**ABSTRACT**

Recently there has been a rapid increase in the number of data sources and publicly accessible data services, such as cloud-based data markets and data portals, that facilitate the collection, publishing and trading of data. Data sources typically exhibit large heterogeneity in the type and quality of data they provide. Unfortunately, when the number of data sources is large, it is for users to reason about the actual usefulness of sources and the trade-offs between the benefits and costs of acquiring and integrating sources. In this demonstration we present SOURCEsIGHT, a system that allows users to explore a large number of heterogeneous data sources, and discover valuable sets of sources for diverse integration tasks. SOURCEsIGHT uses a novel multi-level source quality index that enables effective source selection at different granularity levels, and introduces a collection of new techniques to discover and evaluate relevant sources for integration.

1. INTRODUCTION

Data integration remains among the most cost-intensive tasks in data management, because of the considerable computation and human-curation costs involved in the process [13], and, often times, the monetary cost involved in acquiring data [1]. With a rapid increase in the number of data sources available for consumption, facilitated in part by the rise of data markets [1] and public dataset repositories [2], an additional challenge faced by a user is to identify the sources that are truly valuable and appropriate for her application. This gives rise to the natural question of how can one discover the most valuable sources for integration, i.e., sources that maximize the user’s benefit (i.e., the utility of the data in the final integration result) at the minimum cost.

Several approaches have been proposed to help users reason about the value of integrating multiple data sources. The proposed approaches can be divided into two main categories: (i) *cost-oriented* techniques that focus on characterizing solely the cost of integration, and (ii) *value-oriented* techniques that reason about the trade-off between the benefit and cost of integration. Techniques from the first category estimate the cost of integration via reasoning about the effort required to perform schema-matching, data-cleaning and data-transformation when integrating multiple sources [13, 14]; however, these techniques do not reason about the actual benefit of integrating multiple sources [10]. Value-oriented techniques from prior work focus on characterizing the marginal benefit of integrating a new source [6, 11]. Nevertheless, prior work typically assumes that users have already identified all sources relevant to their application and does not allow users to explore and identify which sources are the most valuable for their desired integration task, nor do they support diverse integration tasks across multiple users.

To understand these limitations, consider a biologist that wants to combine diverse sources of information from functional genomics experiments to make large-scale predictions for protein-protein interaction networks. Each prediction task is associated with an *integration task* that is characterized by the organism and the genes the biologist is targeting. Results of such experiments are usually collected in repositories, such as ArrayExpress\(^1\), that contain vast amounts of experimental data (e.g., ArrayExpress has results from 56,369 different experiments). Due to the large number of sources it is unlikely for users to know in advance all sources that are relevant to their task. Thus, it is essential to allow multiple users to discover and select sources based purely based on their integration task description. For example, one group of users might be interested in human protein-to-protein interaction networks [15] conducted using a specific experimental technique and others in yeast protein-to-protein networks [12].

Recently, we introduced our vision for a *data source management system* [10] that enables users to discover the most valuable sources for their applications. In this demonstration proposal, we present SOURCEsIGHT, a fully functional prototype of such a data source management system. We also discuss extensions to our initial design that allow a user to evaluate the effectiveness of the system (Section 3.5) at selecting sources automatically. The core of the system is built around the paradigm of *source selection* [6] that allows one to reason about the trade-offs between the benefit and cost of integration. The benefit of integration can be quantified using a variety of data quality metrics, such as coverage, accuracy, timeliness and bias [10]. The cost of integration is quantified using a model similar to that of Dong et al. [6]; however, other cost models such as the one introduced by Kruse et al. [13] can be seamlessly incorporated in our system. SOURCEsIGHT offers a number of unique features:

1. Users can describe the domain of their desired integration task using a keyword-based interface and explore sources as well as similar integration tasks that are highly relevant to their desired task.

2. Given an integration task, users can perform source selection using a variety of quality metrics. SOURCEsIGHT casts source selection as a multivariate optimization problem to help users understand the actual trade-offs between different quality metrics and proposes different sets of sources to be used when different weights are assigned to different quality metrics.

3. Finally, the system allows users to interactively edit the recommended solutions by adding or deleting sources. Users can also perform a qualitative comparison between different sets of sources. This is crucial for evaluating the solutions recommended

\(^1\)http://www.ebi.ac.uk/arrayexpress/
by SOURCEVISION and aids the users to understand why a particular set of sources was proposed by the system.

