Dieses Dokument ist eine Zweitveröffentlichung (Postprint) /

This is a self-archiving document (accepted version):

Katrin Braunschweig, Julian Eberius, Maik Thiele, Wolfgang Lehner

OPEN—Enabling Non-expert Users to Extract, Integrate, and Analyze Open Data

Erstveröffentlichung in / First published in:

Datenbank-Spektrum. 2012, 12 (2), S. 121–130 [Zugriff am: 27.01.2023]. Springer. ISSN 1610-1995.

DOI: http://dx.doi.org/10.1007/s13222-012-0091-9

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-831131

SLUB
Wir führen Wissen.

TECHNISCHE UNIVERSITÄT DRESDEN

QUCOSA
Quality Content of Saxony

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

OPEN—Enabling Non-expert Users to Extract, Integrate, and Analyze Open Data

Katrin Braunschweig · Julian Eberius · Maik Thiele · Wolfgang Lehner

Abstract Government initiatives for more transparency and participation have lead to an increasing amount of structured data on the web in recent years. Many of these datasets have great potential. For example, a situational analysis and meaningful visualization of the data can assist in pointing out social or economic issues and raising people's awareness. Unfortunately, the ad-hoc analysis of this so-called *Open Data* can prove very complex and time-consuming, partly due to a lack of efficient system support.

On the one hand, search functionality is required to identify relevant datasets. Common document retrieval techniques used in web search, however, are not optimized for Open Data and do not address the semantic ambiguity inherent in it. On the other hand, semantic integration is necessary to perform analysis tasks across multiple datasets. To do so in an ad-hoc fashion, however, requires more flexibility and easier integration than most data integration systems provide. It is apparent that an optimal management system for Open Data must combine aspects from both classic approaches.

In this article, we propose *OPEN*, a novel concept for the management and situational analysis of *Open Data* within a single system. In our approach, we extend a classic database management system, adding support for the identification and dynamic integration of public datasets. As most web users lack the experience and training required to formulate structured queries in a DBMS, we add support for non-expert users to our system, for example though keyword queries. Furthermore, we address the challenge of indexing Open Data.

K. Braunschweig · J. Eberius · M. Thiele (✉) · W. Lehner
Technische Universität Dresden, Nöthnitzer Str. 46, 01187 Dresden, Germany
e-mail: maik.thiele@tu-dresden.de

K. Braunschweig
e-mail: katrin.braunschweig@tu-dresden.de

J. Eberius
e-mail: julian.eberius@tu-dresden.de

W. Lehner
e-mail: wolfgang.lehner@tu-dresden.de

1 Introduction—Open Data on the Web

The World Wide Web is a seemingly unlimited source for data. Especially the amount of regularly structured data has increased noticeably in recent years. Data published by public or government agencies as part of their Open Data Initiatives accounts for a significant percentage of this data. For instance, the UK government opens up data documenting public spending in the country, the World Bank publishes global data on urban and social development and the United Nations provide data on world health statistics compiled by the WHO. In general, all data that is publicly available under a license that enables reuse without fees or restrictions can be regarded as Open Data. The motivation behind making the data available on the web ranges from ensuring transparency and accountability to encouraging innovation and economic growth as well as educating and influencing people. A prominent application example that clearly benefits from this data is Data-Driven Journalism. In this emerging field, journalists combine and analyze large amounts of public data to either (a) identify facts that support their story or (b) uncover previously undisclosed links or patterns in the data that point to issues in our society. One organization that uses this approach to support their cause is, for example, Transparency International.[1]

[1] www.transparency.org.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

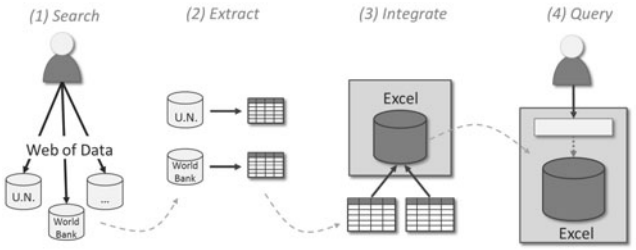

Fig. 1 Manual integration process

Discovering this data and making the information more accessible greatly supports the goal of informing or educating people. To extract the information, data journalists need tools that offer the functionality to combine heterogeneous datasets provided by different agencies and perform ad-hoc data analysis.

To get an idea of the processing steps involved, imagine the following scenario:

> *In light of the financial crisis and the seemingly increasing number of natural disasters like strong earthquakes or tsunamis happening around the world, a data journalists wants to investigate the effects these natural disasters have on the economic and financial strength of a country.*

Without system support, the manual approach to gather and process the necessary data looks as follows (see also Fig. 1):

