

ANALYSIS CERTIFICATE



Account : **Bernard Merialdo**

ID : **g3xu5sj1**

Title : **Thesis ahmad assaf-rapporteurs.pdf**

Folder : **Ahmad Assaf**

Comments : *Not available*

uploaded on the : 10/07/2015 4:16 PM

Similarity document :

 **2%**

Similarities section 4 :

 **<1%**

DETAILED INFORMATION

Title : Thesis Ahmad Assaf-Rapporteurs.pdf

Description : Ahmad Assaf

Analysed on : 10/07/2015 4:38 PM

Login ID : wdri93qu

uploaded on the : 10/07/2015 4:16 PM

Upload type : manual submission

File name : Thesis Ahmad Assaf-Rapporteurs.pdf

File type : pdf

Word count : 11634

Character count : 74906

TOP PROBABLE SOURCES AMONG 1 PROBABLE SOURCE

SIMILARITIES FOUND IN THIS DOCUMENT/SECTION

Matching similarities : **<1 %** 

Assumed similarities : **<1 %** 

Accidental similarities : **<1 %** 

Highly probable sources - 0

Less probable sources - 0

Accidental sources- 4 Sources

Ignored sources - 12

HIGHLY PROBABLE SOURCES

0 Source

Similarity

LESS PROBABLE SOURCES

0 Source

Similarity

ACCIDENTAL SOURCES

4 Sources

























Similarity

- | | |
|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| 1.  Document: bhpz29 - belongs to another user |  <1% |
| 2.  Source Compilatio.net cejrv3 |  <1% |
| 3.  Source Compilatio.net efvx68 |  <1% |
| 4.  Source Compilatio.net emux1 |  <1% |

IGNORED SOURCES


12 Sources


Similarity

- | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| 1.  www.eurecom.fr/.../download/mm-publi-4447.pdf |  27% |
| 2.  hal.inria.fr/.../RUBIX_A_Framework_...th_Linked_Data.pdf |  24% |
| 3.  www.eurecom.fr/.../Publications/Assaf_Troncy-wod12.pdf |  24% |
| 4.  hal.archives-ouvertes.fr/.../hal-00873637/document |  19% |
| 5.  ahmadassaf.com/.../SNARC_An_Approach_for_Aggregating_and_Recommending |  19% |
| 6.  www.researchgate.net/.../224951559_Improvin...g_with_Linked_Data |  15% |
| 7.  arxiv.org/.../pdf/1205.2691.pdf |  15% |
| 8.  arxiv.org/.../pdf/1205.2691 |  15% |
| 9.  www.researchgate.net/.../278815853_SNARC_-_...zed_Social_Content |  8% |
| 10.  hal.archives-ouvertes.fr/.../hal-01082169/document |  7% |
| 11.  2014.eswc-conferences.org/.../files/eswc2014pd_submission_98.pdf |  7% |
| 12.  citeseerx.ist.psu.edu/.../viewdoc/summary |  5% |

SIMILARITIES FOUND IN THIS DOCUMENT/SECTION

Matching similarities : <1 % 

Assumed similarities : <1 % 

Accidental similarities : <1 % 

TEXT EXTRACTED FROM THE DOCUMENT

The social web (Web 2.0) is about websites and services designed and developed to support and foster social interactions [122]. The social web spans services like blogs, wikis, crowdsourcing and social media services. These services are often accompanied with APIs that allow rapid community driven expansion of complementary services. In the work of this thesis, we use the following popular social media services: • Twitter2 : A service allowing users to send rich short messages of 140 characters called "tweets" or "microposts". Twitter allows users to "follow" each other and start private conversations. Users can interact with tweets by replying or forwarding (re-tweeting) them. Tweets can contain multimedia parts like photos or videos and can be tagged with certain keywords preceded by the hash character called "hashtag" e.g., #tag. • Google+3 : A social media service owned by Google focusing on driving interestbased conversations and interactions. In its core, Google+ evolves around the concept of "circles" which enable users organize people into groups or lists. Similarly to Twitter, posts can be tagged with hash-tags entered manually by users or added automatically by Google+. • Stack Exchange4 : A question and answer group of websites. Each website specializes in a topic like technology, politics, food, etc. Users are encouraged to provide answers and helpful comments by providing a reputation award system allowing the content to be self-moderating. • YouTube5 : A video sharing website owned by Google. The site allows users to upload, view and share videos. Moreover, YouTube allows live streaming of events with the ability to interact with the video feed through comments.

2 3

<http://twitter.com> <https://plus.google.com> 4 <http://stackexchange.com> 5 <http://youtube.com>

94

Chapter 6. Background

- Vimeo6 : Another video sharing website with a more focused community of professionals in various areas than YouTube.
- Slideshare7 : A slide hosting service where users can upload their presentations in various formats to be viewed and shared. The website also supports documents, videos and webinars and acts as an educational and e-learning hub.

6 7

<http://vimeo.com> <http://slideshare.com>

Chapter 7

Data Integration in the Enterprise

Companies have traditionally performed business analysis based on transactional data stored in legacy relational databases. The enterprise data available for decision makers was typically relationship management or enterprise resource planning data. However social media feeds, weblogs, sensor data, or data published by governments or international organizations are nowadays becoming increasingly available [28]. The quality and amount of structured knowledge available make it now feasible for companies to mine this huge amount of public data and integrate it in their next-generation enterprise information management systems. Analyzing this new type of data within the context of existing enterprise data should bring them new or more accurate business insights and allow better recognition of sales and market opportunities [94]. These new distributed sources, however, raise tremendous challenges. They have inherently different formats, access protocols or query languages. They possess their own data model with different ways of representing and storing the data. Data across these sources may be noisy (e.g. duplicate or inconsistent), uncertain or be semantically similar yet different. Integration and provision of a unified view for these heterogeneous and complex data structures therefore require powerful tools to map and organize the data. Establishing data knowledge bases in the enterprise can facilitate the provision of data integration services [55]. In this chapter, we present our work in using DBpedia as an internal knowledge base. We further present a set of services that we implemented on top of DBpedia allowing entity disambiguation and enhancing schema matching. These services enable business users to semi-automatically combine potentially noisy data residing in heterogeneous silos. Semantically related data is identified and appropriate mappings are suggested to users. On user acceptance, data is aggregated and can be visualized directly or exported to Business Intelligence reporting tools. Finally, we perform a reverse engineering of the Google Knowledge graph panel to find out what are the most relevant properties for an entity. We compare these results with a survey we conducted on 152 users and show how we can represent and explicit this knowledge using the Fresnel vocabulary.

96

Chapter 7. Data Integration in the Enterprise

7.1

Enterprise Knowledge Bases

A Knowledge Base (KB) is a large repository of structured and unstructured information representing facts about the general world or specific domains. DBpedia1 is an example of a general knowledge base that will be used in this chapter as

an illustration. It is a crowd-sourced community effort to extract information from Wikipedia and present it in structured accessible formats [23]. DBpedia provides dumps which are split into several parts making it easy to import and experiment with the data. In this part, we are mainly interested in the following datasets that hold the main core information of DBpedia:

- Mapping-based Types: Contains types assignment from the DBpedia ontology to the entities extracted from Wikipedia.
- Mapping-based Properties: Contains properties extracted from Wikipedia. Since we can have different names for the same attribute (e.g. birthplace and placeofbirth), DBpedia uses a mapping-based approach to unify these attributes and generate high quality linked data.
- Extended Abstracts: Contains the first section of Wikipedia articles.
- Images: Images and their corresponding thumbnails together with a link to the license.
- Inter-language Links: Links between IRIs from different languages to the English IRI.

ABSTRACTS ASSOCIATIONS INTERLANGUAGE PROPERTIES TYPES

Columns uri, abstract source, type, target uri, sameas uri, typ, value uri, type, incomingno, order

Table 7.1: Tables structure for DBpedia in HANA column store DBpedia is modeled as an RDF graph. A natural way to import DBpedia into SAP HANA would have been to its graph engine, namely AIS. To do that, we would need to map the RDF triples to the AIS data model (see Section 6.3.2). This requires to:

- Create a new Term for every triple's distinct predicate. If the object of the triple is not a subject but a literal, the Term gets a technical type corresponding to

1

<http://dbpedia.org>

7.1. Enterprise Knowledge Bases

97

the datatype (if not known the type string is used). If the object refers to another Info Item, the Term gets the technical type for being an Association.