Related work. Over the past few years, the research community has introduced a number of techniques to support search over large numbers of datasets in the form of Web tables [9, 4, 16]. Unlike SOURCEVISION, these techniques focus on providing rankings of datasets and are agnostic to the cost or benefit of integrating multiple datasets. Apart from Web table search, more generic data search systems, such as Microsoft’s Power BI, have been recently proposed. Nonetheless, while those systems focus on facilitating data integration, they do not provide analysts with the functionality to understand the quality of data sources.

2. DISCOVERING VALUABLE SOURCES

Consider a user who wants to find sources that are relevant to a desired integration task. Such users can be biologists using ArrayExpress, data scientists exploring datasets in a repository such as DataHub [2] or journalists who want to compile information from multiple news sources stored in a repository such as EventRegistry3. While users can provide a keyword-based description of their task, they may not know all sources relevant to it in advance. Hence, our goal is two-fold:

First, users should be able to provide a keyword-based description of their task and discover sources relevant to it. Further, it would be beneficial to explore related and more specialized task descriptions as the set of relevant sources may change significantly. SOURCEVISION is the first system that unifies these two aspects of source exploration into a common interface. Figure 1 shows an example use-case where a journalist wants to write an overview article about the socio-economic situation in Greece. The journalist starts by requesting news sources relevant to the keyword “Greece”. Apart from presenting the relevant sources, SOURCEVISION additionally recommends that it might be beneficial to explore related and more specialized descriptions, such as “Greece and Business and Finance” or “Greece and Labor”, as the set of relevant sources may change significantly. Based on these recommendations the user can revise her integration task description and obtain a more focused set of relevant sources.

Second, sources discovered in the previous step may exhibit large heterogeneity with respect to the data they provide. They may provide stale or erroneous data [5, 8], they may contain duplicate data [3, 8] at different costs, and may exhibit schema or instance heterogeneity. Given this diversity in the sources, the users need to be able to reason about the utility of integrating together a set of sources; specifically, a user should be able to find a set of sources that satisfy any cost constraints that she has, and that, if integrated together, will maximize the benefit of integration.

To support this, SOURCEVISION allows users to perform source selection [6], i.e., choose a subset of sources among a set of relevant sources \( S \) identified using keyword search and exploration. Let \( B(S, I) \) denote the benefit of integrating sources \( S \) with respect to an integration task \( I \), and let \( C(S) \) denote the total integration cost for \( S \). Source selection finds a set of sources \( S_I \) such that

\[
S_I = \arg \max_{S \subseteq S} B(S, I) - C(S).
\]

To quantify the benefit of integration, SOURCEVISION uses a range of rigorous data quality metrics, such as coverage, accuracy, timeliness and bias [10].

Given this variety of quality metrics, SOURCEVISION provides the user with multiple source selection solutions that correspond to different weighting configurations for the available quality metrics. This allows users to explore different trade-offs amongst the available quality metrics and identify the set of sources that best satisfies their quality requirements (see Figure 2).

3. SOURCEVISION DESIGN

Recently, we put forward our vision about the functionalities that data source management systems should support to facilitate effective source selection and the architecture they should follow [10]. SOURCEVISION is a realization of such a system. Next, we present SOURCEVISION’s architecture, how it implements (Sections 3.3 and 3.4) the functionalities described in [10], and new functionalities (Section 3.5) SOURCEVISION provides.

3.1 Architecture overview

SOURCEVISION is built as a layer on top of a data source repository that stores the raw data of data sources. The system consists of three components: a frontend, a source analysis engine and an exploration engine (Figure 3). The basic operations of SOURCEVISION can be divided into an offline phase and an online phase. In the offline phase, the source analysis module constructs an index describing both the content and the quality of each source with

4http://eventregistry.org/
respect to multiple quality metrics. This index is used during the online phase when a user interacts with the system to serve the user requests. We discuss the indexing technique in more detail in Section 3.2. In the online phase, users interact with SOURCEIGHT via its frontend module and user requests are served by the exploration engine. Users can specify a desired integration task by providing a keyword-based description of their integration task. Once a description is provided, users can choose among three main functionalities with respect to the specified integration task. They can (i) choose to explore which keywords and sources are most relevant to their task, (ii) choose to perform source selection, or (iii) choose to perform a qualitative comparison between different sets of sources constructed either manually or automatically via source selection.

