Identity Mapping Solution for Open Data Federation

● Vivian Lee ● Masatomo Goto ● Tetsuya Izu

Open data has gained more popularity in recent years, especially with the launch of open-data government initiatives such as DATA.GOV and DATA.GOV.UK. Open data is any data that are freely available to everyone to use and republish as they wish. However, lack of published standards for open data is causing problems. For example, open data is often in a proprietary format, making it difficult to integrate data sets from different sources. To address these issues, Fujitsu Laboratories of Europe (FLE) has been developing a solution based on identity mapping that will enable isolated open data to be federated in an accurate manner. The solution consists of two fundamental technologies: entity identity reconciliation and mapping-based federation. Our solution facilitates the use of open data in organisations, thereby enabling better decision-making, which should improve their competitive advantage.

1. Introduction

Open Data has gained popularity in recent years with the rise of the Internet and World Wide Web, especially with the new initiatives of open-data government such as DATA.GOV and DATA.GOV.UK. The idea is that certain data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents, or other mechanisms of control.1)

The major open data sources are science and government. The movement toward open government data is of particular interest and has attracted lots of attention worldwide. According to a "2014 Open Data Benchmark Report" conducted by Socrata, Inc., a large proportion (41%) of governments have set a goal of opening up as much of their data as possible right now.2) This is mainly motivated by the idea that making government data available to the public can enhance public service and thereby improve decision making.

However, there is no standard way of publishing open data. This means that every information entity such as an organisation, a company, and/or a person is assigned a unique identifier only within a data source and that each of the heterogeneous data sources is often published in its own format. This situation poses a great challenge if one wants to federate information from heterogeneous open data sources about the same entity to get a more comprehensive view.

This paper presents the EIDER (entity identity reconciliation) platform developed by Fujitsu Laboratories of Europe (FLE). This platform uses an identity mapping technique to meet the great challenge of open data federation. The paper also introduces the core reconciliation technology used for identity mapping, describes how the mapping framework utilises the reconciliation results to achieve more accurate federation, and finishes with application examples of the EIDER solution.

2. Issues and technical challenges in open data

Although open data has clear potential in terms of providing more knowledge about society, there is also a question about its practicality. Data published in a non-standard format are not suitable for automatic input to a computer system. Moreover, how to use them may not be evident.

From a technical point of view, a major challenge during the data chaining process—from receiving the open data to applying the proper data analytic tool—is that data sets are quite often controlled by one administrative system, and each open data provider defines its own format for information entities, neglecting the existence of other information sources. These formats are not only variations of structured data formats like Excel, comma separated value (CSV), and relational database management system (RDBMS) tables, but they can also be in the form of semi-structured formats like XML and even unstructured ones such as a text article on Wikipedia, not to mention the latest Linked Data format consisting of RDF (resource description framework) syntax recommended by Tim Berners-Lee as part of the 5-star deployment scheme.

Leaving the data format problem aside, even if data from different sources could be easily integrated, identifying the data referring to a particular information entity would be problematic. Our discussions with customers have revealed that almost 70% of the effort during data chaining is spent on data cleansing and other pre-processes because each data source assigns its own local identifier to an information entity, so the entities have various identifiers. A standard way of defining these local identifiers is not yet available.

A typical example of the problem is illustrated in **Figure 1**. To collect all the data related to IBM from different data sources, one needs to specify the appropriate ID for each source. That is, to retrieve the stock price, one needs to specify "IBM:US", and to retrieve annual reports from the SEC (U.S. Securities and Exchange Commission) EDGAR (Electronic Data-Gathering, Analysis, and Retrieval system), one needs to specify IBM's CIK (Central Index Key) code, 0000051154. However, how do we know that "IBM:US" and "0000051154" both point to the information entity IBM? This is an initial issue.

3. EIDER platform

Apart from the entity identity reconciliation issue, there are two other major concerns about open data due to its open nature. One is that the quality of the data is not guaranteed, and the other is that the data tend to change very frequently, making it difficult to keep the reconciliation results up-to-date. The EIDER platform is designed primarily to tackle the entity identity mapping issue, but it also addresses these two concerns.

To enable automated open data federation, we implemented three key capabilities in the EIDER platform. Firstly, the platform offers the flexibility for users to select different reconciliation logic according to the characteristics of the data to be processed. Secondly, it can dynamically update data in accordance with changes in the external open data. And lastly it enables information to be dynamic integrated according to users' choice. These capabilities were implemented by developing a suite of technologies in several technical areas.