- *Identifying data sets*: The data is often distributed across several independent platforms or sites on the web and so far, no central access point to search for Open Data exists. This means that several platforms have to be searched individually. Some regular search engines, such as Google, have started indexing public datasets, but cover only a small part of the available data and provide no direct download. Alternatively, most of the dedicated platforms support search functionalities of varying degrees. In our example, the journalist could search the catalog of data published by the United Nations for statistics on natural disasters worldwide as well as the World Bank data for economic indicators such as the GDP.
- *Extracting the data*: After suitable datasets have been identified on one or more platforms, the data must be extracted or downloaded for further processing. Most platforms directly offer download links for their data files. Some platforms also provide tools to export data in different file formats. While these approaches are straightforward, some other platforms make it more difficult to extract the data. By embedding the data into the websites, these platforms prevent direct reuse of the data. Instead, web scraping tools are required to extract the required information.
- *Integrating the data*: The third step that is necessary before the data can be analyzed is the integration of the heterogeneous datasets into a consistent representation. Both, structural as well as semantic integration are required to resolve the ambiguity and enable joint processing of the data. In our scenario, assuming that the journalist downloaded one dataset containing the number of natural disasters per country (per month and year) and another containing the GDP per country (per month and year), a link needs to be established between the country entries in both datasets.
- *Analyzing the data*: Finally, classic BI or spreadsheet tools (e.g. Excel) as well as modern web based data analysis tools (e.g. Google Fusion Tables) can be applied to analyze and visualize the data. While classic BI tools provide all the functionality to process and analyze the data, these tools are often too complex for the average non-expert user. In contrast, web-based data analysis tools address non-expert users in particular, but often lack some of the analysis functionality of classic tools. The journalist in our example could now use Excel, for example, to analyze the differences in GDP in the time before and after a natural disaster in countries that were affected. Further, he could compare the results to those for countries that were not directly affected and visualize the result in various plots.

The overall process of analyzing heterogeneous, distributed data sources is complex and time-consuming, as it involves the use of several different tools. Additionally, professional journalists, for instance, often do not have the training or expertise required to use complex data integration or BI tools. The same goes for many other web users. It is apparent that the lack of support by a single system or tool clearly hinders the use of public data for tasks such as data-driven journalism. We want to address this issue by proposing a system that supports ad-hoc analysis of Open Data by non-expert users. In contrast to other resources of data on the web, Open Data has a number of distinctive characteristics we can utilize to provide better system support. These characteristics include the mostly regular structure of the datasets as well as the rich set of metadata used to describe the content of the datasets. Furthermore, the datasets are in general published on designated Open Data platforms, which reduces the scope of the search space on the web significantly.

Several researchers have addressed similar or closely related data management applications. In the following section, we take a look at some of these applications and show how our scenario fits into this context.

In the remaining sections of this article our contributions are as follows:

1. Identification of characteristic dimensions for the classification of Data Management Systems.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

2. Proposition of *OPEN*—the Open Data Processing Engine, a novel approach to provide system support for the advanced, ad-hoc analysis of Open Data.
3. Review and classification of related work in the field, with respect to the previously defined classification dimensions.

2 Open Data in the Context of Data Management

The domain of Data Management Systems can be characterized using many different dimensions. Two of these dimensions are depicted in Fig. 2, which represents an extension to the original figure presented in [9]. The first dimension on the vertical axis represents the *administrative proximity*. Near indicates that the data sources that are to be managed are under the same or coordinated control, while far indicates that there is only little or no coordination. Higher coordination allows stronger guarantees about consistency or persistency, for example. The second dimension on the horizontal axis is labeled with *semantic integration*, which describes how well the schemas of individual sources have been matched. High integration corresponds to all schemas being matched to a single, global schema, while low integration corresponds to no schema at all.

The figure shows two classic data management approaches located on opposite sides of the spectrum, namely Information Integration Systems and Information Retrieval Systems (referenced as Web Search in the figure). In order to identify further relevant dimensions in this domain, we first characterize and compare these two systems, which serve as some sort of boundaries.

2.1 Information Integration Systems

Information Integration Systems (IIS) are designed to provide a unified view over multiple heterogeneous data sources. Before any data can be queried, a single global schema must be designed, with mappings from local schemas to the global schema. This means a large, often manual effort during the design-time of the system. The use of a global schema resolves the semantic ambiguity inherent in independently compiled data sources and improves the precision of query results. Furthermore, the structure of the data is defined by the schema. With this information, users can specify a wide range of complex, structured queries. It also enables the specification of constraints to ensure the consistency and correctness of the data.

However, due to the rigid global schema, the system is inflexible regarding structural changes and extensions. Especially adding new data sources to an existing global schema can prove very complex and time consuming. This issue is highlighted, for instance, in enterprises where data analysts have a tendency to create so-called spreadmarts to avoid the computational costs of complex integration tasks. In addition, the correct formulation of complex, structured queries requires extra knowledge and training, which can be a barrier for novel or inexperienced users and clearly limits the usability of the system. Overall, Information Integration Systems offer precise, complex query functionality over multiple data sources at the cost of significant design-time integration effort.