### 3.2 Reasoning about the content of sources

We assume that each entry in a source (e.g., a tuple in a table or a news article in a news media source) is associated with a set of context literals and all context literals come from a literal dictionary \( V \). We assume that \( V \) is hierarchically structured and edges between literals can encode containment, equivalence and other semantic relations. For example, \( V \) can be a knowledge base with literals corresponding to real-world entities and concepts and its structure corresponding to the edges connecting these literals. Figure 4 shows an example knowledge base with concept literals being hierarchically structured (e.g., “Country” is subsumed by “Location”) and entity literals being semantically associated with concept literals (e.g., “USA” has a specific “Population”). The dictionary \( V \) allows one to identify the domains covered by each source by analyzing the union of context-literal sets for the entries of the source.

![Figure 4: An example knowledge base with the corresponding correspondence graph.](image)

To reason about the content and quality of different sources we augment dictionary \( V \) with a correspondence graph [10]. An example of a correspondence graph is shown in Figure 4. The nodes in the correspondence graph are either data sources (source nodes) or clusters of literals as dictated by the available sources (c-cluster nodes). The edges in the correspondence graph connect each source node with c-cluster nodes and c-cluster nodes with the corresponding literals in the knowledge base. In the example above, there are two c-cluster nodes, one corresponding to the population of counties in Asia and one to sports in the USA (i.e., “USA” and “Sports”). The edges connecting c-cluster nodes to literals follow conjunctive semantics. Each edge from a source to a c-cluster node is annotated with a quality profile of that source for that specific c-cluster, and each c-cluster node is associated with local information about the dependencies of the data sources that are connected to it.

The correspondence graph serves as a content and quality index for the available sources. To construct it we first learn the latent c-cluster nodes and then compute the quality profiles and data source dependencies for each c-cluster node. To discover the literals associated with each source entry we use Thomson Reuter’s Open Calais\(^4\), an API for semantic annotations with respect to multiple knowledge bases including DBpedia, Freebase and others. To construct the c-cluster nodes we adopt a frequent pattern mining approach based on the FP-growth algorithm [7]. This allows us to discover domains that are prevalent in multiple sources. After discovering the c-cluster nodes, we compute the quality of each source with respect to each c-cluster node it is connected to. To do this, we collectively analyze the content of all sources connected to a c-cluster node to form a single dataset characterizing the content of the c-cluster and then each individual source is compared with the combined data to compute its quality [10].

### 3.3 Specifying an integration task

SOURCEIGHT allows users to specify an integration task \( I = (I_d, I_c) \) by providing a description that corresponds to a context-literal set \( I_d \) defining the domain of the task and potentially a set of integration cost constraints \( I_c \). The system recommends literals and sources that are related to their keyword search, enabling them to either explore sources relevant to their search or refine their initial task description by considering related literals.

Relevant context-literals are all literals that are related to literals in \( I_d \) either via \( V \) or via the correspondence graph. The former accounts for semantic relations across literals while the second accounts for co-occurrence of literals in a source. Similarly, relevant sources to task \( I \) are all data sources that provide entries whose context-literal set is a subset of \( I_d \) or related to \( I_d \) via the literal dictionary \( V \) (e.g., using equivalence or containment relationships).

SOURCEIGHT offers a unified interface for users to explore both context-literals and sources related to their desired integration task (see Figure 1). Given an integration task description \( I_d \), SOURCEIGHT returns the set of top relevant literals to the search of the user as well as the most relevant sources with respect to coverage or other metrics for the specified keyword search. The user can then select any of the recommended sources to view a summary of the literals that the source covers as well as a quality summary of the source for the corresponding keyword search. Users can also choose to update their integration task description \( I_d \) by including new relevant context-literals to \( I_d \).

### 3.4 Multivariate source selection

Given an integration task \( I = (I_d, I_c) \), the user can perform source selection with respect to the context-literals \( I_d \) and the constraints \( I_c \). As discussed in Section 2, SOURCEIGHT considers multiple quality metrics to quantify the benefit of integration. The system casts source selection as a multi-objective optimization problem and finds the set of Pareto optimal solutions corresponding to the source selection problem at hand.