- Data extraction
- Reconciliation management and reconciliation
- Data integration
- Data freshening
- Query handling

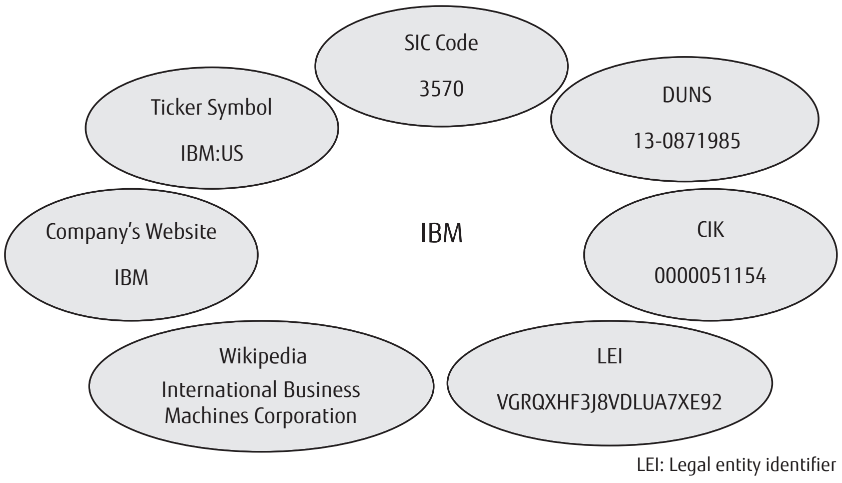

Figure 1
Example of ID reconciliation problem in open data.

4. Architecture

As shown in **Figure 2**, open data from heterogeneous sources are obtained using a data crawler/extractor. These data, called “delta data sources”, are stored in their original format in cache. The reconciliation manager launches reconciliation tasks using different reconciliators that are registered with the reconciliation manager. The results of this process are used to carry out data integration. The final outputs are stored in EIDER data storage for use in handling client application queries. In the following sections, we briefly introduce the concepts of the reconciliation manager and data refresher, focusing on FLE’s key technologies: identity mapping and data integration.

4.1 Reconciliation Manager

During our reconciliation research work, we realised that due to the diverse nature of the open data, relying on a single reconciliation logic would not always provide satisfactory results, especially when an information entity has a long history or is represented in multiple languages.

To maximize the strength of different reconciliation logics, and thereby get better results, we designed the reconciliation manager so that it can use any reconciliation mechanism at runtime. For example, it not only can run reconciliation with FLE’s in-house identity mapping method, which described in the next section, but it can also launch the same reconciliation request through OpenRefine.3)

During the system boot strapping stage, the reconciliation manager scans the reconciliation directory and registers the different reconciliation mechanisms. This information is then reported to the client applications, enabling users to choose different reconciliation mechanisms to compare or to derive more comprehensive results.