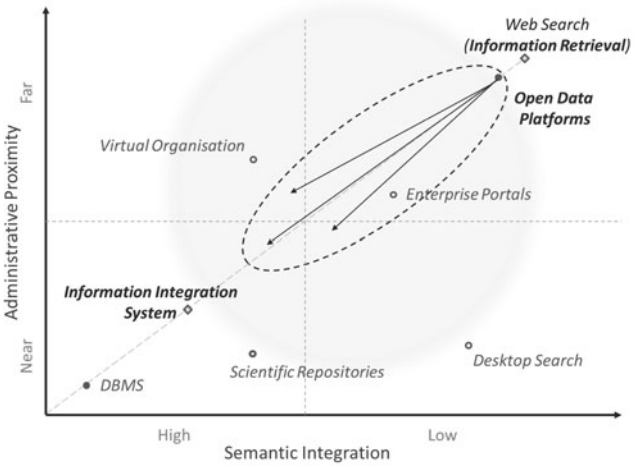

Fig. 2 The domain of data management systems and applications

2.2 Information Retrieval Systems

In contrast to integration systems, Information Retrieval (IR) Systems provide basic search functionality over large amounts of distributed, heterogeneous data without structural or semantic integration. Hence, the design-time effort for these systems is significantly lower compared to IIS. As a result of this lack of structural regulations, the system is more flexible regarding structural heterogeneity as well as dynamic extensions.

However, without clear knowledge of the structure of the data, it is very difficult or impossible even, to formulate complex queries. Common query interfaces in IR systems are basic natural language interfaces that only allow the user to post simple keyword queries. Additionally, some XML-based IR systems support simple structured queries. But while natural language interfaces generally require less extra knowledge than structural queries and are therefore easier to use, they are much more affected by semantic ambiguity. Natural language is inherently ambiguous and imprecise query semantics often result in imprecise answers. In summary, Information Retrieval Systems are flexible systems that provide easy access to the data without prior integration costs, but lack in query complexity and precision.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

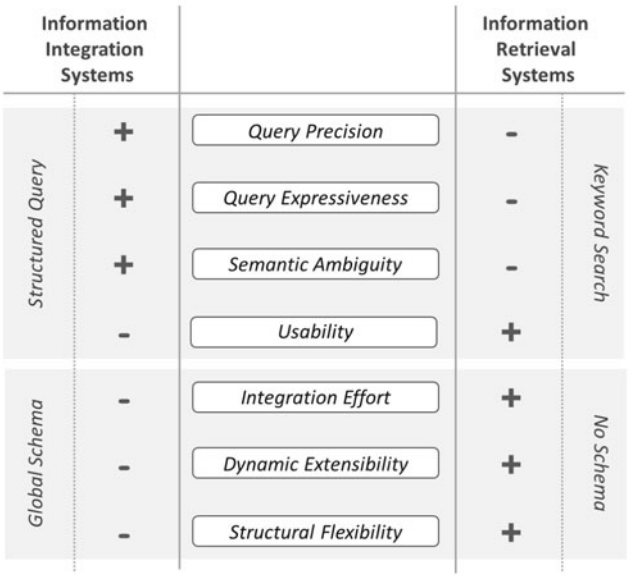

Fig. 3 Information integration systems vs. information retrieval systems

2.3 Characteristic Features and Requirements

From the comparison of Information Integration and Information Retrieval Systems, we derived seven characteristic features to use for the classification of data management systems. A summary of the comparison, including these features, is illustrated in Fig. 3.

Integration Effort refers to the amount of manual or automatic integration work that is necessary before the system becomes operational and its full functionality can be used. *Dynamic Extensibility* describes the ability of a system to require only minimal effort and modifications to add further data sources to the system at runtime. *Semantic Ambiguity* is addressed by semantic integration of the data as well as approaches that resolve semantic ambiguity in the query. The ability of a system to support not only regularly structured data, but also structural irregularities as well as semi-structured or unstructured data is covered by *Structural Flexibility*. *Query Precision* refers to the precision of the query result and *Query Expressiveness* to the diversity of queries that can be expressed through the query interface. Finally, *Usability* primarily refers to the ease of use of the query language.

Using these features, we can now describe the requirements of our application scenario in more detail. To encourage situational processing of Open Data, we need a system that can be extended dynamically with only minimal integration effort. In addition to that, the optimal system must be able to process complex queries with high precision in order to provide advanced analysis functionality. To return high quality results, the system also needs to be able to resolve the semantic ambiguity of the data. As most Open Data users, such as journalists, for example, are no data management experts, the usability of the query interface is also an important factor. The remaining feature, structural flexibility, however, is very important for the processing of web data in general, but less important for the subset of Open Data. We noticed that the majority of valuable information in Open Data can be found in regularly structured datasets stored in CSV or XML files. For our approach, we can capitalize on this characteristic.

Looking at these requirements, we can see that the optimal system for our scenario lies somewhere between Information Integration and Information Retrieval Systems. In [9], Franklin et al. introduced a new abstraction level for this class of data management and integration application, called *Dataspaces*, and the systems which support these application, called *Dataspace Support Platforms (DSSPs)*. The main concept of DSSPs, as introduced by Franklin et al., is to provide basic search functionality without a global schema over all data sources from the beginning and allow the user to incrementally increase the semantic integration between the data sources, whenever more advanced query functionality is required. In Fig. 2, a gray shade outlines the class of dataspace applications. Example application scenarios are, for instance, Personal Information Management, Scientific Data Management and Data Integration on the Web.

The requirements of advanced Open Data analysis are closely related to the more general scenario of web data integration, which addresses the generation of new value by combining data from multiple, previously unconnected datasets on the web. However, as mentioned earlier, a number of characteristic properties set Open Data apart from other types of web data and we can utilize these properties to develop a system optimized for our specific scenario.