Discovering all the solutions on the Pareto front is expensive as one needs to reason about all the potential trade-offs amongst the available quality metrics. To address this issue we use a sampling strategy to recover solutions that correspond to different quality trade-offs. Let \( Q \) be the set of quality metrics under consideration and \( B_q(\cdot) \) be an oracle computing the benefit of integration with respect to quality metric \( q \in Q \) for any set of sources. We compute the total benefit of integration as a weighted linear combination of the individual benefit for each quality metric, i.e., \( B(\cdot) = \sum_{q \in Q} w_q \cdot B_q(\cdot) \). Given this definition of the total benefit

\(^4\)http://www.opencalais.com/
of integration, we sample different combinations for the weights $w_q$ and solve source selection for each of those. Finally, we identify the Pareto optimal solutions amongst the sampled solutions.

All the sampled solutions are presented to the user in a way that makes it easy for her to compare the quality of each solution. The corresponding interface is shown in Figure 2. Users can select a particular solution and view a concise summary of the benefit and cost of integration achieved by it. Users can also drill down and expand only on a subset of the available quality metrics to fully understand specific trade-offs across different solutions. Finally, users can view a detailed description of a source selection solution with information about the sources included in the result and their individual contributions to the quality of the final integration result.

### 3.5 Comparing sets of sources

SOURCEIGHT enables users to understand why a particular set of sources was recommended to them and evaluate the its performance. Users can perform a qualitative comparison between two sets of sources with respect to the same integration task (Figure 5). They are also able to examine how two sets of sources compare against each other with respect to individual quality metrics, as well as the total integration benefit and cost. The sets of sources to be compared can correspond to recommended solutions from source selection or can be manually constructed by the user. SOURCEIGHT offers two ways of manually specifying sets of sources: (i) users can start from a returned source selection solution and add new sources or remove already included sources, or (ii) users can manually select different sources and construct their own set. The former allows them to examine the neighborhood of the returned solution but also serves as a mechanism of convincing the user of the value achieved by the output of source selection. To enable the latter, SOURCEIGHT provides users with the top-$k$ most relevant sources for their integration task with respect to each individual quality metric. As users select sources, these rankings are revised to contain sources that are valuable given the current selection of the user. Rankings are computed with respect to the individual additional value of each source.

### 4. DEMO DETAILS

We will demonstrate the functionality of SOURCEIGHT through hands-on experience with two real-world data repositories. With the data stored on a remote server, users will interact with the system via a web interface. Our goals are two fold: (i) demonstrate the utility of SOURCEIGHT in exploring and selecting beneficial sources for integration, and (ii) demonstrate the effectiveness of automatic source selection for diverse integration tasks.

**Datasets.** The first dataset corresponds to event extractions collected from EventRegistry, a repository that monitors news media from all over the world. Data sources correspond to news domains and quality metrics such as coverage, timeliness or position bias of the articles are inherently relevant. EventRegistry gets updated at fixed intervals by ingesting feeds of newly extracted news articles. EventRegistry is the right fit to demonstrate the usefulness and practicality of SOURCEIGHT due to the large number of data sources available, the available updates over time, and the heterogeneity that the sources exhibit both with respect to their domains and their quality. We plan to use a recent snapshot retrieved from EventRegistry containing at least six months of news article data.

The second dataset corresponds to a Twitter snapshot where Twitter users are viewed as sources. Twitter gives its users an unprecedented ability to deliver news as developments unfold and has been adopted by many researchers as the means of predicting occurrences of events in the future. We believe that analyzing collections of tweets via SOURCEIGHT to identify sets of influential twitter users by employing source selection techniques would be of great interest to the VLDB participants.

**Demonstrating Utility.** Attendees will be able to describe ad-hoc or pre-formulated integration tasks in SOURCEIGHT. Then they will have the opportunity to evaluate the source exploration and source selection functionalities offered by SOURCEIGHT. Our goal is for users to understand the trade-offs between different quality metrics of sources in both datasets we have described above and understand how SOURCEIGHT can guide them to select the most suitable set of sources for their application needs.

**Demonstrating Effectiveness.** For this part of the demo users will mainly interact with the third functionality of SOURCEIGHT, i.e., comparing and contrasting sets of sources. Attendees will compare sets of sources provided by source selection with manually created sets of sources and understand the quality differences across them.

### 5. REFERENCES