4.2 Data freshness

As true for any data federation or integration

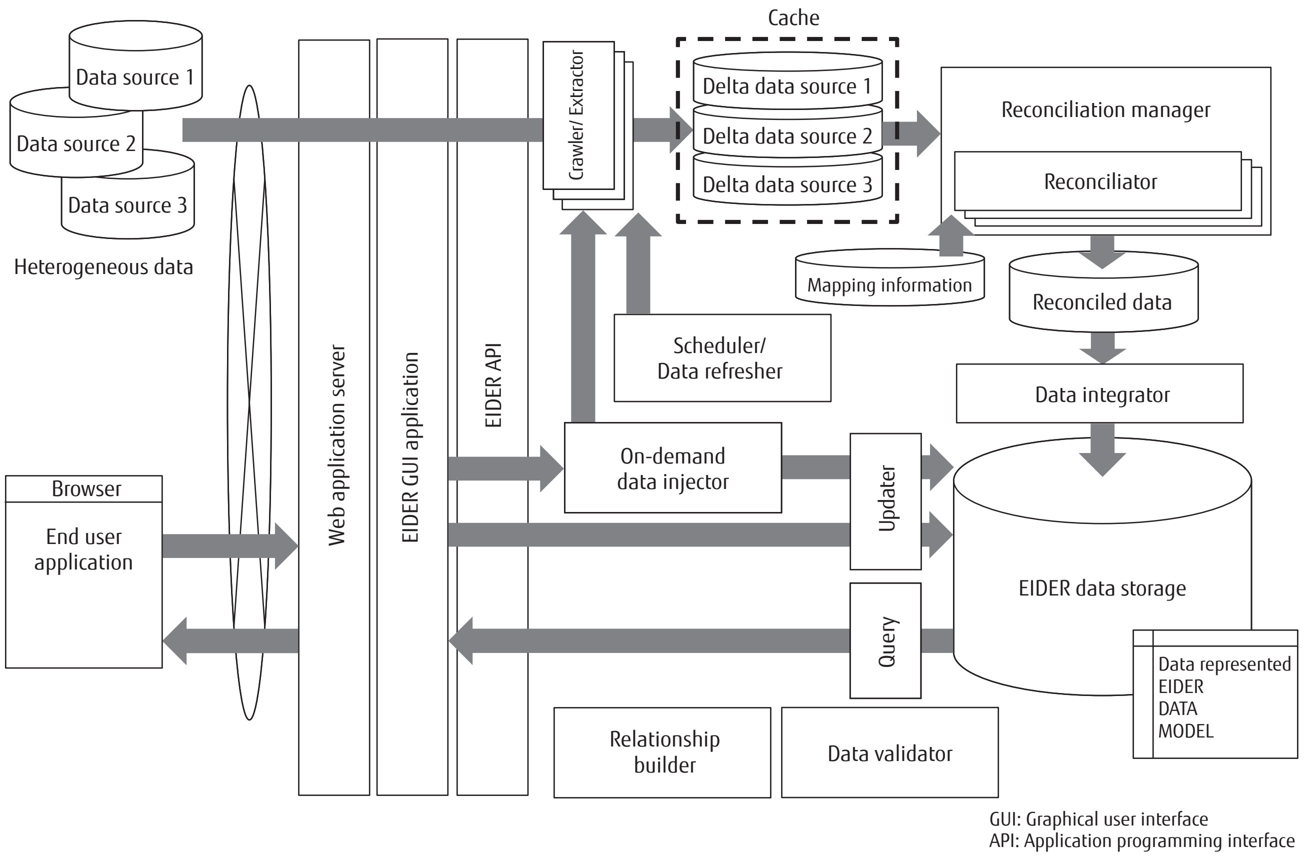

Figure 2
EIDER system architecture.

system, data freshness is an important issue because the system itself does not own or generate the data it stores. Data freshness is particularly important and difficult to achieve in an open data environment, where external open data sources can change at any time.

An important functionality of the EIDER platform is maintaining the freshness of the data. This means that the delta data sources in cache should mirror the changes in the external open data as closely as possible.

Two refresh methods are provided.

- Scheduled refresh
- On-demand refresh

As the names suggest, a refresh task can be either periodically scheduled, or triggered on-demand when a client issues a query to the server back end. The only data refreshed are the data related to the query.

5. Key technology: New method for identity mapping

A common way to reconcile the entity identifiers from heterogeneous open data sources is to use entity name as the mediator. However, name ambiguity can severely affect the accuracy of the reconciliation result. This is because the names are composed of strings of characters and can be affected by social factors, so a definitive or legally registered entity name could have multiple versions. This issue is exacerbated when the data are multilingual from multilingual sources.

Natural language processing (NLP), a well-established technology, can be used to parse chunk of text and analysing the relationship, or extract the relationship among entities that embedded inside the text. However, the various functions and techniques provided by NLP are all isolated and are too narrowly focused to solve specific problems. There is no comprehensive method that provides efficient and yet accurate open data entity identity reconciliation.

To overcome the shortcomings of NLP technology and thereby achieve more comprehensive and accurate reconciliation result, we developed and patented a reconciliation method consisting of three sub-processes executed in parallel. A diagram of the reconciliation component architecture is shown in **Figure 3**:

5.1 Data extractor and storage

This data extractor and data storage are equivalent to the crawler/extractor and delta data sources in the EIDER system architecture shown in Figure 2.

The data extractor retrieves relevant information from open data sources. Due to the various natures of the open data, the functionalities provided by the data extractor include file downloading, web crawling, and structured/semi-structured/unstructured data parsing and extraction.