The example applications introduced before highlight that there is a wide range of different application scenarios that can be regarded as dataspace applications. We argue that it is unlikely to develop a single system supporting all different types of dataspaces. Instead, the different scenarios will benefit more from a system tailored to their specific requirements.

3 *OPEN*—The Open Data Processing Engine

After identifying the specific requirements of Open Data analysis applications and reviewing related scenarios , we now present, *OPEN*, our novel approach specifically designed to address the issues faced when performing complex ad-hoc analysis tasks on Open Data. To make the description more comprehensible, we also come back to the example introduced at the beginning, regarding a journalists goal to analyze the effects of natural disasters on a country's economic strength.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

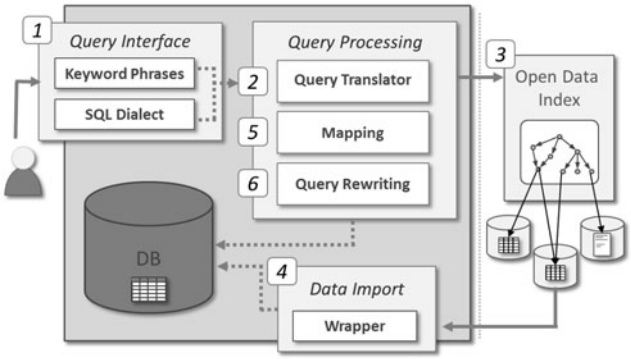

Fig. 4 Integration process with system support

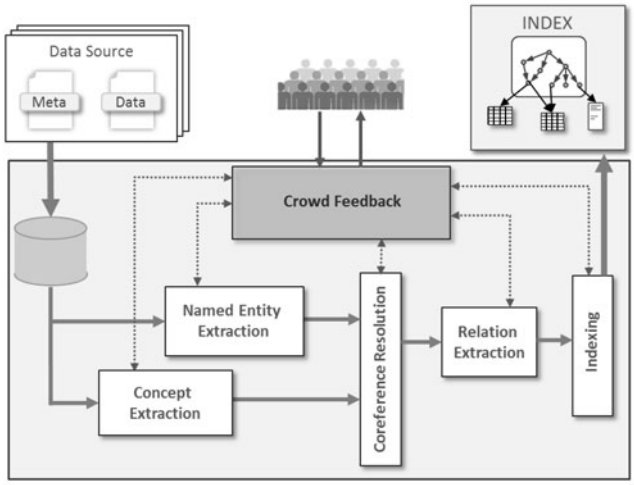

Fig. 5 Information extraction for indexing of open data

To support the user with the integration of heterogeneous, distributed public data, the support platform must be capable of (semi-)automatically searching for datasets according to user requests, fetching suitable datasets, integrating them into the system and restructuring user queries to address the new data.

Figure 4 illustrates our approach. It shows that the *OPEN* system provides a single access point for the user. Instead of having the user interact with several tools, a single system enables the user to query public data. As is also shown in the figure, our system can be regarded as an extension of a classic Database Management System (DBMS), where data is locally stored in a database. There are two main reasons that motivated the selection of this established architecture as the basis for our system: (1) the fact that a majority of the data on Open Data platforms is available in a regularly structured format, and (2) that a DBMS already provides advanced analysis functionality for us to utilize. The remaining extensions to the system include a query translator that distinguishes between local and web queries, a search index optimized for public datasets, a dataset retrieval component that loads public data into the database and a query rewriting component.

To describe the functionality of our system, we distinguish between two separate processing phases. The first phase, which is performed offline before the system becomes operational, is the indexing of public datasets. This preprocessing step is necessary to provide a single access point for the system and enable an automatic and efficient search for datasets. In the second phase, the system can then perform online processing of user queries on the previously indexed data.

3.1 Indexing Open Data

The usability of our approach depends largely on the efficiency and accuracy of the search for datasets. On the one hand, the number of datasets that are considered for a query needs to be reduced significantly. Checking all available datasets is too computationally expensive and obviously not applicable. On the other hand, semantic ambiguity must be addressed during the search to provide more accurate results. Since search terms can be ambiguous, a naive string matching approach would return wrong or unsuitable datasets. The costs of the search can be reduced greatly by using a semantic search index.

As mentioned before, the datasets are originally distributed across a large number of Open Data platforms and there is no single, uniform access point available, yet. Therefore, we must register these platforms manually in our system and provide wrappers to their respective APIs, in order to index and access the datasets. For our running example, we would need the U.N. and World Bank data platforms to be registered in the system.

Semantic Indexing In order to retrieve relevant datasets with high precision it is not sufficient to use an inverted index that only contains keywords that appear in the description or the schema of the datasets, as is common practice on most public data platforms. Instead, we use an index that stores semantic concepts and real world entities, as well as relationships between them. Here we can benefit from another characteristic of Open Data. Most datasets are augmented with a rich set of metadata, such as name of the publisher, a description, tags, categories or geographic coverage, before being published on Open Data platforms. This information describes not only the data but also its context and can be used to identify similar concepts or entities. Figure 5 illustrates our approach.

