Information Systems 38 (2013) 656–687

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Incrementally improving dataspaces based on user feedback☆

Khalid Belhajjame*, Norman W. Paton, Suzanne M. Embury, Alvaro A.A. Fernandes, Cornelia Hedeler

School of Computer Science, University of Manchester, Oxford Road, Manchester, United Kingdom

ARTICLE INFO

Article history:
Received 10 May 2011
Received in revised form
24 January 2013
Accepted 28 January 2013

Available online 9 February 2013

Keywords:
Dataspaces
Pay-as-you-go
User feedback
Data integration
Feedback propagation

ABSTRACT

One aspect of the vision of dataspaces has been articulated as providing various benefits of classical data integration with reduced up-front costs. In this paper, we present techniques that aim to support schema mapping specification through interaction with end users in a pay-as-you-go fashion. In particular, we show how schema mappings, that are obtained automatically using existing matching and mapping generation techniques, can be annotated with metrics estimating their fitness to user requirements using feedback on query results obtained from end users.

Using the annotations computed on the basis of user feedback, and given user requirements in terms of precision and recall, we present a method for selecting the set of mappings that produce results meeting the stated requirements. In doing so, we cast mapping selection as an optimization problem. Feedback may reveal that the quality of schema mappings is poor. We show how mapping annotations can be used to support the derivation of better quality mappings from existing mappings through refinement. An evolutionary algorithm is used to efficiently and effectively explore the large space of mappings that can be obtained through refinement.

User feedback can also be used to annotate the results of the queries that the user poses against an integration schema. We show how estimates for precision and recall can be computed for such queries. We also investigate the problem of propagating feedback about the results of (integration) queries down to the mappings used to populate the base relations in the integration schema.

1. Introduction

The problem of data integration has been investigated for the past two decades with the aim of providing end users with integrated access to data sets that reside in multiple sources and are stored using heterogeneous representations [28,35]. The recent increase in the amount of structured data available on the Internet, due in significant measure to the Deep Web [36,29,37], has created new opportunities for using data integration technologies. Yet, in spite of the significant effort devoted to data integration, there seems to have been a limited impact in practice. By and large, data integration solutions are manually coded and tightly bound to specific applications. The limited adoption of data integration technology is partly due to its cost-ineffectiveness [27]. In particular, the specification of schema mappings (by means of which, data structured under the source schemas is transformed into a form that is compatible with the integration schema against which user queries are issued) has proved to be both time and resource consuming, and has been

☆ The work reported in this paper was supported by a grant from the EPSRC.

* Corresponding author. Tel.: +44 7 72 53 69 128.
E-mail addresses: khalidb@cs.man.ac.uk (K. Belhajjame), norm@cs.man.ac.uk (N.W. Paton), smembury@cs.man.ac.uk (S.M. Embury), alvaro@cs.man.ac.uk (A.A.A. Fernandes), chedeler@cs.man.ac.uk (C. Hedeler).

0306-4379/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2013.01.006

recognized as a critical bottleneck to the large scale deployment of data integration systems [27,38,42].

To overcome the above issue, there have been attempts to derive schema mappings from information obtained using schema matching techniques [46,16,40,15]. In their simplest form, matchings are binary relationships, each of which connects an element of a schema, e.g., a relational table in a source schema, to an element that is (predicted to be) semantically equivalent in another schema, e.g., a relational table in the integration schema. Schema matching techniques can be used as a basis for the generation of complex mappings that specify, for example, how the instances of one element of an integration schema can be computed by using the instances of two or more elements in source schemas [44,56].

The mappings that are output by the above methods are based on heuristics. Therefore, many of them may not meet end user needs. Consider, for example, the case of Clio [44]. To specify complex mappings that involve two or more relations in the source schemas, these relations are combined using, for example, a join predicate that capitalizes on referential integrity constraints between the relations in question. While intuitive and useful, this approach does not guarantee that the mapping obtained meets the requirements of end users. The fact that the mapping that meets user requirements may not be generated is not due to faulty behavior of the algorithm implemented by the Clio tool, but because the information provided by the matches used as inputs does not allow the correct mapping to be identified. This raises the question as to how the generated schema mappings can be verified.

A handful of researchers have investigated the problem of schema mapping verification [9,13]. For example, the Spicy system provides functionalities for checking a set of mappings to choose the ones that represent better transformations from a source schema into a target schema [9]. To do this, instance-level data obtained using the mappings under verification are compared with instance-level data of the target schema, which are assumed to be available. The Spider system is another example of a tool for mapping verification [13,4]. Specifically, the tool assists users in debugging schema mapping specifications by computing *routes* that describe the relationship between source and target data.

Using the above tools, the verification of schema mappings takes place before the data integration system is set-up, which may incur a considerable up-front cost [23,27]. In this paper, we explore a different approach in which alternative schema mappings co-exist, and are validated against user requirements in a pay-as-you-go fashion. Instead of verifying schema mappings before they are used, we assume that the data integration system is setup using as input schema mappings that are derived using mapping generation techniques. These mappings are then incrementally annotated with estimates of precision and recall [53] derived on the basis of feedback from end users. Our approach to mapping annotation is consistent with the dataspaces aim of providing the benefits of classical data integration while reducing up-front costs [27]. We do not expect users to be able to (directly) confirm the accuracy of a given mapping nor do we require them to give feedback based on the mapping specification [13]. Instead, the feedback expected from users provides information about the usefulness of the results obtained by evaluating queries posed using the generated mappings. Specifically, given a query that is issued by the user against the integration schema, a.k.a. global schema, it is reformulated in terms of the sources using the candidate mappings that express the elements of the integration schema in terms of the sources. Note that the reformulation phase may yield multiple queries, since the integration elements are likely to be associated with more than one candidate mapping. Reformulated queries are then evaluated by querying the sources. The user can then provide feedback by commenting on the tuples returned as a result of evaluating reformulated queries. Specifically, a feedback instance provided by the user specifies if a given tuple is expected or unexpected[1].

Given the feedback instances provided by the user, we then annotate the mappings. Specifically, we estimate the precision and recall of the mappings, given the results they return, based on the feedback supplied by the user. For example, consider a mapping m that is a candidate for populating a relation r in the integration schema. Based on user feedback that picks tuples that belong to r and tuples that do not, we estimate the precision and recall of the results retrieved using m. They are no more than estimates because we do not assume the user has complete knowledge of the correct extent to be returned and, therefore, do not ask the user to judge every tuple returned. In this paper, we report on an evaluation of the quality of the resulting mapping annotations for different quantities of user feedback. The feedback specified by users may be inconsistent with their expectations. For example, a user may mistakenly tag an expected tuple as a false positive. We investigate that the impact inconsistent feedback may have on the quality of the computed mapping annotations.

Individual elements of the integration schema will frequently be associated with many candidate mappings. We consider a setting in which the candidate mappings are generated based on a large number of matches obtained using multiple matching mechanisms. Therefore, evaluating a user query using all candidate mappings incurs a risk of dramatically increasing the query processing time, and of obtaining a large collection of results, the majority of which are unlikely to meet user needs. We present a method that, given user feedback and user requirements in terms of precision and recall, selects the set of mappings that are likely to meet the stated requirements. Specifically, this method casts the problem of mapping selection as a constrained optimization problem, i.e., that of identifying the subset of the candidate mappings that maximize the recall (resp. precision) given a minimum threshold for the precision (resp. recall).

[1] As we shall see later in Section 2, a feedback instance can also be used to specify that a given attribute value, or combination of attribute values, is expected or unexpected. That said, in this paper, we mainly consider tuple-based feedback instances that comment on given tuples.