The majority of data sources provide structured/

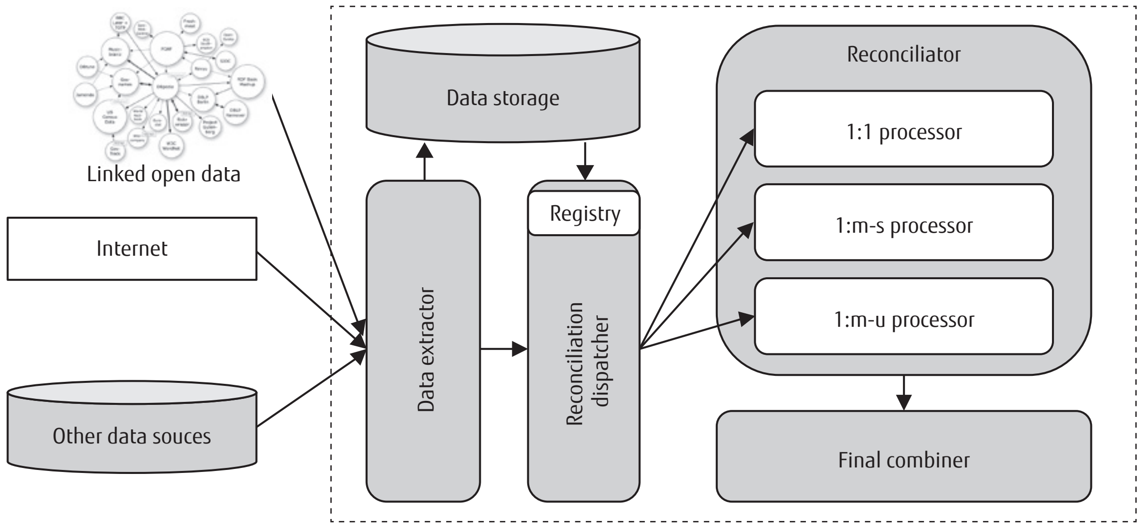

Figure 3
Reconciliation component architecture.

semi-structured data, so the downloaded data is saved in its original format in data storage. Although the reconciliation component can extract data at runtime, for performance reasons, it is better to store at least a majority of the data in a local network, thus avoiding network access overhead.

5.2 Reconciliation dispatcher

Once the data are stored in data storage, the reconciliation dispatcher inspects each source to produce a mapping value between the key and name for each data source. The process flow is as follows.

1) Check key values to see if there are duplicate entries.
2) If there are no duplicates, the mapping value is 1:1.
3) If there are duplicate entries, further check the name values. If the name values are the same to each of the duplicated entries, this means the mapping is still 1:1; otherwise, it is 1:m.
4) In the case of 1:m, further check the name value type. If it is a string and the sampling value length is less than a certain threshold, for example 100 characters, the mapping value is 1:m-s, and the string is said to contain only a company name. If the length is much longer than 100 characters, it can be assumed that it is a block of text containing company name information that needs to be parsed. Therefore, the mapping value is 1:m-u.

The mapping values can be extended depending on the situation, but here we only define three types, this is because these types are sufficient to deal with the file types that we are currently process - mainly CSV format. An example of each type of mapping is shown in **Figure 4**.

The mapping values generated by reconciliation dispatcher are then recorded in a registry, which is physically located either in memory or in a file system depending on the environment. An example of reconciliation dispatcher registry is shown in **Table 1**.

Using the information stored in the registry, the Reconciliator is able to group the data files according to their mapping values, and initialise the corresponding processors in the reconciliator.

5.3 Reconciliator

The reconciliator is the core component that processes the entity name reconciliation tasks. It consists

Table 1
Reconciliation dispatcher registry example.

Data Source	Mapping Value	Data Type
file://data/file1.csv	1.1	CSV
file://data/file2.ttl	1:m-s	turtle
file://data/file5.csv	1:m-u	CSV/text
file://data/file4.xml	1:m-s	XML
···	···	···

Example of 1:1 mapping

Key	Name	···
110	My Company Ltd.	···
120	Your Company Corp.	···
130	···	···

Example of 1:m-s mapping

Key	Name	···
110	My Company Ltd.	···
110	My Company Limited	···
120	Your Company Corp.	···

Example of 1:m-u mapping

Fujitsu Ltd. (富士通株式会社 Fujitsu Kabushiki-Kaisha) is a Japanese multinational information technology equipment and services company headquartered in Tokyo, Japan. It is the world's third-largest IT services provider measured by revenues (after IBM and HP)

Figure 4
Mapping examples.

of the 1:1 processor, 1:m-s processor, and 1:m-u processor submodules, each of which processes the data sources corresponding to its mapping value.

- 1:1 processor

This processor inspects the data sources that have a 1:1 mapping value, uses the name values from one data source as an index (this data source is defined manually in accordance with the user's preference), and reconciles the rest of the data sources in pairs:

<data source 1> reconcile <data source 2> => result 1
<data source 1> reconcile <data source 3> => result 2
<data source 1> reconcile <data source 4> => result 3
<data source 1> reconcile <data source 5> => result 4

Results 2, 3, and 4 are appended to result 1 upon completion.

The reconciliation method uses string similarity comparison, e.g. the Jaro-Winkler distance[4] algorithm to measure the similarity between the two name strings. The result of the comparison is a score with a value between 0 and 1, where 0 equates to no similarity and 1 is an exact match. The system can use any value as the criteria for similarity comparison. However, here, on the basis of experimental results, we defined 0.98 as the minimum score needed to say that two variations of a name point to the same entity:

if $f(name_1, name_2) >= 0.98$, *then* $name_1 \approx name_2$.

Although the selection of this minimum score is flexible, to obtain more accurate result, we have decided to choose a higher standard. Any matching data missed in this stage due to the high standard will still be caught by the following two processors.

- 1:m-s processor

This processor is designed to reconcile names for data sources having a mapping value of 1:m-s. It uses a semantic referencing technique to build a relationship between the two names. This technique is applicable to different types of data files, e.g. CSV, JSON (JavaScript Object Notation), and Turtle (Terse RDF Triple Language). For illustration purposes, here use RDF Turtle to explain the theory because RDF is a well-known semantically rich language.

RDF data consists of statements, each of which is produced in the form of subject-predicate-object expressions. The data types for subject, predicate, and object are typically URLs, for example,

<http://dbpedia.org/page/IBM>
dbpedia-owl:foundedBy dbpedia:Thomas_J._Watson.

This represents a simple statement, "IBM was founded by Thomas J. Watson", in RDF syntax. The URL surrounded by angle blankets represents the subject, the second URL represents the predicate, and the last URL represents the object. An object can also have a literal value, e.g. a string value.

For objects that have a URL type and multiple values, the inference result for the different values can only be that they belong to the same class, for example

<http://dbpedia.org/page/IBM>
dbpedia-owl:foundedBy dbpedia:Thomas_J._Watson
<http://dbpedia.org/page/IBM>
dbpedia-owl:foundedBy dbpedia:Charles_Renlett_Flint,

where objects dbpedia:Thomas_J._Watson and dbpedia: Charles_Renlett_Flint are both a type of person. Since these two URLs can be further dereferenced, they must contain different information and thus are not identical resources.

For objects that have a literal type and multiple values, the inference result can be translated as similar to. For at least at the semantic level, there is no obvious difference between these two objects because they are not further dereferenceable. This is especially true when the object value being examined is name, which is purely for identification purpose, and not necessary for an entity. When the 1:m-s processor processes this type of data, for example,

<http://dbpedia.org/page/IBM>
dbpprop:name "International Business Machines Corporation"
<http://dbpedia.org/page/IBM>
dbpprop:name "IBM Corp.",

it looks for the predicate value. If it contains "name" or variations of name, the statement is semantically describing the value of the name for the subject. The processor then further exams the object type. If it is literal, and there are multiple object literal values for one subject and predicate, the inference is reasoned to be

"International Business Machines Corporation" similar to "IBM Corp.".

With this method, the reconciliation result can be further enlarged compared to that obtained with only the 1:m-s Processor because it covers scenarios in which the similarity score is less than 0.98 although the names actually represent the same entity.

An example of such a property is the "dbpedia-owl:wikiPageRedirects" property. For example, if the “dbpedia-owl:wikiPageRedirects” property is pre-configured, the objects linked to by this particular predicate are known to contain links to other articles/pages with slightly different name labels.

- 1:m-u processor

This processor is designed to process unstructured data, e.g. a block of text, containing one or more values of name information. It is particularly helpful in a multi-lingual text scenario in which company names are written/translated into different languages. While the primary language content that it can process is English, the general technique is applicable to other languages as well.

The core technique for finding two or more matching names is pattern matching. For example, regular expressions are used to describe a sequence pattern (a text string), which is a sequence of characters forming a pattern that is searched for in the target text. A regular expression processor is used to process this regular expression into a nondeterministic finite automaton (NFA) in order to recognize the substrings matching the regular expression. Many programming languages provide regular expression capabilities, e.g. a processor to extract the searched for substring out of a block of text. As long as the 1:m-u processor is constructed using a major language, e.g. Java, Python, or C++, parsing a regular expression statement is not an issue. Given a text string like

> _“Fujitsu Ltd. (富士通株式会社 Fujitsu Kabushiki-Kaisha) is a Japanese multinational information technology equipment and services company headquartered in Tokyo, Japan. It is the world’s third-largest IT services provider measured by revenues (after IBM and HP)”,_

we can construct an example regular expression:

> _Pattern=“^(.*)(\(.*\)) is a|an .* company|corporation“._

And the return result looks like

> _Fujitsu Ltd. (富士通株式会社 Fujitsu Kabushiki-Kaisha)._

Further analysis can be done to extract Fujitsu Ltd. and _富士通株式会社_ again to refine the pattern matching result, but this is a relatively easy and local operation, involving only simple sub-string manipulation.

- Final combiner

The final combiner can be seen as a union operation on the reconciliation results produced by the reconciliator in the previous steps. However, this is not a simple merge-all operation. It is another cross-reference type reconciliation using the following processing logic.

Given resultsSet$_1$ (produced by the 1:1 Processor), resultsSet$_2$ (produced by the 1:1-s Processor), and resultsSet$_3$ (produced by the 1:1-m Processor), if any of the names in resultsSet$_1$ match at least one name in resultsSet$_2$, we can conclude that resultsSet$_1$ can be reconciled with resultsSet$_2$. The same algorithm can be used to reconcile resultsSet$_1$ and resultsSet$_3$ to produce the final result.