Information Extraction In order to index datasets based on semantic concepts and entities, this information needs to be extracted from the structured data and, if available, associated semi-structured metadata, first. We apply a number

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

of existing algorithms to automatically extract the information, such as rule-based algorithms for named entity extraction or noun phrase chunking for concept extraction [6]. A survey of further existing information extraction research is presented in [16]. Due to the residual imprecision of the algorithms, we regard the extracted concepts, entities and relationships as hypotheses. To receive a list of distinct concepts and entities for each dataset, we also need to perform coreference resolution and de-duplication. Again, we can adopt existing methods. The goal is to develop an adaptable pipeline that automatically combines the individual processing steps, as shown in Fig. 5.

Without human feedback of some kind, however, high quality results regarding object identification and semantic disambiguation are very hard to achieve. While domain and integration experts would surely be best qualified for this task, they are not always available, as is the case in our example scenario. Semantic integration is inherently a human assisted process and the sheer amount of datasets and the scope of domains on the web make classic human-assisted integration approaches unfeasible. Therefore, we need to find an appropriate substitute. As illustrated in Fig. 5, we utilize crowd feedback to assist in the semantic integration of the data.

Crowd-Sourcing Compared to the regular users of a system or experts assisting in the processing, the crowd usually comprises a much larger number of people with no obvious connection to the application domain or the system. So-called *crowd-sourcing* approaches have been successfully applied to a number of other applications such as query processing [10] and entity linking [7]. The concept of these approaches is that simple, individual tasks can be posted on so-called *Crowd-Sourcing Platforms*, where people can register and solve the tasks for money or other forms of compensation. Common examples of such tasks are describing the content of a given image or evaluating the identity between two given entities. In our case, we ask the crowd to verify the hypotheses generated by the various extraction algorithms, such as the type of named entity or a relationship between two concepts ([8] and [12]) also leverage crowd-sourcing for entity extraction and type verification.

Crowd-sourcing, however, also introduces new challenges. For example, it is impossible to guarantee that all members of the crowd have the knowledge and expertise to give the correct answer to each task. Therefore, further control mechanisms are required to ensure a certain level of quality. One such mechanisms is to introduce redundancy. Having several people solve the exact same task and using the most commonly given answer as the final answer can statistically reduce the level of uncertainty. In [14], Oleson et al. further introduce so-called *gold units* to verify the quality of the answers given by the crowd. Another challenge is identifying the optimal task granularity for the automatic generation of tasks. The same result can be achieved by generating many simples tasks or fewer, but more complex tasks. While fewer tasks require less compensation, the complexity of the task often requires more detailed instructions and can lead to misunderstandings on behalf of the crowd members. In contrast, for simple tasks, it is easier to identify task categories and provide universal phrasing patterns for each category. Therefore, simple tasks are better suited if automatic generation of tasks is necessary.

Optimization Since redirecting tasks to the crowd requires both, time and money, it is not feasible to have every little piece of information verified by the crowd. Therefore, it is important to identify those entities, concepts or relationships that most require verification. We apply several algorithms to automatically verify the extracted hypotheses and use a confidence model to monitor the estimated correctness of the hypotheses. Only hypotheses with a confidence value below a predefined threshold require verification by the crowd.

After all necessary information has been extracted and verified, the datasets from the web are indexed automatically and can be searched for by the user of our system.

The overall process of indexing the public datasets is a preprocessing step that can be performed independent of any application or user requirements. Therefore, we can imagine two different scenarios for the positioning of such an index: (1) as part of the DBMS or (2) as a stand-alone index on the web with search functionality provided via an API.

3.2 Processing User Queries

After the public datasets have been indexed, users can post ad-hoc queries on the web data as well as local data, using the *OPEN* system. The major processing steps are depicted in Fig. 4.

Query Formulation Since our system is an extension of a classic DBMS, the query interface provides support for structured queries by default. To address unexperienced user as well, the interface also provides the option to post semi-structured queries reminiscent of keyword queries instead. Queries can be placed even when no data is in the local database, yet. Recalling the running example, our journalist wants to query public datasets on the web, looking for earthquake statistics per country for February 2011 as well as indicators for economic growth such as the GDP, in the same time frame. Since he does not know whether this information is actually availably on the web, if it is stored in separate datasets or if a combined dataset already exists, he starts with a naive approach and enters all search terms in one query. The first query in Fig. 7 shows a possible, semi-structured query, including search terms for country names,

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

Earthquakes in February 2011		
country name	**quantity**	**...**
Argentina	2	...
Chile	4	...
France	0	...
...	...	...

GDP per capita estimates 2011			
country	**Jan**	**...**	**Dec**
Germany	40,275	...	40,152
France	40,663	...	39,460
USA	45,758	...	47,199
...	...	...	...

