that its *Area* attribute contains the intersection of the values from *Area* attributes in *national insurance* and *indices of deprivation*, while the rest of the attributes are not merged.

Operation node. Clicking on an *operation* node shows the profiling data that was used to choose a certain merge operation. The information shown contains a table listing inclusion dependencies and another table for candidate keys. This helps decide whether the used profiling data is meaningful. In Figure 2, to understand how the *national insurance* dataset is merged with *indices of deprivation*, the user accesses the information about their merge by clicking on the corresponding *operation* node (③). This shows the profiling data that was used to decide their join on *Area* (❸), which comprises two candidate keys on *Area* attributes in both sources that share one full inclusion dependency.

Steer mapping generation. Understanding the generated mappings can be crucial in the selection of proper mappings to populate a target. Thus, a user can understand if the chosen sources should be merged or not, or if Dynamap was misled by any faulty matches or profiling data. After the user explores the mapping explanation and pins down any wrong decisions, they can alter through the interface the input that previously led to incorrect merges. The steering can be done through *(i)* adding/deleting source-to-target matches, e.g., remove those between a source attribute that represents a different concept than the one required by the target, *(ii)* removing misleading inclusion dependencies, e.g., remove those between semantically different source attributes that have common values, *(iii)* removing candidate keys on attributes which should not be keys, but were detected due to the (possibly) scarce data in the source, or *(iv)* removing unnecessary datasets from the search space. For example, in Figure 2, the merge opportunities that Dynamap finds seem correct as all joined attributes are suitable pairs in terms of their meaning and value overlap. However, the snippet of the target (❶) shows that many tuples have the same attribute values, for which only the *Population* values differ, indicating there might be an incorrect join with the source(s) that contributes to the *Population* attribute. By accessing the attribute information (❼) corresponding to the last merge in the mapping (⑤) it is shown that the value provenance of *Population* is from the *housing density* source. The next step is to analyze the data in the source by clicking on the corresponding leaf node (⑥) for which a snippet of its extent (❻) is shown. Analyzing the dataset fragment, it can be observed that the match is incorrect although the *Population* attribute in *housing density* has exactly the same name as the target, which makes it seem a correct match. The two attributes represent different types of population: in the source, the population information is per ward, not per area as required by the target. Thus, it can be concluded that the match is inaccurate. In such situations, it is not straightforward to differentiate between two (possibly close) semantic meanings, especially when the attributes contain numerical values. Learning about this incorrect match, the user steers the mapping generation process by discarding the *Population* match. After a new rerun of the mapping generation process with the updated input, the new resulting mappings produce sensible results in terms of aligned data per area. However, given that the *housing density* source does not contribute anything else to the target, it is automatically eliminated from the search space, thus, the new mappings do not involve it.

5 CONCLUSIONS

We have illustrated mapping generation in the wild with open data. Not only do we produce useful mappings, but also the system is transparent – the user can see what has been done, and steer the production of future mappings by altering and/or correcting the three types of input: inaccurate matches, misleading profiling data (candidate keys and inclusion dependencies), and unnecessary sources.

The problem of automated mapping generation might seem as an overly-engineered approach when the number of input sources is small and an expert user can hand-craft the mappings. However, the problem becomes increasingly complicated when tens or hundreds of sources are involved and they have a plethora of merge opportunities. Thus, automating mapping generation becomes an essential component for data integration. Moreover, given autonomous input sources, finding correct mappings completely automatically might become an unfeasible task. This gives rise to the problem of understanding complex mappings that involve numerous, autonomous sources. Helping the user understand the mappings leads to steering the mapping generation process by providing proper input, based on which the system can make correct decisions.

Acknowledgements. We are pleased to acknowledge the support of the UK Engineering and Physical Sciences Research Council, through the VADA Programme Grant (EP/M025268/1).

REFERENCES

[1] Z. Abedjan, L. Golab, F. Naumann, and T. Papenbrock. 2018. *Data Profiling.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00878ED1V01Y201810DTM052

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

[3] J. Attard, F. Orlandi, S. Scerri, and S. Auer. 2015. A systematic review of open government data initiatives. *Government Information Quarterly* 32, 4 (2015), 399–418. https://doi.org/10.1016/j.giq.2015.07.006

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

[5] R. Fagin, L. M. Haas, M. Hernandez, R. J. Miller, L. Popa, and Y. Velegrakis. 2009. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications.* LNCS, Vol. 5600. Springer Berlin Heidelberg, 198–236. https://doi.org/10.1007/978-3-642-02463-4_12

[6] G. H. L. Fletcher and C. M. Wyss. 2006. Data Mapping as Search. In *Advances in Database Technology - EDBT.* 95–111. https://doi.org/10.1007/11687238_9

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

[8] L. Mazilu, N. W. Paton, A. A. A. Fernandes, and M. Koehler. 2019. Dynamap: Schema Mapping Generation in the Wild. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM.* 37–48. https://doi.org/10.1145/3335783.3335785