Once the names are reconciled, identifier reconciliation can be easily performed by simply using the following data integration module-the open data federation framework.

An example of the final reconciliation results is shown in **Table 2**.

6. Key technology: Federation

Data federation is not a new subject—there have been many attempts to improve federation performance over the past 40 years, e.g. data warehousing in the early 1980s and mediated schema in 2009. Starting around 2010, new approaches to semantic integration and to resolving the issues created by data modelling began to emerge.

Over the years, schema, data models, and data architecture semantics have led to many problems in data federation and integration, and each new solution solved only one aspect of the problem. The situation

Table 2
Example of reconciliation results.

Company_Name	CIK_Name	LEI_Name	Company_Ticker _Symbol_List_Name
International Business Machines Corp	International Business Machine Corp.	International Business Machine Corporation	International Business Machines Co.

has grown worse in the current open data era, where most of the data sources are in non-standard formats and do not have properly defined schema, data models, and semantic definitions not to mention to federate information embedded inside an unstructured data source, e.g. word document, emails, or twitter stream.

To address these problems, we have developed a federation framework based on a mapping file. It starts with construction of a simple mapping file, followed by entity reconciliation. Finally, mash-up data is created based on the reconciliation results. The architecture of the federation framework is illustrated in **Figure 5**.

The data extractor and data storage are the same as described in section 3.2.1. The mapping file editor provides an easy way for users to compose mapping files. It serves as a graphical user interface (GUI) wizard that guides the user through the composing process. The idea is to make the process as intuitive as possible as well as to eliminate syntax errors. The mapping file registry stores the mapping file to be used by the reconciliation manager and the federator.

6.1 Mapping file

Given a scenario in which the user would like to find the identifiers assigned by various financial regulators (see section 4.1) to a company, e.g. IBM, Microsoft, or Apple, from the LEIs issued by the CICI utility and from the CIKs contained in the EDGAR company fillings, together with its stock market ticker symbol, and then federate the results into a database, either a single CSV file or an RDBMS table. The solution may be to use each company name one at a time to search for the LEI, CIK, and ticker symbol from various websites by hand and then manually combine the results. This is a practical approach when the number of companies is small, but it would be very time consuming even for 100 companies. It would be even more so if the user wants more sophisticated financial data from various reports, or even from the same database, but distributed in different systems.

The federation framework and its mapping file concept were developed to address this scenario. Instead of analysing the schemas and data models of open data sources and then creating a mediated schema, the mapping file starts the federation process by asking the user "what do you want?" rather than "what do you have?". Acting as a template for the user, the mapping file tells the system how to federate the data to satisfy the user's request. A mapping file example for the scenario described is shown in **Figure 6**. It contains the company names, CIK identifier, LEI identifier.

The mapping file defines four concepts; 1) type, 2) name, 3) pattern, 4) source.

The concept values are separated by semicolons in the mapping file, and each line ends with a

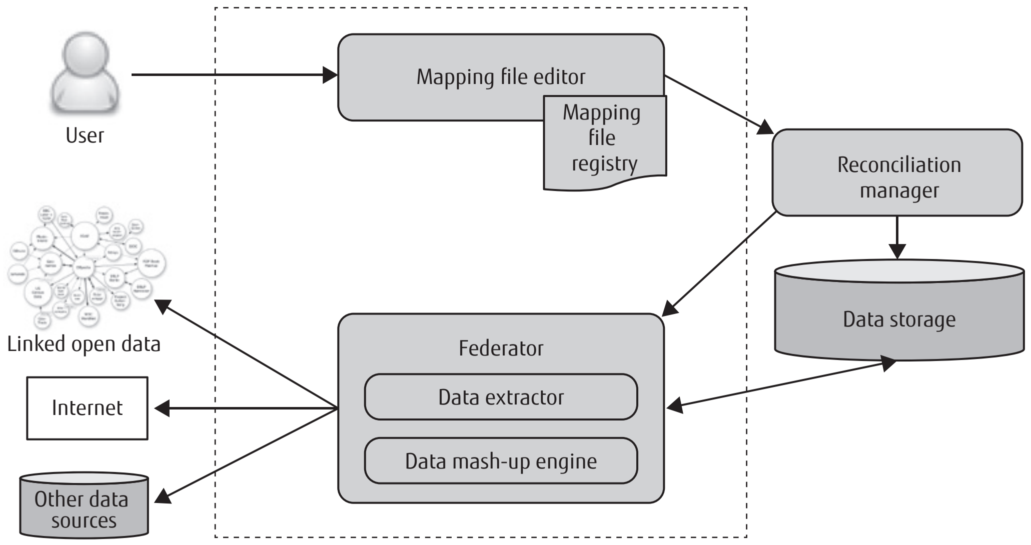

Figure 5
Federation framework architecture.

semicolon.

The output of a mapping file is a CSV file because CSV files have plain text format and can be easily read by any system without introducing errors. A CSV file can be used either as the final output or as intermediate output, which can subsequently be converted into another format. Each column in the output CSV file corresponds to a line in the mapping file.

1) Type

Two types are defined here: *key* and *map*. A key data source is used as index for the mapping process, to perform federation on the rest of the map type data sources. This is similar to a table's primary key but with a much less rigid structure.

2) Name

Name defines a column. The column name is the basis for the generated CSV file.

3) Pattern

Pattern is used to tell the system what kind of data to look for from a particular data source. It can be any primitive type or a string pattern in which data is embedded; for example, 00000 indicate a numerical value with five digits. This is a very powerful concept, especially for an unstructured data search.

4) Source

Source indicates where the data can be read. The value for source can be an http/https URL or a file system path. For a file system path, the path is retrieved either by configuring a *system.properties* file or by using the default path provided by the software.

Special notations are defined for the data sources that are in tabular format to enable the framework to find the exact value in each cell. Example notations are shown in the following line.

[type=key; name=LEI_Code; pattern=String; source=file:///query1.csv#Identifier@byColumn<LINK>file:///generated/csv/company-name_cik_lei_ticker-symbol.csv#LEI_Code@bySearch;]

This can be summarized as following table.

#	the string after the # key represents a column name inside the tabular file
@byColumn	@byColumn tells the system that this value search should be carried out by taking the value in that column.
@bySearch	@bySearch tells the system that this value search should be carried out by searching the entire CSV file using the given primary key value.
<LINK>	<LINK> indicates a situation in which a value is searched for in two different CSV files. This enables tunnelling when data is not in one file.

The four concepts defined by the mapping file are the very basic designs. The mapping file enables the use of richer extensions beyond type, name, pattern, and source in the case of insufficiency, as long as the mash-up engine (section 6.3) is also enhanced to understand the extended concept.

6.2 Sequence file

A sequence file is designed to resolve dependency issues among the mapping files. It simulates a data workflow process for which multiple mapping files are needed. For example, in addition to identity mapping, a user might also want a report on the fiscal year 2014 statements made by EDGAR and NASDAQ. This report can be processed using the two mapping files in sequence:

1) Define company identity mapping file, as shown in Figure 6.
2) Define fiscal year 2014 statements mapping file, in which financial figures are retrieved using identities produced in step 1.

These two mapping files are then simply place in a sequence file:

Company-identity.map
Fiscal-year-2014-statement.map.

When the framework processes the sequence file, it processes the data in accordance with the order of

```
type=key; name=Company_Name; pattern=String; source=file:///company_name.data;
type=map; name=CIK_Code; pattern=String; source=http://www.sec.gov/edgar/searchedgar/cik.htm;
type=map; name=LEI_Code; pattern=String; source=https://www.ciciutility.org/search.jsp;
```

Figure 6
Company identity mapping file example.

mapping files.

6.3 Data mash-up engine

For the example shown in Figure 4, the mash-up engine reads the first column value (which has type=key) into memory, the list of company names from the "company_name.data" file. Note that ".data" is a proprietary file type invented together with the federation framework; it contains a list of values that can be used as an index to search for other values in other data sources. When the mash-up engine reads the value of "International Business Machines Corp", it performs a lookup in Table 2, which contains the reconciliation results, and finds "International Business Machine Corp." as the name to use for extracting or searching for the CIK code (an identity issued by EDGAR) from the U.S. Securities and Exchange Commission (SEC) Website,[5] "International Business Machine Corporation" as the name to use for extracting or searching for the LEI code (the identity issued by the CICI utility), and "International Business Machines Co." as the name to use for searching for the ticker symbol from the file located at file:///company_ticker_symbol_list.csv#Symbol@bySearch. An example of such kind of process output is illustrated in **Table 3**.

Note that the content in Table 3 is a very simple example for illustration purposes only. It is possible to perform more complicated federation by using the same processing logic.

7. Possible applications

By using an initial prototype, FLE has successfully applied our solution to two different domains: finance and tourism.

7.1 Financial application

To avoid another "Lehman Shock", many activities are conducted to enable data analysed against the data all over the world, which also happened in the standardisation space. For financial application, we aimed at gathering information about companies all over the world. The information collected includes, but is not limited to, name (formally registered name and aliases from other open data sources), address, description, various identifiers assigned by different organisations and regulators, merger and acquisition history, and related companies, e.g. competitors, partners, parent company, and subsidiaries.