Fig. 6 Example open data tables

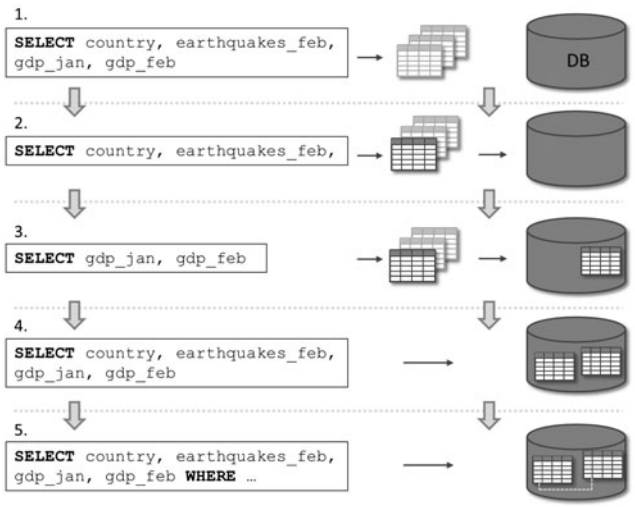

Fig. 7 Example queries and processing steps

earthquakes in February, as well as the GDP in January and February.

Query Translation To support queries on web data as well as local data, modifications to the query translator are required. During the translation process, keywords or query elements are classified as either *known* or *unknown*. *Known* elements are all elements that can automatically be matched to table or column names that already exist in the database. All query elements that can not be matched are regarded as *unknown* elements. If no local tables exist in the database, all elements are regarded as *unknown*. While a classic DBMS requires an exact match between query terms and table or column names, our system follows a fuzzy matching approach.

In Example Query 1, all search terms are identified as *unknown*, since no data has been inserted into the database, yet.

Index Search At this point, we shift from the closed world assumption of a classic DBMS to an open world assumption, by regarding data that is not in the database as *unknown* instead of *non-existent*. We further assume that the missing data does exist somewhere on the web. Therefore, the system automatically generates one or more keyword queries to search for the datasets in the index. Similar to other information retrieval tasks, searching for the dataset that contains the requested information is unlikely to deliver a single, exact match. Instead, the fuzzy search retrieves several matches, which are ranked before a selection of top k results is presented to the user.

Coming back to our example, assuming that no perfect dataset that already combines all the required data exists in the index, datasets matching only some of the search terms are returned for Query 1. This could prompt our journalist to reformulate the query to receive more specific results. As shown in Query 2 in Fig. 7, he directs the search towards earthquake statistics first. This time, the system returns several datasets matching the query, for example a dataset provided by the U.N. or a dataset from the United States Geological Survey.

The final decision, which dataset is to be fetched and integrated into the system, is made by the user. It gives the user a certain amount of control over his data and avoids wrong or unwanted results.

Data Import After a suitable dataset is selected by the user, the complete dataset is fetched automatically from the respective platform and loaded into a new table in the DBMS. Most Open Data platforms provide download links for CSV or XML files containing the data. These links are stored in the index and the files can be retrieved whenever the dataset is requested. After a data file is downloaded, a wrapper for the specific file format is required to load the data from the file and insert it into a new table in the database. If the Open Data platform allowed value level access to the datasets, it would also be possible to extract only the required columns or rows of a potentially much larger datasets. This option would require a query protocol that enables queries on the data via HTTP.

After the data is inserted into the database, the user can now analyze the data or search for further datasets. In our example, the journalist formulates another query, this time searching for economic indicators (see Query 3 in Fig. 7). Again, the most suitable dataset is selected from the top-k matches returned by the system. For the remaining processing steps in the example, we assume the datasets shown in Fig. 6 to be the ones selected by the user.

Mapping After all data is added to the database, the analysis can be performed across multiple tables without further integration by the user. To identify suitable attributes to join

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

the tables, automatic schema and instance matching algorithms are applied. Although some public datasets contain system-generated, local IDs to identify tuples and serve as join attributes, these are not useful in our open world scenario. They are not globally unique and usually not meaningful outside the scope of the original database. Therefore, we need to look for *natural* IDs like names or attributes that uniquely identify objects. For some datasets, a combination of several columns must be used to obtain a unique identifier. Since the matching is a computationally expensive operation, we store the relations between tables to reuse them when necessary and save the cost of recomputing. In our running example, the country names can be used as such an identifier to join the two tables. As shown in Figure 7, this information is stored in the system catalog and can be reused for any further queries that require these two tables to be joined.

Query Rewriting After the import of all datasets and the identification of the join attributes, the original query must be adapted in order to return the correct result. A structured query is generated automatically, referencing the correct columns in the new tables and including the correct join attributes. For the example query, the tables in Figure 6 are joined on the *country* column, which is also returned in the result set. Additionally, the *quantity* column from the first table as well as the *Jan* and *Feb* columns from the second table are selected and returned in the result set.

Through incremental query refinement, the user can now further analyze the data. If it becomes clear during the analysis that more data is needed, additional datasets can be added dynamically. With this functionality the system supports the exploratory nature of real life data analysis. Similar to other dataspace support platforms, integration of the data is carried out in a pay-as-you-go fashion initiated by the user through specific queries.

4 Classification of Related Work

To approach the goal of developing a support platform for data-driven journalism and similar applications based on Open Data, we took a closer look at the individual requirements to identify relevant related work in the field.