- Create a new Info Item for each distinct RDF subject.
- Store all literals as Attributes that assigned to the Info Item which corresponds to the subject of the triple.
- Create Associations to the subject's Info Item that point to the Info Item which has the URI of the object of the triple. However, since AIS is still under major development, various limitations and performance issues prevented us to import a reliable and functional version of DBpedia. As a result, we decided to import DBpedia into HANA's column store. Table 7.1 shows the tables structure used.

7.1.1

Entity Disambiguation with DBpedia in SAP HANA

After successfully, importing DBpedia into HANA's column store, we need to create a service that is able to disambiguate a query string (full documentation of the API is found in appendix D). HANA has a built-in (fuzzy) text search function that can be used. However, relying on string matching only is not sufficient. To better rank the search results, we combined a link-based rank approach that takes into account the number of incoming associations as shown in Equation 7.1.

URI	Apple	Apple Inc	Apple Records	Apple II	Apple IIGS	Apple Corps	Fiona Apple	Apple IIe	Apple Hong	Apple DOS	Text Search Score	No. of Incoming
1.00000	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	31	393
362	261	95	39	39	12	7	6	Combined Score	1.00000	0.85711	0.84527	0.80673
0.74337	0.72199	0.72199	0.71169	0.70978	0.70940							

Table 7.2: Results of combining text search score with the number of incoming associations for query "apple" While the number of outgoing associations can vary between entities, it is rather considered as an indicator for how well described an entity in DBpedia is. The number of incoming associations is less dependent on one single entity, it takes into account how many other entities link to the entity thus reflecting its popularity.

$$\text{incomingWeight} = \text{incomingNo} \times \text{txtScore} / \text{largestIncomingNo} \quad (7.1)$$

98

URI	Apple	Apple Inc	Apple Records	Apple II	Apple IIGS	Apple Corps	Fiona Apple	Apple IIe	Apple Hong	Apple DOS
Text Search Score	1.00000	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711	0.70711
No. of Incoming	31	393	362	261	95	39	39	12	7	6
Combined Score	0.87366	0.85711	0.83344	0.75634	0.62963	0.58688	0.58688	0.56627	0.56245	0.70940

Chapter 7. Data Integration in the Enterprise

Text Search Score 1.00000 0.70711 0.70711 0.70711 0.70711 0.70711 0.70711 0.70711 0.70711 0.70711 No. of Incoming 31 393 362 261 95 39 39 12 7 6 Combined Score 0.87366 0.85711 0.83344 0.75634 0.62963 0.58688 0.58688 0.56627 0.56245 0.70940

Table 7.3: Results of the enhanced equation 7.2 and its effect on the overall score for query "apple"

$$\text{incomingWeight} = (\text{incomingNo} / (\text{largestIncomingNo} - \text{incomingNo})) + \text{txtScore} / \text{largestIncomingNo} \quad (7.2)$$

(7.2)

7.2

Enhancing Schema Matching

Schema matching is typically used in business to business integration, metamodel matching, as well as ETL processes. For non-IT specialists the typical way of comparing financial data from two different years or quarters, for example, would be to copy and paste the data from one Excel spreadsheet into another one, thus creating redundancies and potentially

introducing copy-and-paste errors. By using schema matching techniques it is possible to support this process semi-automatically, i.e. to determine which columns are similar and propose them to the user for integration. This integration can then be done with appropriate business intelligence tools that provide visualizations. One of the problems in performing the integration is the quality of data. The columns may contain data that is noisy or incorrect. There may also be no column headers to provide suitable information for matching. A number of approaches exploit the similarities of headers or similarities of types of column data. In this section, we propose a new approach that exploits semantic rich typing provided by our entity disambiguation API in Section 7.1.1.

7.2.1

Related Work

While schema matching has always been an active research area in data integration, new challenges are faced today by the increasing size, number and complexity of data sources and their distribution over the network. Datasets are not always correctly typed or labeled and that hinders the matching process.

7.2. Enhancing Schema Matching

99

In the past, some work has tried to improve existing data schemas [114] but literature mainly covers automatic or semi-automatic labeling of anonymous datasets through Web extraction. Examples include [41] that automatically labels news articles with a tree structure analysis or [149] that defines heuristics based on distance and alignment of a data value and its label. These approaches are however restricting label candidates to Web content from which the data was extracted. [38] goes a step further by launching speculative queries to standard Web search engines to enlarge the set of potential candidate labels. More recently, [101] applies machine learning techniques to respectively annotate table rows as entities, columns as their types and pairs of columns as relationships, referring to the YAGO ontology. The work presented aims however at leveraging such annotations to assist semantic search queries construction and not at improving schema matching. With the emergence of the Semantic Web, new work in the area has tried to exploit Linked Data repositories. The authors of [139] present techniques to automatically infer a semantic model on tabular data by getting top candidates from Wikitology [50] and classifying them with the Google page ranking algorithm. Since the authors' goal is to export the resulting table data as Linked Data and not to improve schema matching, some columns can be labeled incorrectly, and acronyms and languages are not well handled [139]. In the Helix project [68], a tagging mechanism is used to add semantic information on tabular data. A sample of instance values for each column is taken and a set of tags with scores are gathered from online sources such as Freebase2. Tags are then correlated to infer annotations for the column. The mechanism is quite similar to ours but the resulting tags for the column are independent of the existing column name and sampling might not always provide a representative population of the instance values.

7.2.2

Proposition

Open Rene (formerly Google Rene) 3 is a tool designed to quickly and efficiently process, clean and eventually enrich large amounts of data with existing knowledge bases such as Freebase. The tool has however some limitations: it was initially designed for data cleansing on only one data set at a time, with no possibility to compose columns from different datasets. Moreover, Open Rene has some strict assumptions over the input of spreadsheets which make it difficult to identify primitive and complex data types. Open Rene makes use of a modular web application framework similar to OSGi called Buttery4. The server-side written in Java maintains states of the data (undo/redo history, long-running processes, etc.) while the client-side implemented in

23

<http://www.freebase.com/> <http://openrefine.org/> 4 <http://code.google.com/p/simile-butterfly/>

100

Chapter 7. Data Integration in the Enterprise

JavaScript maintains states of the user interface (facets and their selections, view pagination, etc.). Communication between the client and server is done through REST web services. The AutoMapping Core (AMC) [119] is a novel framework that supports the construction and execution of new matching components or algorithms. AMC contains several matching components that can be plugged and used, like string matchers (Levenshtein, JaroWinkler, etc.), data types matchers and path matchers. It also provides a set of combination and selection algorithms to produce optimized results (weighted average, average, sigmoid, etc.). RUBIX is the framework we created to enable business users to semi-automatically combine potentially noisy data residing in heterogeneous silos. Semantically related data is identified and appropriate mappings are suggested to users. On user acceptance, data is aggregated and can be visualized directly or exported to Business Intelligence reporting tools. We first map cell values with instances and column headers with types from popular datasets from the Linked Open Data Cloud. RUBIX leverages Open Rene and defines three new Buttery modules to extend the server's functionality (namely Match, Merge and Aggregate modules) and one JavaScript extension to capture user interaction with these new data matching capabilities.

7.2.3

Activity Flow

This section presents the sequence of activities and interdependencies between these activities when using our framework that is built on top of the entity disambiguation API in Section 7.1.1. Figure 7.1 gives an outline of these activities.