We identified several data sources. We choose LEI[6] data as the most reliable source for company names. For companies primarily based in the U.S., the data downloaded from the GMEI Utility[7] was the key source. If a company did not have an LEI, we used the legally registered name with legal entities as the primary source. For the U.K., the legal entity for company name registration is the "Companies House"[8] while for the U.S., it depended on the state. Companies in the U.S. also have another important identity, the CIK, issued by the SEC. In addition, all the big stock exchanges assign a ticker symbol to companies that trade with them. Example exchanges are NASDAQ, the New York Stock Exchange, the London Stock Exchange, and the Tokyo Stock Exchange. DBPedia endpoint[9] also provides information on 85 583 companies worldwide.

By using our reconciliation method to reconcile company names, we are able to produce name mappings across heterogeneous data sources and were thus able to accurately identify the same company in various open data sources. This enabled us to automatically federate open data for a company more accurately. This method thus provides a framework for implementing new analysis methods.

7.2 Tourism application

We also used the EIDER platform to federate tourism information, especially data on monuments.

Table 3
Federation results.

Company_Name	CIK_Code	LEI_Code	Ticker_Symbol
Microsoft Corp	0000789019	INR2EJN1ERAN0W5ZP974	MSFT
International Business Machines Corp	0000051143	VGRQXHF3J8VDLUA7XE92	IBM
...	...	...	...

We focused on gathering monument information for Andalusia in Spain.

Tourism is one of the major industries in European countries such as Spain. The presence of incorrect and/or inappropriate data in open data would greatly affect the industry. With the EIDER platform technology, users can easily find incorrect and/or inappropriate data, enabling the problem to be quickly solved.

The information we collected included, but was not limited to, the various names found in different sources, local identifiers, location (longitude and latitude), style, and year built.

The data sources include MOSAIC,[10] Spanish DBPedia endpoint,[11] and Yelp.[12] The Yelp data was extracted using their REST API and were downloaded as JSON files. The DBPedia data were retrieved by sending a SPARQL query to the public endpoint, which returned information for a total of 1351 monuments.

Application of our reconciliation logic resulted in the generation of twice as many matching results than simple string matching. We were thus able to federate heterogeneous data about the same monument in an automated and accurate way.

8. Conclusion

Entity identification in open data is still a relatively new topic. The only reference we have found so far is the white paper presented by James and Nigel[13] in 2014. Nevertheless, our initial federation results for two applications have shown good improvement compared to the traditional method. Our results have attracted much interest from both the technical and business communities.

We will further enhance our research by incorporating more reconciliation algorithms through the reconciliation manager, and by enhancing data freshness with the scheduled and on-demand techniques.

References

1) Auer, S.R. et al.: DBPeida: A Nucleus for a Web of Open Data. The Semantic Web. Lecture Notes in Computer Science, Vol. 4825, pp. 722–735 (2007).
2) Socrata: 2014 Open Data Benchmark Report. *http://discover.socrata.com/2014-benchmark-report-email-thank-you.html?aliId=7973252*
3) OpenRefine. *http://openrefine.org*
4) W. E. Winkler: String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Proceedings of the Section on Survey Research Methods (American Statistical Association), 1990, pp. 354–359.
5) U.S. Securities and Exchange Commission: EDGAR Company Filing | CIK Lookup. *http://www.sec.gov/edgar/searchedgar/cik.htm*
6) LEI ROC. *http://www.leiroc.org*
7) GMEI Utility. *https://www.gmeiutility.org/*
8) Companies House. *https://www.gov.uk/government/organisations/companies-house*
9) DBPedia endpoint. *http://dbpedia.org/sparql*
10) MOSAIC. *http://www.iaph.es/patrimonio-inmaterial-andalucia/*
11) Spanish DBPedia endpoint. *http://es.dbpedia.org/sparql*
12) Yelp. *https://www.yelp.com/developers/documentation/v2/search_api*
13) J. Powell et al.: Creating Value with Identifiers in an Open Data World. Open Data Institute and Thomson Reuters, March 2014.

Vivian Lee
Fujitsu Laboratories of Europe Ltd
Ms. Lee is currently engaged in research and development of technologies for integration, federation, and reconciliation of big and open data.

Tetsuya Izu
Fujitsu Laboratories of Europe Ltd
Dr. Izu is engaged in R&D on big data utilization technologies.

Masatomo Goto
Fujitsu Laboratories of Europe Ltd
Mr. Goto is engaged in research on big and open data analysis and further standardisation of the eXtensible Business Reporting Language (XBRL). He is a member of the XBRL Standards Board at XBRL International and also a member of the International Financial Reporting Standards (IFRS) Taxonomy Consultative Group of the IFRS Foundation.