An ideal approach would obviously combine the best of both worlds, i.e. the flexibility, extensibility and usability of IR systems as well as the expressiveness and precision of structured queries, but without the integration effort. Such a system, however, is very unlikely, since some of these characteristics are conflicting. A *pay-as-you-go* approach as proposed in [9] is more realistic. Instead of investing in a complete integration before the system can be used, a flexible IR-style system that provides basic search functionality is used as a starting point. If at some point the user requires more sophisticated query functionality, which in turn requires a better integration between the data sources in question, the system enables the user to put in the effort and dynamically add the required information. While [9] introduces the idea behind such an approach, the literature does only provide little information on the implementation of complete dataspace support platforms following this approach. However, a long list of techniques and approaches exist, that address individual aspects that are related to or form part of such a system. Recalling the characteristic dimensions established in Sect. 2 we present a selection of related work that addresses one or more of the features.

First, we take a look at two DSSP approaches proposed in the literature. In [13], Madhavan et al. introduce a new data integration architecture for structured web data called *PayGo*, which follows the pay-as-you-go principle of DSSPs. This architecture mainly focuses on the scale and semantic heterogeneity of structured web data by introducing a repository of schemas. Instead of having to design a single global schema, the repository contains a multitude of smaller schemas clustered into topics. Keyword queries are routed to the relevant sources using approximate semantic mappings. While this approach reduces the effort to achieve semantic integration between web data, it does not provide the query precision and functionality required for our scenario.

A second pay-as-you-go approach is the concept of *iTrails* presented by Vaz Salles et al. [17]. Developed for the context of Personal Information Management, iTrails is used to incrementally add semantic integration to their Personal Dataspace Management System called *iMeMex* [3]. Starting with no schema, mappings between data sources can be added in a declarative fashion via *trails*, which are used to rewrite the user queries to improve the precision. Without further support, however, the definition of these trails is too complex for non-expert users. In a larger application scenario such as the web, manually declaring semantic trails would be unfeasible.

To identify techniques that are better suited for our approach, we also studied several concepts or methods that only address some of the desired features and do not represent complete dataspace support platforms.

At first, we take a look at ontologies, which are formal descriptions of domains using concepts and relationships. In the context of semantic integration, ontologies can be seen as an alternative to a relational schema. Matching heterogeneous data sources to agreed-upon concepts and allowing query functionality only based on these concepts, reduces the semantic ambiguity and increases the precision of query results. However, the effort required to define such an ontology and match data sources to these concepts is similar to the effort required to define a global schema. In [4],

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

Approach	Features						
	Precision	Expressiveness	Ambiguity	Usability	Effort	Extensibility	Flexibility
DSSP - PayGo			x	x	x	x	
DSSP - iMeMex + iTrails	x		x		x	x	x
Ontology	x		x				
Semantic Web / OWL	x	x	x				
Linked Data / RDF			x			x	x
Automatic Schema Matching			x		x		
Keyword Search over Rel. Data				x			

Fig. 8 Related systems and techniques

Calvanese et al. introduce a system that accesses data via an ontology instead of a schema. Ontologies also form an important part of the *Semantic Web*, a movement with the goal of improving sharing and reuse of data on the web by introducing standardized formats and techniques to add more semantics to the data. To enable the definition of concepts and relationships in ontologies, as well as define constraints and enable reasoning, ontology languages have been designed. In connection with the Semantic Web, the *Web Ontology Language (OWL)* is the most common language. The definition of constraints and rules enables more complex and precise queries over the data. However, the usability as well as the effort to define these constraint and the ontology, rule this approach out as an option for our scenario. Another approach introduced in connection with the semantic web is the concept of Linked Data, which addresses the publication format of data on the web. Instead of publishing (semi-)structured data in its original form, data is stored in RDF triples, a flexible format that can handle the structural heterogeneity of web data. Additionally, values from one data source can be linked with values from another source, also using RDF triples, similar to hyperlinks linking HTML documents. Datasets can also be linked to ontologies. A more detailed description of Linked Data is given in [2]. By reusing common ontologies, the semantic integration between the data can be improved and semantic ambiguity resolved, while keeping the autonomy of the original data sources. Our scenario, however, requires the user to have more control over the data in order to perform advanced, repeatable analysis tasks.

The next approach is a concept originally developed in the context of classic data integration. In order to reduce the manual effort required to design a global schema and define mappings between the local schemata and the global schema, a number of algorithms have been proposed to automate the task. A survey of automatic schema matching algorithms is presented in [15]. In [5], Chiticariu et al. present a technique to semi-automatically generate an integrated schema from multiple schemas. Automatic schema matching techniques are relevant and useful for our scenario, as we address non-expert users in a scenario where integration experts are not available and automatic processing is required.

Finally, we take a look at keyword search over relational data. In scenarios where users either do not have the expertise to formulate structured queries (for example in SQL) or where the structure of the data is unknown to the user so that structured queries cannot be defined, keyword queries are a suitable alternative. The keywords are interpreted to understand the semantics and then mapped to the internal schema of the system. A number of different algorithms have been proposed in the literature, for example [1] and [11]. Keyword search increases the usability of the system, but the required automatic mapping between keywords and schema can also negatively effect the precision of query results. However, in our scenario, keyword search is a useful approach to address the inexperienced users.

Figure 8 summarizes the individual features that are addressed by the selected approaches. Note that the presented list is by no means exhaustive and only presents a small selection of related approaches.