Figure 7.1: RUBIX Activity Workow The datasets to match can be contained in les (e.g., CSV, Excel spreadsheets,

7.2. Enhancing Schema Matching

101

etc.) or dened in Open Rene projects (step 1). The inputs for the match module are the source and target les and/or projects that contain the datasets. These projects are imported into the internal data structure (called schema) of the AMC [118] (step 2). The AMC then uses a set of built-in algorithms to calculate similarities between the source and target schemas on an element basis, i.e. column names in the case of spreadsheets or relational databases. The output is a set of similarities, each containing a triple consisting of source schema element, target element, and similarity between the two. These results are presented to the user in tabular form (step 3) such that s/he can check, correct, and potentially complete the mappings (step 4) as shown in Figure 7.2.

Figure 7.2: Screenshot showing the results mapping view in RUBIX Once the user has completed the matching of columns, the merge information is sent back to Open Rene, which calls the merge module. This module creates a new project, which contains the union of the two projects where the matched columns of the target project are appended to the corresponding source columns (step 5). The user can then select the columns that s/he wants to merge and visualize by dragging and dropping the required columns (step 6). Once the selection has been performed, the aggregation module merges the ltered columns and the result can then be visualized (step 7). As aggregation operations can quickly become complex, our default aggregation module can be replaced by more advanced analytics on tabular data. The integration of such a tool is part of future work.

7.2.4

Data Reconciliation

Reconciliation enables entity disambiguation, i.e. matching cells with corresponding typed entities in case of tabular data. Google Rene already supports reconciliation with Freebase but requires conrmation from the user. For medium to large datasets, this can be very time-consuming. To reconcile data, we therefore rst identify the columns that are candidates for reconciliation by skipping the columns containing numerical values or dates. We then use the disambiguation API in section 7.1.1

102

Chapter 7. Data Integration in the Enterprise

to query for each cell of the source and target columns the list of typed entities candidates. Results are cached in order to be retrieved by our similarity algorithms.

7.2.5

Matching Unnamed and Untyped Columns

The AMC has the ability to combine the results of dierent matching algorithms. Its default built-in matching algorithms work on column headers and produce an overall similarity score between the compared schema elements. It has been proven that combining dierent algorithms greatly increases the quality of matching results [119][135]. However, when headers are missing or ambiguous, the AMC can only exploit domain intersection and inclusion algorithms based on column data. We have therefore implemented three new similarity algorithms that leverage the rich types retrieved from Linked Data in order to enhance the matching results of unnamed or untyped columns. They are presented below.

7.2.5.1

Vector-based Similarity

The rst algorithm that we implemented is based on vector algebra. Let v be the vector of ranked candidate types returned by the disambiguation API for each cell value of a column. Then:

K

$v :=$

$i=1$

$a_i \quad t_i$

(7.3)

where a_i is the score of the entry and t_i is the type returned the disambiguation API. The vector notation is chosen to indicate that each distinct answer determines one dimension in the space of results. Each cell value has now a weighted result set that can be used for aggregation to produce a result vector for the whole column. The column result V is then given by:

n

$V :=$

$i=1$

v_i

(7.4)

We compare the result vector of candidate types from the source column with the result vector of candidate types from the target column. Let W be the result vector for the target column, then the similarity s between the columns pair can be calculated using the absolute value of the cosine similarity function: $s := |(V \cdot W)| / |V| |W|$ (7.5)

7.2. Enhancing Schema Matching

103

7.2.5.2

Pearson Product-Moment Correlation Coefficient (PPMCC)

The second algorithm that we implemented is PPMCC, a statistical measure of the linear independence between two variables (x, y) [90]. In our method, x is an array that represents the total scores for the source column rich types, y is an array that represents the mapped values between the source and the target columns. The values present in x but not in y are represented by zeros. We have:

SourceColumn $\{ \{R_1, Csr_1\}, \{R_2, Csr_2\}, \{R_3, Csr_3\} \dots \{R_n, Csr_n\} \}$ TargetColumn $\{ \{R_1, Ctr_1\}, \{R_2, Ctr_2\}, \{R_3, Ctr_3\} \dots \{R_n, Ctr_n\} \}$

(7.6)

Where R_1, R_2, \dots, R_n are different rich type values retrieved from Freebase, $Csr_1, Csr_2, \dots, Csr_n$ are the sum of scores for each corresponding r occurrence in the source column, and $Ctr_1, Ctr_2, \dots, Ctr_n$ are the sum of scores for each corresponding r occurrence in the target column. The input for PPMC consists of two arrays that represent the values from the source and target columns, where the source column is the column with the largest set of rich types found. For example: $X = [Csr_1, Csr_2, Csr_4, \dots, Csr_n]$ $Y = [0, Ctr_2, Ctr_4, \dots, Ctr_n]$ Then the sample correlation coefficient (r) is calculated using: $r =$

$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$

(7.7)

$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$

(7.8)

$s_x^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

$s_y^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$

Based on a sample paired data (x_i, y_i) , the sample PPMCC is: $r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$

$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

$s_x = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$

$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$

(7.9)

Where \bar{x} is the standard score, \bar{x} and s_x are the standard score, sample mean and sample standard deviation, respectively.

7.2.5.3

Spearman's Rank Correlation Coefficient

The last algorithm that we implemented to match unnamed and untyped columns is Spearman's rank correlation coefficient. It applies a rank transformation on the input data and computes PPMCC afterwards on the ranked data. In our experiments we used Natural Ranking with default strategies for handling ties and NaN values. The ranking algorithm is however configurable and can be enhanced by using more sophisticated measures.

104

Chapter 7. Data Integration in the Enterprise

7.2.6

Column Labeling

We showed in the previous section how to match unnamed and untyped columns. Column labeling is however beneficial as the results of our previous algorithms can be combined with traditional header matching techniques to improve the quality of matching. Rich types retrieved from Freebase are independent from each other. We need to find a method that will determine normalized score for each type in the set by balancing the proportion of high scores with the lower ones. We used Wilson score interval for a Bernoulli parameter that is presented in the following equation:

$$z \sqrt{\frac{p(1-p)}{n}} + \frac{z^2}{2n}$$

w =

$$2 \left(\frac{p(1-p)}{n} + \frac{z^2}{4n} \right)^{1/2}$$

$$\frac{2}{1 + \frac{z^2}{n}}$$

(7.10)

Here p is the average score for each rich type, n is the total number of scores and $z/2$ is the score level; in our case it is 1.96 to reflect a score level of 0.95.

7.2.7

Handling Non-String Values

So far, we have covered several methods to identify the similarity between “String” values, but how about other numeral values such as dates, money, distance, etc. ? For this purpose, we have implemented some basic type identifier that can recognize dates, money, numerical values, numerals used as identifiers. This will help us in better match corresponding entries. Adjusting AMC’s combination algorithms can be of great importance at this stage. For example, assigning weights to different matchers and tweaking the configuration can yield more accurate results.

7.2.8

Experiments

We present in this section results from experiments we conducted using the different methods described above. To appreciate the value of our approach, we have used a real life scenario that exposes common problems faced by the management in SAP. The data we have used come from two different SAP systems: the Event Tracker and the Travel Expense Manager. The Event Tracker provides an overview of events (Conferences, Internal events, etc.) that SAP Research employees contribute to or host. The entries in this system contain as much information as necessary to give an overview of the activity like the activity type and title, travel destination, travel costs divided into several sub categories (conference fees, accommodation, transportation and others), and duration related information (departure, return dates). Entries in the Event Tracker are generally entered in batches as employees fill in their planned events that they wish

7.2. Enhancing Schema Matching

105

to attend or contribute to at the beginning of each year. Afterwards, managers can either accept or reject these planned events according to their allocated budget. On the other hand, the Travel Expense Manager contains the actual expenses data for the successfully accepted events. This system is used by employees to enter their actual trip details in order to claim their expenses. It contains more detailed information and aggregated views of the events, such as the total cost, duration calculated in days, currency exchange rates and lots of internal system tags and identifiers. Matching reports from these two systems is of great benefit to managers to organize and monitor their allocated budget. They mainly want to: 1. Find the number of the actual (accepted) travels compared with the total number of entered events. 2. Calculate the deviation between the estimated and actual cost of each event. However, matching from these two sources can face several difficulties that can be classified in two categories: column headers and cells. Global labels (or column headers as we are dealing with spreadsheet files) can have the following problems: 1. Missing labels: importing files into Google Sheets with empty headers will result in assigning that column a dummy name by concatenating the word “column” with a number starting from 0. 2. Dummy labels or semantically unrelated names: this is a common problem especially from the data coming from the Travel Expense Manager. This can be applied to columns that are labeled according to the corresponding database table (i.e. lbl_dst to denote destination label). Moreover, column labels do not often convey the semantic type of the underlying data. The second category of difficulties is at cell (single entry) level: 1. Detecting different date formats: we have found out that dates held coming from the two systems have different formats. Moreover, the built-in type detection in Google Sheets converts detected date into another third format. 2. Entries from different people can be made in different languages. 3. Entries in the two systems can be incomplete, an entry can be shortened automatically by the system. For example, selecting a country in the Travel Expense Manager will result in filling out that country code in the exported report (i.e. France = FR).

106

Chapter 7. Data Integration in the Enterprise

4. Inaccurate entries: this is one of the most common problems. Users enter sometimes several values in some fields that correspond to the same entity. For example, in the destination column, users can enter the country, the airport at the destination, the city or even the exact location of the event (i.e. office location). The data used in our evaluation consists of around 60 columns and more than 1000 rows. Our source data set will be the data coming from Event Tracker, and our

target data set will be the data from the Travel Expense Manager. By manually examining the two datasets, we have found out that most of the column headers in the source table exist and adequately present the data. However, we have noticed few missing labels in the target table and few ambiguous column headers. We have detected several entries in several languages: the main language is English but we have also identified French, German. Destination eld had entries in several formats: we have noticed airport names, airports by their IATA code, country codes, and cities. Running AMC with its default matchers returns the matching results shown in Table 7.4.

Source Column Reason for Trip Begins On Ends On Total Trip Amount Pd by Comp Period Pers.No. M/Km Curr. CrCy
 Target Column Reason for Trip Trip Begins On Trip Ends On Total Cost Trip Destination Receipt Amount Paid by Company
 Period Number Sequential no. Total Miles/Km Currency Currency Similarity Score 1 0.8333334 0.8 0.7333335 0.7272725
 0.7142875 0.6904762 0.6666667 0.5555556 0.55 0.5 0.5

Table 7.4: Similarity Scores Using the AMC Default Matching Algorithms The AMC has perfectly matched the two columns labeled "Reason for Trip" using name and data type similarity calculations (the type here was identified as a String). Moreover, it has computed several similarities for columns based on the pre-implemented String matchers that were applied on the column headers and the primitive data types of the cells (Integer, Double, Float, etc.). However, there is no alignment found between the other columns since their headers are not related to each other, although the actual cell values can be similar. AMC's default configuration has a threshold of 50%, so any similarity score below that will not be shown.

7.2. Enhancing Schema Matching

107

The Cosine Similarity algorithm combined with the AMC default matchers produces the results shown in Table 7.5.

Source Column Reason for Trip tr dst Begins On Ends On Amount Curr. CrCy Total Trip Pd by Comp Period Trip Pers.No.
 M/Km Target Column Reason for Trip Trip Begins On Trip Ends On Receipt Amount Currency Currency Total Cost Trip
 Destination Paid by Company Period Number Trip Number Sequential no. Total Miles/Km Similarity Score 1 0.9496432
 0.9166667 0.9 0.8571428 0.75 0.75 0.7333335 0.7321428 0.6904762 0.6666667 0.6666667 0.5555556 0.55

Table 7.5: Similarity Scores Using the AMC Default Matching Algorithms + Cosine Similarity Method We notice that we have an increased number of matches (+2), and that the similarity score for several matches has improved. For example, the "tr dst" column is now aligned to the blank header. This shows that our approach allows performing schema matching on columns with no headers. For simplicity reason we have used the default combination algorithm for AMC which is an average of the applied algorithms (AMC's native and Cosine). We should also note that we have configured AMC's matchers to identify a "SIMILARTY UNKOWN" value for columns that could not be matched successfully, which will allow other matchers to perform better. For example, our semantic matchers will skip columns that do not convey semantic meaning thus not affecting the score of other matchers. Moreover, the relatively high similarity score of "tr dst" column is explained by the fact that the native AMC matching algorithm has skipped that column as it does not have a valid header, and the results are solely those of the Cosine matcher. Likewise, the Cosine matcher skips checking the "Cost" columns as they contain numeric values, and the implemented numerical matchers with the AMC's native matcher results are taken into account. Our numerical matchers' implementation gives a perfect similarity score for columns that are identified as date or money or IDs. However, this can be improved in the future as we can have different date hierarchy and numbers as IDs can present different entities. Combining this approach with the semantic and string matchers was found to yield good matching results. The (PPMCC) Similarity algorithm combined with the AMC default matchers

108

Chapter 7. Data Integration in the Enterprise

produces the results shown in Table 7.6.

Source Column Reason for Trip tr dst Begins On Ends On Total Trip Amount Curr. CrCy Pd by Comp Period Trip Pers.No.
 M/Km Target Column Reason for Trip Trip Begins On Trip Ends On Total Cost Trip Destination Receipt Amount Currency
 Currency Paid by Company Period Number Trip Number Sequential no. Total Miles/Km Similarity Score 1 0.97351624
 0.8333334 0.8 0.7333335 0.7321428 0.7142857 0.7041873 0.6931407 0.6904762 0.6666667 0.6666667 0.5555556 0.55

Table 7.6: Similarity Scores Using the AMC Default Matching Algorithms+ the PPMCC Similarity Method

Source Column Reason for Trip Begins On Ends On Total Amount Pd by Comp Currency2 Trip Pers.No. M/Km

Target Column Reason for Trip Trip Begins On Trip Ends On Total Cost Receipt Amount Paid by Company Curr. Trip
 Number Sequential no. Total Miles/Km

Similarity Score 1 0.8333334 0.8 0.7333335 0.7142857 0.6904762 0.6689202 0.6666667 0.5555556 0.55

Table 7.7: Similarity Scores Using the AMC Default Matching Algorithms + Spearman Similarity Method We notice that by plugging the Spearman method, the number of matches and similarity results have decreased (-4). After several experiments we have found that this method does not work well with noisy datasets. For instance, the similarity results returned by Cosine, Pearson's and Spearman's matchers for the {tr dst, empty header} pair is much higher: 95%, 97% and 43% respectively. To properly measure the impact of each algorithm, we have tested the three algorithms (Cosine, PPMCC and Spearman) alone by de-activating the AMC's default matchers on the above data set. We have noticed that generally, the Cosine and PPMCC matchers perform well, resulting in more matching and better similarity score. However, the Spearman method was successful in finding more matches but

7.2. Enhancing Schema Matching

with a lower similarity score than the others. To better evaluate the three algorithms, we have tested them on four different datasets extracted from the Travel Expense Manager and Event Tracker systems. We ensured that the different experiments will cover all the cases needed to properly evaluate the matcher dealing with all the problems mentioned earlier. We have found that generally the Cosine method is the best performing algorithm compared to the other two especially when dealing with noisy datasets. This was noticed particularly in our fourth experiment as the Cosine algorithm performed around 20%

better than the other two methods.

After investigating the dataset, we have found that several columns contained noisy and unrelated data. For example, in a "City" column, we had values such as "reference book" or "NOT KNOWN". To gain better similarity results we decided to combine several matching algorithms together. By doing so, we would benefit from the power of the AMC's string matchers that will work on column headers and our numeral and semantic matchers. The Cosine and PPMCC Similarity algorithms combined with the AMC default matchers produces the results shown in Table 7.7.

Source Column Reason for Trip tr dst Curr. Crncy Begins On Ends On Amount Total Trip Country/Group Pd by Comp Period
Trip Pers.No. M/Km Target Column Reason for Trip Currency Currency Trip Begins On Trip Ends On Receipt Amount Total
Cost Ctr2 Paid by Company Period Number Trip Number Sequential no. Total Miles/Km Similarity Score 1 0.96351624
0.79221311 0.78173274 0.77777785 0.76666665 0.7380952 0.7333335 0.7194848 0.6904762 0.6666667 0.6666667
0.5555556 0.55

Table 7.8: Similarity Scores Using the Combination of Cosine, PPMCC and AMC's defaults The combination of the above mentioned algorithms have enhanced generally the similarity scores for the group. Moreover, we notice that the column "Trip Country/Group" was matched with "Ctr2". This match was not computed singularly by any of the previous algorithms. However, we notice that the match {Trip, Trip Destination} is now missing, probably as the similarity score is below the defined threshold. Now, we will try and group all the mentioned algorithms. The combination of all Similarity algorithms with the AMC default matchers produces the results shown in

110 Table 7.8.

Source Column Reason for Trip tr dst Curr. Crncy Begins On Trip Country/Group Ends On Amount Total Trip Pd by Comp
Period Trip Pers.No. M/Km

Chapter 7. Data Integration in the Enterprise

Target Column Reason for Trip Currency Currency Trip Begins On Ctr2 Trip Ends On Receipt Amount Total Cost Trip
Destination Paid by Company Period Number Trip Number Sequential no. Total Miles/Km

Similarity Score 1 0.8779132 0.80033726 0.79380125 0.7708334 0.767311 0.7625 0.7410714 0.7333335 0.7321428
0.6904762 0.6666667 0.6666667 0.5555556 0.55

We notice that now we have an increased number of matches (15 compared to 14 in the previous trials). The column {Trip, Trip Destination} is matched again and the newly previously matched column {Trip Country/Group, Ctr2} has a higher similarity score. We have found that combining matching algorithms resulted in higher number of matches. Several tuning methods can be applied in order to enhance the similarity score as well. Trying other combination algorithms instead of the naive average will be an essential part of our future work.

7.3

Important Properties for Entities

Entities are generally described with a lot of properties. However, not all properties have the same importance. Some properties are considered as keys for performing instance matching tasks while other properties are generally chosen for quickly providing a summary of the key facts attached to an entity. In contrast to entities, it is difficult to assess which properties are more "important". In this section we provide a method enabling business users to select what properties should be used when depicting the summary of an entity. For example, when our analyst Dan wishes to enrich his reports with external data, he is overwhelmed by the number of dimensions he can add. We reverse engineered the Google Knowledge graph panel (see Figure 7.3) to find out what are the most "important" properties for an entity according to Google that can be used to enrich business reports. We compare these results with a survey we conducted on 152 users.

7.3.1

Reverse Engineering the Google KG Panel

7.3. Important Properties for Entities

111

Web scraping is a technique for extracting data from Web pages. We aim at capturing the properties depicted in the Google Knowledge Panel (GKP) that are injected in search result pages [14]. We have developed a Node.js application that queries all DBpedia concepts that have at least one instance which is owl:sameAs with a Freebase resource (since Freebase is the knowledge base behind the graph panel) in order to increase the probability that the search engine result page (SERP) for this resource will contain a GKP. We assume in our experiments that the properties displayed in Figure 7.3: Google knowledge graph panel played for an entity are type and context for the city of Nice, France dependent (country, time, query) which can affect the results. Moreover, we filter out generic concepts by excluding those who are direct subclasses of owl:Thing since they will trigger ambiguous queries. We obtained a list of 352 concepts. Algorithm 1 Google Knowledge Panel reverse

```

1: INITIALIZE equivalentClasses(DBpedia, Freebase) AS vectorClasses
2: Upload vectorClasses for querying processing
3: Set n AS number-of-instances-to-query
4: for each conceptType vectorClasses do
5: SELECT n instances
6: listInstances SELECT-SPARQL(conceptType, n)
7: for each instance listInstances do
8: CALL
http://www.google.com/search?q=instance
9: if knowledgePanel exists then
10: SCRAP GOOGLE KNOWLEDGE PANEL
11: else
12: CALL http://www.google.com/search?q=instance+conceptType
13: SCRAP GOOGLE KNOWLEDGE PANEL
14: end if
15: gkpProperties GetData(DOM, EXIST(GKP))
16: end for
17: COMPUTE occurrences for each prop gkpProperties
18: end for
19: gkpProperties

```

For each of these concepts (e.g., Band, Organization, ArchitecturalStructure), we retrieve n instances (in our experiment, n was equal to 100 random instances). For example, for the concept Band we retrieved: !Action Pact!, 12 Stones, 20 Fingers, etc. For each of these instances we issue a search query to Google containing the in_5
<https://github.com/ahmadassaf/KBE/blob/master/results/dbpediaConcepts.json>

112

Chapter 7. Data Integration in the Enterprise

stance label. Google does not serve the GKP for all user agents and we had to mimic a browser behavior by setting the User Agent to a particular browser. We use CSS selectors to check the existence of and to extract data from a GKP. An example of a query selector is `.om` (all elements with class name `om`) which returns the property DOM element(s) for the concept described in the GKP. From our experiments, we found out that we do not always get a GKP in a SERP. If this happens, we try to disambiguate the instance by issuing a new query with the concept type attached. However, if no GKP was found again, we capture that for manual inspection later on. Listing 1 gives the high level algorithm for extracting the GKP. The full implementation can be found at <https://github.com/ahmadassaf/KBE>. Instructions for installing and running the tool are available in section A.3. We finally observe that this experiment is only valid for the English Google.com search results since GKP varies according to top level names.

7.3.2

Evaluation

We conducted a user survey in order to compare what users think should be the important properties to display for a particular entity and what the GKP shows.

7.3.2.1 User survey

We set up a survey⁶ on February 25th, 2014 and for three weeks in order to collect the preferences of users in term of the properties they would like to be shown for a particular entity. We selected only one representative entity for nine classes: TennisPlayer, Museum, Politician, Company, Country, City, Film, SoccerClub and Book. 152 participants have provided answers, 72% from academia, 20% coming from the industry and 8% having not declared their affiliation. 94% of the respondents have heard about the Semantic Web while 35% were not familiar with specific visualization tools. The detailed results⁷ show the ranking of the top properties for each entity. We only keep the properties having received at least 10% votes for comparing with the properties depicted in a GKP. We observe that users do not seem to be interested in the INSEE code identifying a French city while they expect to see the population or the

points of interest of this city.

7.3.2.2 Comparison with Google Knowledge Graph

The results of the Google Knowledge Panel (GKP) extraction⁸ clearly show a long tail distribution of the properties depicted by Google, with a top N properties (N

The survey is at <http://eSurv.org?u=entityviz>
<https://github.com/ahmadassaf/KBE/blob/master/results/agreement-gkpusers.xls>
<https://github.com/ahmadassaf/KBE/blob/master/results/survey.json>

7 6

7.4. Summary

113

being 4, 5 or 6 depending on the entity) counting for 98% of the properties shown for this type. We compare those properties with the ones revealed by the user study. Table 7.9 shows the agreement between the users and the choices made by Google in the GKP for the 9 classes. The highest agreement concerns the type Museum (66.97%) while the lowest one is for the TennisPlayer (20%) concept. We think properties for museums or books are more stable than for types such as person/agent which vary significantly. We acknowledge the fact that more than one instance should be tested in order to draw meaningful conclusions regarding what are the important properties for a type.

Classes Agr. TennisPlayer 20% Museum 66.97% Politician 50% Company 40% Country 60% City 60% Film 60% SoccerClub 50% Book 60%

Table 7.9: Agreement on properties between users and the Knowledge Graph Panel With this set of 9 concepts, we are covering 301, 189 DBpedia entities that have an existence in Freebase, and for each of them, we can now empirically define the most important properties when there is an agreement between one of the biggest knowledge base (Google) and users preferences.

7.3.2.3 Modeling the preferred properties with Fresnel

Fresnel⁹ is a presentation vocabulary for displaying RDF data. It specifies what information contained in an RDF graph

should be presented with the core concept fresnel: Lens [121]. PROV-O10 is a vocabulary to describe semantically rich metadata with focus on providing detailed provenance, license and access information. We use those two vocabularies to explicitly represent what properties should be depicted when displaying an entity¹¹. This dataset can now be re-used as a conformation for any consuming application (see Appendix C for a snippet of the generated Fresnel le).

7.4

Summary

In this chapter, we presented an entity disambiguation API built on top of SAP HANA. We used this service in a framework to enable mashup of potentially noisy enterprise and external data. The API is used to annotate business reports with rich types. As a result, the matching process of heterogeneous data sources is improved. Our preliminary evaluation shows that for datasets where mappings were relevant yet not proposed, our framework provides higher quality matching results. Additionally,

<http://www.w3.org/2005/04/fresnel-info/> <http://www.w3.org/TR/prov-o/> 11
<https://github.com/ahmadassaf/KBE/blob/master/results/results.n3>

10 9

114

Chapter 7. Data Integration in the Enterprise

the number of matches discovered is increased when Linked Data is used in most datasets. In addition, we have shown that it is possible to reveal what are the “important” properties of entities by reverse engineering the choices made by Google when creating knowledge graph panels and by comparing users preferences obtained from a user survey. This is fundamentally different from the work in [134] where the authors created a generalizable approach to open up closed knowledge bases like Google’s by means of crowd-sourcing the knowledge extraction task. We are aware that this knowledge is highly dynamic, the Google Knowledge Graph panel varies across geolocation and time.

Chapter 8

Semantic Social News Aggregation

8.1

Introduction

With the rapid advances of the Internet, social media become more and more intertwined with our daily lives. The ubiquitous nature of Web-enabled devices, especially mobile phones, enables users to participate and interact in many different forms like photo and video sharing platforms, forums, newsgroups, blogs, micro-blogs, bookmarking services, and location-based services. Social networks are not just gathering Internet users into groups of common interests, they are also helping people follow breaking news, contribute to online debates or learn from others. They are transforming Web usage in terms of users’ initial entry point, search, browsing and purchasing behavior [48]. A common scenario that often happens while reading an interesting article, coming across a nice video or participating in a discussion in a forum is the growing interest to check related material around the information read. To do so, users might go to Twitter, Google+ or YouTube. They can try several times with several keywords to obtain the desired results. In the end, they might end up with several browser tabs opened and get distracted by the information overload from all these resources. The same happens in companies when business users are interested in information provided by corporate web applications like enterprise communities. In this chapter, we present SNARC, a semantic social news aggregator that leverages live rich data that social networks provide to build an interactive rich experience on both the Internet and Intranets. The service retrieves news related to the current page from popular platforms like Twitter, Google+, YouTube, Vimeo, Slideshare, StackExchange and the Web. As a possible front-end implementation, we have created a Google Chrome extension which enriches the user experience by augmenting related contextual information to entities on the page itself, as well as displaying related social news on a floating sidebar.

116

Chapter 8. Semantic Social News Aggregation

8.2

Underlying Mechanism

The back-end of SNARC consists of three major components: a document handler that creates a “Semantic Model” representing any web resource, a query layer that is responsible for disseminating queries to the supported social services and a data parser which processes the search results, wraps them in a common social model and generates the desired output.

8.2.1

Document Handler

The main idea behind SNARC is to provide a uniform model for web entities, whether they are blog entries, multimedia objects or micro-posts. To do so, SNARC creates a “Semantic Model” containing all the annotations and meta-data needed to query and reconcile social results.

Figure 8.1: SNARC's Document Handler The Semantic Model is created by the Document Handler (see Figure 8.1) which receives a web page URL and performs these three main steps: 1. Text Extraction: Fetch the webpage that corresponds to the received URL and extract the textual content using a set of heuristics. These latter identify the main content of the page by stripping unwanted HTML tags and rank the different sections based on their semantics, class names and order. In the beginning we have used Alchemy API1 to perform text extraction; but we have chosen to implement a simpler method ourselves which saved us an extra API call. 2. Language Detection: Detect the web page language using the Language Detection service of Alchemy API. This is necessary to match the desired language with compatible services like Twitter, YouTube, etc.

1

<http://www.alchemyapi.com>

8.2. Underlying Mechanism

117

3. Semantic Annotation: Annotating the extracted text is the most important step in this process. We use Zemanta Suggest2 and Alchemy API in order to extract: • Tags: These are the nest-grained queryable "keywords" that we use to retrieve the social results. From our experiments, combining tags results in better findings than using entities or concepts. However, we plan to evaluate the combination of keywords, entities and concepts in order to find the top-queryable terms that will retrieve the most relevant results on different abstraction levels. Tags retrieved from these services are ranked by condense values calculated by their internal algorithms, these values are normalized for each service. According to our experiments we have found that Alchemy's Keywords Extraction API returns a large set of closely related keywords (i.e. Android, Android Phone, Android Tablet, ...). To construct a good query we therefore need to provide a certain level of abstraction. We perform a cleaning process on those keywords by applying the Levenshtein distance to rule out closely related keywords by disregarding those with lower condenses. We perform a similar process on the result of the union between the keywords returned by Alchemy and Zemanta to ensure a sparse keywords set. • Semantic Entities: Entities provide a higher abstraction level of the document. They are used to reconcile the social results in order to maintain relevancy with the document. Similar to the keywords extraction services, the entities retrieved are ranked and contain outbound links to the matched entities on DBpedia, Wikipedia, Freebase, etc. A union is made between the results from Alchemy and Zemanta to ensure a wider coverage of entities. When a match is found, we merge the links from the two sources to ensure that we include all the resources that can be used to augment extra information about that entity in the document. • Categories: These are high-level taxonomies that can generally describe the document's content. A taxonomy is used to narrow down our query scope when targeting services like YouTube. In our Semantic Document model we define two possible category sets, one retrieved from Alchemy's Text Categorization API3 and the other retrieved from Zemanta Suggest API that follows the DMOZ categorization scheme4 .

2 3

<http://developer.zemanta.com/docs/suggest/> <http://www.alchemyapi.com/api/categ/categories.html> 4
<http://www.dmoz.org/desc/Top>

118

Chapter 8. Semantic Social News Aggregation

At the end of this process, we will have constructed the needed elements (keywords, entities and high level categories) wrapped in our Semantic Model to be passed to the query generator. For example, a summary of the Semantic Model for a web page titled "Turkey protests: Erdogan in 'nal' warning5 " looks like: 1. Categories: Culture Politics, Regional and Society 2. Keywords: Taksim Square, Protesters, Gezi Park, Mr Erdogan, Istanbul ... 3. Entities: Gezi Park, Recep Tayyip Erdogan, Taksim Square, Justice and Development Party (Turkey), Police of Turkey ...

8.2.2

Query Layer

In this component, the calls to the social services are made. SNARC uses the extracted keywords from the Semantic Document in order to construct the queries and disseminate them to the appropriate services. Figure 8.2 shows the different steps in order to retrieve a set of social results.

Figure 8.2: SNARC's Query Layer

1. Query Builder: Responsible for identifying targeted services and building tailored queries for each service. For example, if the processed document is categorized as a computer or technology related one, Stackoverflow service will

5

<http://www.bbc.co.uk/news/world-europe-22889060>

8.2. Underlying Mechanism

119

be targeted with the queries constructed. However, other categories will correspond to different services from the Stack Exchange websites6 . 2. Query Federator: Responsible for federating the queries identified in the previous step to the

corresponding services. To enhance performance, we tried to reduce the number of external calls. Yahoo Query Language (YQL)⁷ helped us in minimizing the number of calls and batching them into a single one. It is an expressive SQL-like language that lets you query, filter, and join data across Web services. However, we have found that we cannot fully rely on YQL due to their API calls limit and the restriction on the query execution time that is set to 30 seconds. To overcome this, we have implemented a fallback mechanism that federates the queries to the selected social services and groups the result to be passed afterwards to the parser. To further optimize the number of calls, we have decided to take the top two ranked keywords. We do not apply logical operator (AND/OR) in our queries; instead, we perform one-to-one mapping between each keyword and query. Indeed, we have found that gathering keywords even if semantically related might bring up noise in the results. However, as mentioned earlier, a part of the future work will be investigating the best method to construct the most relevant queryable entity using different logical operators.

3. Caching: The main setback in the query layer was the variable limited number of calls we can make to external APIs. To overcome this, we have implemented a simple cache mechanism that saves the results on disk up to an hour. There are several cache levels; the first is a URL level one where the results of the parsed queries are cached. For example, if a user visited a certain article on the CNN webpage the results might take up to 15 seconds to appear, whereas a second user visiting the same article minutes afterwards will have the cached results in few seconds. The second level is keyword and service specific. This can be very helpful as users generally browse articles of related topics or interests (semantic concepts), so for each user we can end up with the same high level concepts being requested frequently. An important thing to note is that the caching is done on the server side and is disk-based. The social services queried can be grouped as follows:

1. Multimedia Services: They include Slideshare, Vimeo and YouTube. Slideshare and YouTube allow the results to be fetched in a specific language that was detected in the previous step. In addition to that, YouTube search services are

67

<http://stackoverflow.com/sites> <http://developer.yahoo.com/yql/>

120

Chapter 8. Semantic Social News Aggregation

called twice; the first call is done to the YouTube V2 API⁸ where we specify in addition to the keywords a high level category to be targeted. To do so, we have manually created a category mapping file that maps the high-levels categories of Alchemys API and DMOZ to those provided by YouTube. The second call is done to YouTube V3 API⁹. The new feature provided by Google in this version is the ability to search using a semantic concept that corresponds to a Freebase concept ID; it proves to retrieve better results than the normal search. Freebase concept calls are cached for longer periods as they are less prone to changes.

2. Micro-posts Services: They include Twitter, Google+ and Stackoverflow. Language filtering is done where applicable.
3. General Search: This includes similar results found via Google search or those retrieved from the Zemanta API call. They are general articles or blog posts related to the current active page.

8.2.3

Data Parser

This is the last step where the results are unified and wrapped in a single social model. Figure 8.3 shows the different steps needed to produce the final parsed results that will be pushed back to the front-end.

Figure 8.3: SNARC's Data Parser

1. Live Reconciliator: Social (or folksonomic) tagging has become a trending method to describe, search and discover content on the web. Folksonomies empower users by giving them total freedom in choosing their categories and

89

<https://developers.google.com/youtube/2.0/> <https://developers.google.com/youtube/v3/>

8.2. Underlying Mechanism

121

keywords that they think describe best the content. This contrasts with taxonomies that over-impose hierarchical categorization of content [145]. However, in services like Twitter and Google+, tagging has been abused in a way that increased noise in the stream of results. To overcome this problem, we align the incoming stream of posts with the set of semantic concepts or keywords that describe the document. There are several approaches and tools like [77, 47, 124, 145] that aim at solving this problem. In SNARC we rely on two levels of reconciliation: one uses the high-level taxonomy (categories); and the other uses the vector of entities defined in the Semantic Document. For example, if SNARC wants to reconcile a blog post result retrieved from a general search, it constructs a Semantic Document Model for that result and applies the Cosine Similarity on the vector of ranked entities for each Semantic Model. Currently, we only reconcile against blog posts as it is very straightforward to construct a Semantic Document Model for them. However, an integral part of the future work will be the integration of SNARC's model to micro-posts and video search services.

2. Social Modeler: Every social network has its own underlying data model. To overcome this problem, we need to present the social results in a common wrapper. To do so, we have created an optimized universal social model that contains all the necessary data to model social information and can be reused in other projects. The model contains service related attributes (service name and type), author information (author's name and profile link) and general post information (title, thumbnail, link, embed code and time). Listing 8.1 shows the model for some of the social networks about an article for understanding critical CSS¹⁰.

```
[service]=>twitter[type]=>micropost[time]=>1 hour ago[title]=>RT @ProjectPeachUK: Need a practical guide to our HTML5 live web? Book your tour now via twitter or our live webapp http://t.co/DgtAB", 25)[link]=>641623665579270144[author]=>TheOnlyHTML5God[thumbnail]=>ht
```

Listing 8.1: Social Modeler output snippet for different social networks

3. Time Sorter and Results Shuer: To better display the results on the front-end, we unify the time representation and sort the results based on it.

10

<http://smashingmagazine.com/2015/08/understanding-critical-css/>

122

Chapter 8. Semantic Social News Aggregation

Afterwards we pick the top N results and shuffle them to generate a random order.

8.3

Front-End

SNARC is a service that generates a JSON file containing the results wrapped in our universal social model. As a possible front-end implementation, we have implemented a chrome extension that loads SNARC on any web page or application (see Figure 8.4). This implementation offers more flexibility to users by loading related social news anytime on any webpage or application. The results are visualized using a sliding panel on one of the screen edges, extracted entities are highlighted in the page itself and a short excerpt is displayed when hovering over them.

8.4

Summary

Aggregating relevant social news is not an easy task. SNARC performs the task in a nice and intuitive way that allows the user to discover what is happening instantly and without the need to navigate away from the current page. One of the important things to consider for the future is the integration of better reconciliation features and tools to ensure the display of relevant social posts. Moreover, real-time feature that can also push new related posts would be a great addition. Figure 8.4: SNARC's User Interface - The Google Chrome Extension

Conclusion of Part II

In this part, we presented the various parts required to enable Data Integration in the enterprise. First, we created an internal knowledge base by importing DBpedia into SAP HANA. On top of that, we built a set of services that enable entity disambiguation, semantic enrichment and schema matching. We presented RUBIX, a framework enabling mashup of potentially noisy enterprise and external data. The implementation is based on Open Rene and uses our entity disambiguation service to annotate data with rich types. As a result, the matching process of heterogeneous data sources is improved. We have also shown that it is possible to reveal what are the "important" properties of entities by reverse engineering the choices made by Google when creating knowledge graph panels and by comparing users preferences obtained from a user survey. Our motivation is to represent this choice explicitly, using the Fresnel vocabulary, so that any application could read this configuration file for deciding which properties of an entity is worth to visualize. Last, we cover the aspect of integrating external data coming from social media outlets. Data nowadays is spread over heterogeneous silos of archived and live data. People willingly share data on social media by posting news, views, presentations, pictures and videos. We presented SNARC, a service that uses semantic web technology and combines services available on the web to aggregate social news. SNARC brings live and archived information to the user that is directly related to his active page. Going back to our scenario, the proposed frameworks and services will allow our analyst Dan to be able to find and match various reports he is working on. Moreover, he will be able to augment extra measures and dimensions to his reports using DBpedia. In addition, Dan will be able to monitor relevant social feeds. This will allow him to either embed social snippets directly in his reports or to discover new information sources.

Chapter 9

Conclusions and Future Perspectives

In this chapter, we summarize the major achievements of this thesis and we give an outlook on future perspectives.

9.1

Scenario Flashback

Going back to our scenario defined in Section 1.2, we have defined two main personae. The first is a data analyst called Dan who works with the Ministry of Transport in France. He receives a memo from his management to create a report comparing the number of car accidents that occurred in France for this year, to its counterpart in the United Kingdom (UK). In addition, he is asked to highlight accidents related to illegal consumption of alcohol in both countries. The second is a data portal administrator called Paul. He is affiliated with the British Open Data portal (data.gov.uk). His daily job includes acquiring, preparing and publishing relevant datasets on the portal. He always strives to maintain a spam-free, high-quality portal.

9.2

Achievements

This thesis thoroughly describes the different steps aiming at realizing the vision of enabling self service data provisioning in the enterprise (see Figure 9.1). The work presented is beneficial to both our personae introduced. The contributions made are: Contributions for Data Portals Administrators Our data portal administrator Paul is always looking to expand his portals in terms of the number of datasets hosted, without compromising in their portal's data quality. In Chapter 3 (component B in Figure 9.1), we surveyed the landscape of various models and vocabularies that described datasets on the web. We found a shortcoming when it comes to having a complete descriptive dataset model taking into account access, license and provenance information. As a result, we proposed

9.2. Achievements

125

Figure 9.1: Annotated architecture diagram for enabling self-service data provisioning a Harmonized Dataset Model (HDL) that Paul will use as a basis to extend and present the datasets he controls. Paul now also knows what are the major dataset models out there, and what kind of metadata data owners need to fully represent their dataset. The mappings proposed in Section 3.3 will allow him to easily integrate data from various data management systems into his own. In Chapter 4 (component A in Figure 9.1), we proposed Roomba, an automatic dataset profiles generation and validation tool that can be easily extended to perform various profiling tasks. Out of the box, Paul can use Roomba to automatically x datasets metadata issues, and notify the datasets owners of the other issues to be manually fixed. In Chapter 5 (component C in Figure 9.1), we proposed a comprehensive objective quality framework applied to the Linked Open Data. Moreover, after surveying the landscape of existing data quality tools, we identified several gaps and the need for a comprehensive evaluation and assessment framework and specifically for measuring quality on the dataset level. As a result, we presented an extension of Roomba that covers 82% of the suggested datasets objective quality indicators. Paul will be able now to identify spam and low quality datasets. In addition to that, data available in his portal will now have rich semantic information attached to it. For example,

126

Chapter 9. Conclusions and Future Perspectives

temporal and spatial information extracted will be assigned into the corresponding fields in HDL. As an exemplary result, various datasets will be easily identifiable to cover various parts of the UK. Contributions for Data Analysts Our data analyst Dan believes that "more data beats better algorithms" and is always hunting for high quality data to produce accurate reports to the management team. By examining the rich datasets metadata presented in HDL he will be able to make fast decisions whether the dataset examined is suitable or not. He will also have vital information about the licensing and limitations for using this data internally. He will also have assurances on the dataset quality, which will help choose the best candidates out of ranked list. Dan will be able to have direct access to rich and high-quality dataset descriptions generated by Roomba. Moreover, the topical profilers in Roomba will be able to identify occurrences of alcohol related terms like "wine" in various datasets. Query expansion methods can be used to relate alcohol to wine allowing him to find the datasets he wants. In Chapter 7 (component D in Figure 9.1), we presented an entity disambiguation API built on top of SAP HANA. This API is used in RUBIX, a framework we proposed to enable mashup of potentially noisy enterprise and external data. Dan now has access to various datasets that he found matching his query to the portal administered by Paul. He will be also able to use the schema matching services to find and merge those datasets in his reports. Having imported those dataset into Lumira, he will be also able to use the internal knowledge base to apply various Figure 9.2: UI Prototype of semantic data semantic enrichments on this data. Figure 9.2 shows a possible integration of such service in SAP Lumira where Dan

9.3. Perspectives

127

is presented with a ranked set of properties retrieved by the algorithm proposed in Section 7.3. In Chapter 8 (component E in Figure 9.1), we proposed SNARC, a semantic social news aggregation service that allows the user to explore relevant news from internal or external sources. Dan is also a modern person, who is always trying to fresh information and believes in the wisdom of the crowd. Having SNARC services integrated with Lumira, he is also able to see a feed of relevant social media items that can be of interest to him. He actually follows a link in some tweet that he saw and was able to find relevant pieces of pointers that he would like to investigate further. In summary, the contributions above pave the way to build a set of smart services to enable analysts easily find relevant pieces of information and administrators fight spam and be able to maintain high quality data portals. The work presented in this thesis goes beyond the fact that attaching metadata to datasets is vital, but propose a set of services that can automatically achieve that in seamless manner.

9.3

Perspectives

This thesis could be extended in the following directions: • Data Profile Representation The proposed Harmonized Dataset Model (HDL) is currently available as a hierarchical JSON file. An enhancement would be to rene HDL and present it as a fully edged OWL ontology. In addition, HDL can be extended to propose also a set of enumerations as values to ensure a unified fine-grained representation of a dataset. Moreover, while we presented the mappings between various models in a table structure, presenting those mappings in a machine readable format will allow various tools like Roomba to use it. • Automatic Dataset Profiling It has been noticed that the issues surrounding metadata quality affect directly dataset search as data portals rely on such information to power their search index. There are various extensions to our tool Roomba that can help in automatically building and enhancing dataset profiles. An example of these extension would be the integration of statistical and topical profilers allowing the generation of full comprehensive profiles. We would also like to extend Roomba to

be able to run over other data portal types like DKAN or Socrata. This extension can be done by leveraging the data models mappings we proposed. In addition to all that, a possible enhancement will be ability to correct the rest of the metadata either automatically or through intuitive manually-driven interfaces.

128

Chapter 9. Conclusions and Future Perspectives

- **Objective Linked Data Quality** Ensuring data quality in Linked Open Data is a complex process as it consists of structured information supported by models, ontologies and vocabularies and contains queryable endpoints and links. In this thesis, we managed to narrow down the set of quality issues surrounding Linked Data to those who can be objectively measured and assessed by automatic tools. Our proposed tool covers 85% of the quality indicators proposed. A possible extension would be to integrate tools assessing models quality in addition to syntactic checkers with Roomba. This will provide a complete coverage of the proposed quality indicators. Moreover, there are currently no weights assigned to the quality indicators. A valid contribution would be to suggest weights to those indicators which will result in a more objective quality calculation process.
- **Enterprise Data Integration** A vital component to Data Integration in the enterprise is the existence of enterprise knowledge bases. Integrating additional linked open data sources of semantic types such as YAGO and evaluate our matching results against instance-based ontology alignment benchmarks such as OAEI1 or ISLab2 are possible future directions. Moreover, our work can be generalized to data classification. The same way the AMC helps identifying the best matches for two datasets, we plan to use it for identifying the best statistical classifiers for a sole dataset, based on normalized scores.

1 2

<http://oaei.ontologymatching.org/2011/instance/index.html> <http://islab.dico.unimi.it/iimb/>

Bibliography

[1] Ziawasch Abedjan, Tony Gruetze, Anja Jentzsch, and Felix Naumann. Proling and mining RDF data with ProLOD++. In 30th IEEE International Conference on Data Engineering (ICDE), pages 1198–1201, 2014. [2] Maribel Acosta, Amrapali Zaveri, Elena Simperl, and Dimitris Kontokostas. Crowdsourcing Linked Data quality assessment. In 12th International Semantic Web Conference (ISWC), 2013. [3] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets. In 2nd International Workshop on Linked Data on the Web (LDOW), 2009. [4] Rula Anisa and Amrapali Zaveri. Methodology for Assessment of Linked Data Quality. In 1st Workshop on Linked Data Quality (LDQ), 2014. [5] Ahmad Assaf, Eldad Louw, Aline Senart, Corentin Follenfant, Raphaël Troncy, e and David Trastour. RUBIX: a framework for improving data integration with linked data. In International Workshop on Open Data (WOD'12), pages 13–21, 2012. [6] Ahmad Assaf and Aline Senart. Data Quality Principles in the Semantic Web. In 6th International Conference on Semantic Computing ICSC '12, 2012. [7] Ahmad Assaf, Aline Senart, and Raphaël Troncy. SNARC - An Approach e for Aggregating and Recommending Contextualized Social Content. In The Semantic Web: ESWC 2013 Satellite Events, Revised Selected Papers, pages 319–326, 2013. [8] Ahmad Assaf, Aline Senart, and Raphaël Troncy. Roomba: Automatic Valie dation, Correction and Generation of Dataset Metadata. In 24th World Wide Web Conference (WWW'14), Demos Track, Florence, Italy, 2015. [9] Ahmad Assaf, Raphaël Troncy, and Aline Senart. An Objective Assessment e Framework & Tool for Linked Data Quality - Enriching Dataset Proles with Quality Indicators (Major Revision). International Journal on Semantic Web and Information Systems (IJSWIS), 2015. [10] Ahmad Assaf, Raphaël Troncy, and Aline Senart. HDL-Towards a Harmoe nized Dataset Model for Open Data Portals. In 2nd International Workshop on Dataset PROFiling & fEderated Search for Linked Data, Portoroz, Slovenia, 2015.

130

Bibliography