5 Conclusion and Outlook

The amount of data on the web is growing constantly and with it its potential to provide new information for people. Especially data published in the course of Open Data initiatives can be very valuable, as it reflects the state of important aspects of our society. However, there is currently no single system that supports non-expert users, such as journalists, with the search for and analysis of these heterogeneous, distributed public datasets. To address this issue, we have analyzed the specific requirements of a system that can support the ad-hoc analysis of Open Data. We first identified seven characteristic features of Data Managements Systems in general and then derived the specific requirements accordingly. Furthermore, we used these features to classify related work in the field in order to evaluate the suitability of the respective techniques.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum." 12 (2), S. 121–130. ISSN: 1610-1995.
http://dx.doi.org/10.1007/s13222-012-0091-9

As our main contribution, we introduced *OPEN*, a novel approach towards a single system that enables users to search for public datasets on the web as well as perform advanced analysis on the data. We presented our system as an extension of a classic DBMS. To reduce the integration effort for the user, we proposed an incremental integration process that is triggered directly by user requests and utilizes automatic schema and instance matching techniques. Additionally, we introduced a crowd-sourcing based indexing approach to create a semantic index from the public datasets that can be queried by our enhanced DBMS.

The envisioned concept of the *OPEN* system introduces a number of novel challenges. These include challenges specific to crowd-sourcing, such as quality control and optimal task granularity. Another challenge are modifications to the query interface that become necessary due to the dynamic addition of external datasets to the database. An important question here is, for example, how to address the new tables in a query if the original table names are too long or too generic to be usable. Further challenges are connected to the application of automatic matching techniques. When matching keywords to table or column names, for example, we need to study how to handle multiple possible matches. Addressing these challenges will be the subject of future work.

References

1. Bergamaschi S, Domnori E, Guerra F, Trillo Lado R, Velegrakis Y (2011) Keyword search over relational databases: a metadata approach. In: Proceedings of the 2011 international conference on management of data (SIGMOD '11), pp 565–576
2. Bizer C, Heath T, Berners-Lee T (2009) Linked data—the story so far. Int J Semantic Web Inf Syst **5**(3)
3. Blunschi L, Dittrich PJ, Girard OR, Kirakos S, Marcos K, Salles AV (2007) A dataspace odyssey: the iMeMex personal dataspace management system. In: CIDR, pp 114–119
4. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Poggi A, Rodriguez-Muro M, Rosati R, Ruzzi M, Savo DF (2011) The MASTRO system for ontology-based data access. J Web Semant 2:43–53
5. Chiticariu L, Hernández MA, Kolaitis PG, Popa L (2007) Semi-automatic schema integration in Clio. In: Proceedings of the 33rd international conference on very large data bases (VLDB '07), pp 1326–1329
6. Cunningham H, Maynard D, Bontcheva K, Tablan V (2002) Gate: a framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th anniversary meeting of the association for computational linguistics (ACL'02)
7. Demartini G, Difallah DE, Cudré-Mauroux P (2012) Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: Proceedings of the 21st international conference on world wide web (WWW '12). ACM, New York, pp 469–478. http://doi.acm.org/10.1145/2187836.2187900. doi:10.1145/2187836.2187900
8. Finin T, Murnane W, Karandikar A, Keller N, Martineau J, Dredze M (2010) Annotating named entities in twitter data with crowdsourcing. In: Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with amazon's mechanical turk (CSLDAMT '10). Association for Computational Linguistics, Stroudsburg, pp 80–88. http://dl.acm.org/citation.cfm?id=1866696.1866709
9. Franklin M, Halevy A, Maier D (2005) From databases to dataspaces: a new abstraction for information management. SIGMOD Rec 34:27–33
10. Franklin MJ, Kossmann D, Kraska T, Ramesh S, Xin R (2011) Crowddb: answering queries with crowdsourcing. In: Proceedings of the 2011 international conference on management of data (SIGMOD '11). ACM, New York, pp 61–72. http://doi.acm.org/10.1145/1989323.1989331. doi:10.1145/1989323.1989331
11. Hristidis V, Papakonstantinou Y (2002) Discover: keyword search in relational databases. In: Proceedings of the 28th international conference on very large data bases (VLDB '02), pp 670–681
12. Lawson N, Eustice K, Perkowitz M, Yetisgen-Yildiz M (2010) Annotating large email datasets for named entity recognition with mechanical turk. In: Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with amazon's mechanical turk (CSLDAMT '10). Association for Computational Linguistics, Stroudsburg, pp 71–79. http://dl.acm.org/citation.cfm?id=1866696.1866708
13. Madhavan J, Cohen S, Dong XL, Halevy AY, Jeffery SR, Ko D, Yu C (2007) Web-scale data integration: you can afford to pay as you go. In: CIDR, pp 342–350
14. Oleson D, Sorokin A, Laughlin GP, Hester V, Le J, Biewald L (2011) Programmatic gold: targeted and scalable quality assurance in crowdsourcing. In: Human computation
15. Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB J 10:334–350
16. Sarawagi S (2008) Information extraction. Found Trends Databases 1(3):261–377. doi:10.1561/1900000003
17. Vaz Salles MA, Dittrich JP, Karakashian SK, Girard OR, Blunschi L (2007) iTrails: pay-as-you-go information integration in dataspaces. In: Proceedings of the 33rd international conference on very large data bases (VLDB '07), pp 663–674

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden