LinksB2N: Automatic Data Integration for the Semantic Web

Manuel Salvadores, Gianluca Correndo, Bene Rodriguez-Castro,
Nicholas Gibbins, John Darlington, and Nigel R Shadbolt

Intelligence, Agents, Multimedia (IAM) Group
School of Electronics and Computer Science
University of Southampton, UK
{ms8,gc3,nmg,b.rodriguez,jd,nrs}@ecs.soton.ac.uk

Abstract. The ongoing trend towards open data embraced by the Semantic Web has started to produce a large number of data sources. These data sources are published using RDF vocabularies, and it is possible to navigate throughout the data due to their graph topology. This paper presents LinksB2N, an algorithm for discovering information overlaps in RDF data repositories and performing data integration with no human intervention over data sets that partially share the same domain.
LinksB2N identifies equivalent RDF resources from different data sets with several degrees of confidence. The algorithm relies on a novel approach that uses clustering techniques to analyze the distribution of unique objects that contain overlapping information in different data graphs. Our contribution is illustrated in the context of the Market Blended Insight project[1] by applying the LinksB2N algorithm to data sets in the order of hundreds of millions of RDF triples containing relevant information in the domain of business to business (B2B) marketing analysis.

1 Introduction

Despite the progress made by the Semantic Web automated data integration is not yet a reality and the number of data sources continues to rise. Mappings between those data sources is a time-consuming task that requires specific software, designed and developed for each data integration scenario. The granularity and domain of the growing number of public data sources that populate the Web tend to vary. It is likely that two different data sources from different providers but from related domains will contain information that partially overlaps. It is also very likely that this information is represented differently in each data source. The data sources that are the subject of study in our research are made of RDF[2] graphs and the information they contain is captured in RDF triples.

[1] The Market Blended Insight project (DTI Project No: TP/5/DAT/6/I/H0410D) is funded under the UK Governments Technology Programme.

[2] http://www.w3.org/rdf/ (accessed on 04/2009)

This paper presents a novel mechanism to automatically identify overlapping information between multiple data sources. Our approach navigates the RDF data sources discovering the RDF properties that can express whether a specific RDF resource is unique and using clustering techniques discovers equivalences in other data sources.

The remaining sections of the paper are structured as follows. The next section introduces the project and the use case that motivated this research. Section 3 presents an overview of existing techniques that were considered but that did not fully meet the requirements of our problem. Sections 4 and 5 covers the algorithm that solved our problem in detail and finally, Section 6 summarizes the conclusions and possible future directions for this work.

2 Motivation

Market Blended Insight (MBI) is a project with a clear objective of making a significant performance improvement in UK business to business (B2B) marketing activities in the 5-7 year time frame. A thorough overview of the project, including the technologies involved to achieve its goals is presented in [11].

A challenge that stands out is the integration of business data sources using Semantic Web technologies. The problem of integrating a series of data sources is usually approached by integrating them in pairs. Every pair presents its own integration challenges that typically are solved with a custom solution for that specific combination. In a worst-case scenario there might be as many custom solutions as different pair-wise combinations of data sources.

In the case of the MBI project, data sources with a variety of provenance were integrated in a scenario that presented the following characteristics:

(a) Same concepts are represented with different structures (i.e. different format for addresses).
(b) The data was incomplete (i.e. incomplete addresses, missing postal codes).
(c) Data from different sources contained variations of the same literal and errors.
(d) There are no shared key values that crossed the data sets.
(e) Synonyms or vernacular variations are widely used in different data sets (i.e. “Earnest Smith & Son” versus “E. Smith and Sons”).

As it is demonstrated along the rest of the paper LinksB2N filters these characteristics and creates the necessary links between different RDF graphs to perform queries that join RDF statements from different sources.

3 State of the art

Many approaches to data integration have been proposed in computer science during the years. Many issues related to data integration have been successfully solved using web related technologies: XML for solving syntactic heterogeneities;

RDF for defining a common data description language and, finally, OWL to provide primitives for describing powerful data models.

On a high level, the architecture of distributed information systems is based on a mediator based architecture, called also Virtual Integration Architecture, described in [13] where ad-hoc data source wrappers reconcile schema heterogeneities into a common data model. Some examples of systems that implemented distributed data systems based on the mentioned approach are: SIMS [2], OBSERVER [8] and KRAFT [10].

Semantic web community has devoted much effort to solve the problem of mediating heterogeneous schema in order to create more flexible and general data wrappers. In ontology mapping [6] the challenge is to discover automatically alignments between entities described in different ontologies exploiting lexical similarities, lattice structure or instance classification learning techniques.

Schema reconciliation is an important issue in data integration. However, it is not enough to integrate heterogeneous data sources on the web, a data space that normally has a great redundancy of entity representations. An important aspect of data integration is the capability to integrate information about same entities described in different data sets. This problem is known within database community as the Record Linkage [9] task and it is adopted when there is necessity to join data sets that do not have a unique database key in common (i.e. national insurance number or passport number). In semantic web, due to the central nature of URIs for describing resources, there is a similar issue, named the problem of coreference [5], of discovering where two or more distinct URIs are used for a single non-information resource.

The problem of finding such connections and maintaining referential integrity [1] is increasingly important because of the nascent Web of Data. In fact, the Web it is likely to be flooded with structured information where data sets are likely to be overlapping since there are no defined authorities to solve such issues. Data publisher collaboration to mediate ontologies and share resources can provide an effective added value to data integration communities of practice [3], but many issues must be tackled both on the schema and entity identity level.

Clear evidence of the prominence of the task of record linkage in the Semantic Web can be found looking at the activities of the Linked Data community which aim to foster links between entities. The Linked Data community is a rapidly growing community of organizations who are using the Web as a means to share and integrate structured information. Providing linkages between entities from a growing collection of datasets is proving to be a challenging task that must scale to the Web.

The degree of automation of such a task, that cannot be reasonably fulfilled completely by hand, will heavily affect future integration of data sources. Consequently some general procedure that does not heavily involve users in discovering such links must be founded. Knowing if a common key exists between two collections of data could greatly improve the chances of discovering overlapping entities. The adoption of standard naming or identification schemes can help engineers to create batch procedures that encode heuristics for discovering instances uniqueness exploiting patterns that properties should follow. This is the approach followed by tools like Silk [12] that allows user to explicitly specify similarity criteria between instance data properties and then aggregate such results in a single value of confidence about the uniqueness of two instances. Although more general than ad-hoc approaches, methods like Silk requires that the two schema are known in advance, and the link discovery must be supported by a heuristic that is given explicitly by users.

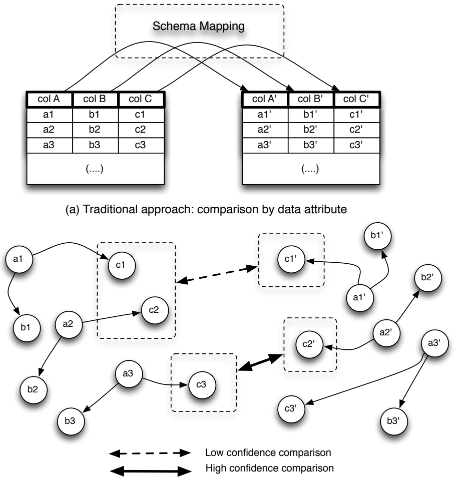
However encoding such heuristic can be sometime problematic; the schema may not be well known by the users or the data may be too noisy to apply algorithmic procedures. In this case statistical approaches are more suitable, due to their robustness to noise and lesser involvement of user feedback in the procedure. Such methods try to mine huge amount of data in order to discover record similarities that could end up in entity equivalence relationships.

The method here described does not rely on information from schema or schema comparison, therefore it is quite suitable for automatic discovery of RDF coreferences. This very feature distinguishes LinksB2N approach to other approaches in semantic web like Silk [12] or previous approaches in database literature [4]. Furthermore, schema comparison could be also unreliable since the authors of an ontology could be unavailable or just be different from those who adopted the ontology for publishing the data. That stresses even more the demand of data driven procedures that can be of support to users willing to import and exploit web accessible data.

4 Algorithm Overview

The proposed solution for discovering overlapping information in different RDF graphs is the LinksB2N algorithm. The algorithm is based on the idea that "The unique combination of RDF predicates associated with RDF resources is what defines their existence as unique". From this simple concept the algorithm identifies the RDF predicates that makes each RDF resource unique in such a way as to be used to find its equivalent resource(s) in other data sets.

Traditional approaches follow heuristics that are constrained by entity relationship models (see Figure 1.a), these solutions look at the table records and based on context settings find equivalences between different data-sources. The record linkage accuracy for these solutions is high when is known the pieces of information that lead the matching between entities. However in the Semantic Web due to the variety of data is highly complex to know which data attributes are the ones to be used when linking entities. The problem can scale in complexity if not always the same piece of information is located in the same data attribute. For instance, addresses are most of the times formatted as *address line 1*, *address line 2*, ... and it is never clear where to find: *the building name*, *the house number* or *the flat number*. LinksB2N uses a different approach and ranks the RDF properties based on their uniqueness performing cross comparisons not based on a user predefined settings but on a statistical approach (see Figure 1.b).



(a) Traditional approach: comparison by data attribute

(b) LinkB2N: Cluster Based Comparison

Fig. 1. Method Comparisons

Part of LinksB2N fundamentals is finding implicit links between graphs by:

- Identifying RDF predicates with dispersed values where the level of dispersion is measured through data clustering techniques.
- Automatically recognizing the RDF predicates that are expressing the same information in order to compare them together.
- Comparing RDF literals and associating them a confidence factor for each positive comparison.

The novelty of this work is the use of the distribution of RDF literals to find mappings between RDF resources that belong to different RDF graphs. This approach provides a flexible mechanism where the logic to link data across repositories is not dependent on the sources being integrated but based on a system driven by statistical decisions.

Traditional approaches tend first to apply schema mapping algorithms, and second to perform equivalences as row based comparisons. This solution performs reasonably well when the underlying data topologies share a minimal structure, as it happens with relational databases. Relational databases are table-relationship based structures and therefore data integrations can be based on a shared topology of tables, rows, columns and relationships. On the other hand RDF is not tied to any particular schema, and provides the flexibility to describe data in arbitrary graph structures which means that well-established solutions for data integration cannot be applied.

To resolve this problem, LinksB2N relies on RDF graph navigability and analyzes the data at the instance level. The algorithm evaluates each RDF property

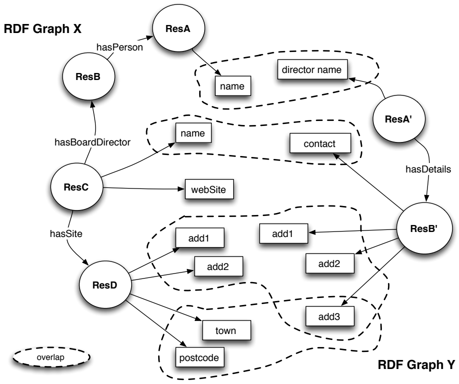


Fig. 2. RDF Graph Overlap Example

linked to a typed RDF resource[3] and clusters its values based on their similarity. Using these clusters, the algorithm finds suitable RDF predicates for guiding the comparison between the graphs. Figure 2 illustrates an example of an RDF graph overlap where resources A,B,C and D from graph X overlap with A' and B' from graph Y.

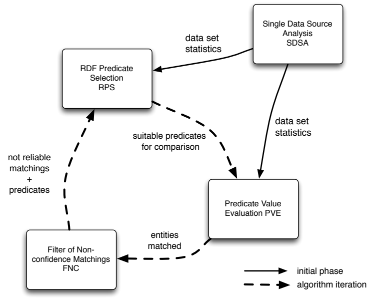

Single Data Source Analysis SDSA
data set statistics
RDF Predicate Selection RPS
data set statistics
suitable predicates for comparison
not reliable matchings + predicates
Predicate Value Evaluation PVE
entities matched
Filter of Non-confidence Matchings FNC
initial phase
algorithm iteration

Fig. 3. Algorithm phases

As designed, the algorithm works in four phases (see Figure 3):

Single Data Source Analysis (SDSA): The first phase collects graph statistics independently and creates clusters of similar values for each RDF predicate. The SDSA phase navigates the graphs using SPARQL[4].

[3] Throughout the paper, an RDF resource is considered *typed* when it holds at least one *rdf:type* predicate.

[4] http://www.w3.org/tr/rdf-sparql-query/ (accessed on 04/2009)

RDF Predicate selection (RPS): Using the clusters produces at the SDSA phase as input, the RPS phase generates pairs of RDF predicates to be compared. At this phase the pairs that are connected together meet a minimum threshold of similar values. This minimum threshold gets opened (decreased) as the algorithm iterates from the FNC phase.

Predicate Value Evaluation (PVE): For each pair of RDF predicates the PVE phase evaluates equivalence between the RDF objects and attaches a confidence ratio for each RDF resource matched.

Filter of Non-confidence Matchings (FNC): The FNC phase filters the linked RDF resources by applying another iteration of the RPS phase for those that do not fulfill a minimum confidence. The iteration is performed from the RPS phase and opens the selection of predicates to include more matchings between instances.

The algorithm terminates when no more predicates can be selected at the RPS phase or, all the entities have been matched above the confidence threshold.

5 Algorithm Phase by Phase

This section details each of the phases and illustrates the explanations with examples from data sets where LinksB2N has been applied. The presented scenario studies links between two data sets, A and B, that share the Marketing domain.

Data Set A: Holds data on all the companies in the UK[5] (3.5 million entities). This data contains two types of RDF resources: Sites and Organizations; which are described in Figure 4.

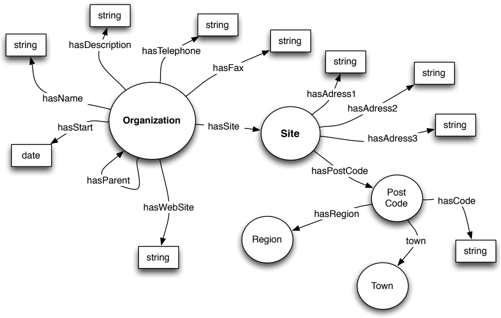


Fig. 4. Data Set A Schema

Data Set B: Holds information from some of the MBI project partners, containing names, addresses and other internal data in spreadsheet-like data structures (see Figure 5)

[5] The provider of data set A is Dun & Bradstreet http://www.dnb.co.uk/

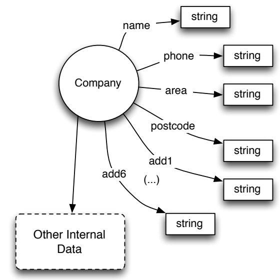


Fig. 5. Data Set B Schema

The data sets are populated with real company names and contact data as well as addresses, number of employees and other internal information. Even though the data sets A and B share the same conceptual entities, they are described with different structures and contain incomplete literals with errors and transformations as described in section 2. This integration scenario meets the level of complexity that LinksB2N is expected to solve:

5.1 Single Data Source Analysis (SDSA)

The SDSA phase creates clusters of similar RDF objects, propagating the cluster statistics to the RDF predicates to which they are bound. The analysis is processed independently for each RDF graph. LinksB2N takes as inputs the remote SPARQL and by using SPARQL standard as interface all the RDF data sets published on the Web are potential evaluation tests.

The SDSA phase itself it is decomposed on three sub-phases:

(1) RDF Types Detection: The first step navigates the graphs constraining a SPARQL query to the pattern {*?uri rdf:type ?type*} (lines 2 and 3). The algorithm traverses the result of the latest query retrieving all the predicates and objects for each of the RDF types (lines from 4 to 7). The final output is indexed through the *index* function (lines 8 and 9):

```
1: function SDSA(graph,threshold)
2:   r=query("SELECT ?uri ?type WHERE {
3:            GRAPH <graph> { ?uri rdf:type ?type }}" )
4:   for each uri, type in r:
5:     r_i=query("SELECT ?pred ?obj WHERE {
7:                GRAPH <graph_data> { <uri> ?pred ?obj }}")
8:     for each pred, obj in r_i:
9:        index(uri,type,pred,obj,graph,threshold)
```

(2) RDF Predicate Index The *index* function invokes the creation of the cluster (line 5) but previously registers counters for: (a) number of instances per RDF type (line 2) and (b) number of bounded RDF predicates per RDF type (line 3). These counters together with the clusters are the main inputs to the uniqueness function developed in Section 5.2:

```
1: function index(uri,type,pred,obj,graph,threshold)
2:   graph_level[type] += 1
3:   type_level[type][predicate] += 1
4:
5:   process_cluster(uri,pred,obj,graph_data,threshold)
```

(3) Cluster Creation: This function creates clusters of similar values based on the Levenshtein distance [7]. In order to make this possible, firstly are collected all the RDF statements with same RDF type and RDF predicate as the ones from the input to the *process_cluster* function (lines 2 and 3). Secondly, the output of the latest query is traverse in order to compare the distance between the input object *obj* and the potentially equivalent variable *?similar* (lines 5 to 7):

```
1: function process_cluster(uri,type,pred,obj,graph,threshold)
2:   r=query("SELECT ?s ?p ?similar WHERE { GRAPH <graph> {
3:           ?s rdf:type <type> . ?s <pred> ?similar. }}")
4:
5:   for each s, p, similar in r:
6:      if distance(obj,similar) > threshold:
7:         predicate_level[type][uri][pred] += 1
```

The above steps indicate how the data is navigated and analyzed using Semantic Web standards. The data produced on these steps is classified into three levels, each of them provides insight statistics for: (a) RDF graphs, (b) RDF types and (c) RDF predicates. These levels of information together with sample outputs are in-depth described below:

RDF Graph Level: This level gives the highest description level representing the RDF types together with the number of instances per type within each RDF graph. The following structure represents some of the data generated at this level for Data Sets A and B:

```
graph       rdf:type            instances
--------    ----------------    ---------
datasetA    nsA:Organisation    3 164 092
            nsA:Site            2 789 996
datasetB    nsB:Company             4 000
```

RDF Type Level: This level holds the distinct RDF predicates together with the number of RDF resources bound to them. Moreover, this information is grouped by the RDF type where the RDF predicate projects its presence. The following structure shows data samples generated at the RDF type level:

```
rdf:type            rdf:property     bounded     clusters
----------------    ------------     --------    ----------
nsA:Organisation    nsA:hasWebSite    145 020       126 000
                    nsA:hasName     3 164 092     2 100 157
                       ( ... )
```

RDF Predicate Level: Keeps the cluster information that will be used to measure the degree of confidence for each equivalence. The following structure shows samples of data for this level[6]:

```
rdf:type            rdf:property     rdf:res     rdf:literal      clt_sz
----------------    ------------     --------    -------------    -------
nsA:Organisation    nsA:hasName      nsA:comp1   "Comp X Ltd."        120
                                     nsA:comp2       "Comp Z "          5
                                 ( ... )
```

Partial statistics produced for Data Set A and Data Set B are presented in Figure 6. The chart shows the number of RDF objects (bar in black) and Clusters (bar in grey) for a subset of data attributes.

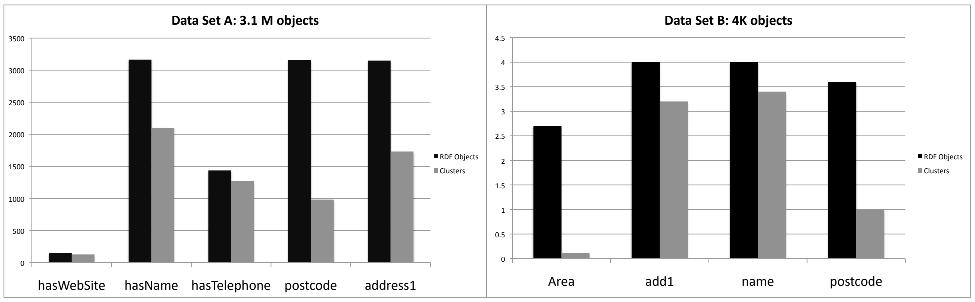


Fig. 6. RDF objects vs Clusters in Data Sets A and B

It is important to state that clusters of one individual are considered also clusters. Therefore, the more clusters the more unique the property is because it means that all the values are gathered on numerous small clusters. On the other hand, if the number of clusters then all the values are collected into the same cluster and therefore the property is low rated on uniqueness. From this logic we can assume that properties that represent highly unique information are the one containing "almost" as many clusters as RDF objects.

The latest explanation lead us to interpret how these statistics can be used for finding information overlaps. As design LinksB2N seeks for highly populated and

[6] *clt_sz* (fifth column) stands for cluster size and represents the number of similar literals

very unique RDF properties to conduct the matching algorithm. And looking at the chart in Figure 6 we can discover a trade-off between selecting RDF properties according to these criteria. For instance, in Data Set A the "more unique" attribute is *hasWebSite* whereas, it is at same time the least populated.

The next Section (5.2) covers in-depth how LinksB2N models the selection of the most suitable RDF predicates.

5.2 RDF Predicate Selection (RPS)

The RPS phase is where the RDF predicates from different RDF graphs are selected to compare their bound RDF literals. This phase queries the statistics produced at the SDSA phase and analyzes the uniqueness of RDF predicates based on two main variables: the proportion of bound objects as *PBO(type, predicate)* and the uniqueness of the RDF predicate as *U(type, predicate)*. These functions are modeled as follows:

$$PBO(type, predicate) = \frac{bound\_objects(type, predicate)}{objects(type)}$$

$$U(type, predicate) = \frac{clusters(type, predicate)}{bound\_objects(type, predicate)}$$

Where:

bound_objects(type,predicate) Is the number of objects bound to an specific *type* and *predicate*.
objects(type) Is the number of RDF resources typed as *type*.
clusters(type,predicate) Is the number of clusters created in SDSA phase for an specific *type* and *predicate*.

$U(type, predicate)$ will return values close to 1 for properties which have "almost" as many clusters as bound values [7]. On the other hand, the values close to 0 will represent properties with low numbers of highly populated clusters (attributes with a low degree of uniqueness) which are less suitable for finding equivalences. Together with $U(type, predicate)$ the level of population of an RDF predicate is measure through the PBO function. This function quantifies between 0 and 1 the proportion of bound RDF predicates for an RDF type. Figure 7 shows the predicate selection strategy in which, for the first algorithm iterations the RPS phase focusses on the RDF predicates that represent more unique attributes and as the algorithm iterates then attributes that represent less unique information are selected.

that do not represent unique attributes are included for the cross comparison.

The final RPS indicator for a given RDF predicate is measure as the average U and PBO average[8]:

[7] Assuming that, the more clusters the less populated they are, those properties will hold unique RDF literals. This argument is covered at the end of Section 5.1

[8] t and p in the RPS function stands for *type* and *predicate* respectively.

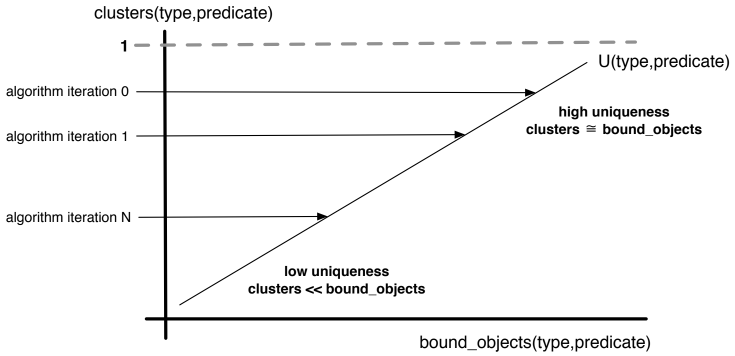


Fig. 7. RDF Predicate Selection (RPS) Strategy

$$RPS(t, p) = (U(t, p) + PBO(t, p))/2$$

For the sample data sets A and B the ranking produced by the RPS is presented in Figure 8 which shows the top RDF predicates from each RDF graph. It can be appreciated how RDF predicates with the same meaning differ on their statistics. For instance, the post codes in Data Set A provide a poor uniqueness factor of 0.31, while in Data Set B for the same attribute is 0.93. The same happens between the predicates *hasName* and *name* from A and B respectively, this gives an insight into the variation between different data sets from the same domain.

To this point, the RPS phase has identified the RDF predicates suitable for comparison, which make one RDF resource different from another. The next step is to match together the RDF predicates from different RDF graphs that hold similar RDF literals. This matching is achieved by stating a minimum threshold for equivalent RDF literals between the RDF predicates. This implies intersecting the RDF literals for each pair of RDF predicates, by applying this technique it is possible to find small data overlapping evidences. These evidences will put together RDF predicates for further sophisticated comparisons in PVE phase. Table 1 represents the pairs of RDF predicates from data set A ($preds_A$) and B ($preds_B$) with an equivalence threshold of 5%.

One of the advantages of LinksB2N gets uncovered at this point as it is shown how cross comparisons are performed between data attributes. Data attributes that apparently would not provide any matchings to disambiguate entities from the user perspective. For instance, in Table 1 the algorithm has matched together *hasName* with *add1* and *address1* with *add2*.

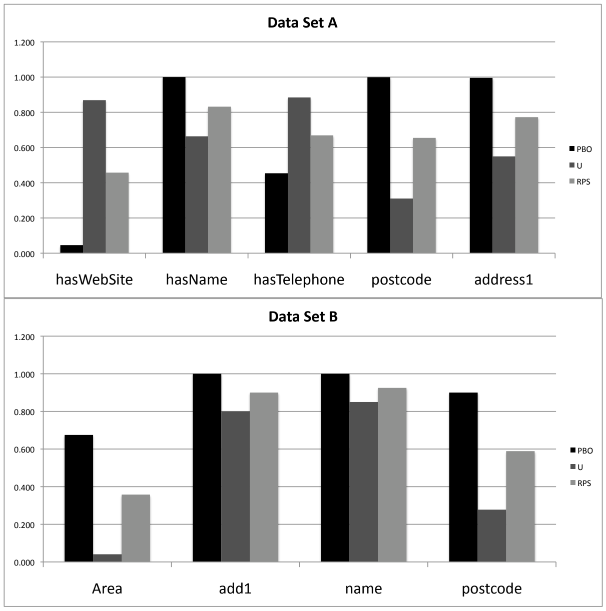


Fig. 8. Functions PBO , U and RPS for Data Sets A and B

Table 1. RDF Predicate Matching with 5% threshold

$preds_A$	$preds_B$	common values
address1	add2	1 559
address1	add1	3 035
postcode	postcode	3 692
hasName	name	1 050
hasName	add1	344

5.3 Predicate Value Evaluation (PVE)

The PVE phase is where the comparison between data sets is performed and receives as an input a list of predicates pairs $(preds_A, preds_B)$ (see table 1) where from the values will be compared in four different phases:

(1) The *equal* comparison is applied between the original RDF literals.
(2) The RDF literals are transformed to upper case and filtered to only alpha numeric characters, and the *equal* comparison is applied to them.
(3) Distance based comparisons are applied between the original RDF literals.
(4) The transformation to the RDF literals made in (2) is applied, and distance based comparisons are processed.

When any of the comparison steps (1) to (4) successfully matches two RDF literals consequent steps are not evaluated, this logic allows to significantly reduce the number of comparisons to be performed by the steps (3) and (4).

For each positive comparison a confidence factor is attached. This confidence factor is calculated from three variables: the size of clusters (clt_sz) where the compared literals ($preds_A, preds_B$) are located, the step where the comparison is performed ($step$) and the distance (d) between the RDF literals.

The confidence factor is calculated as a function where the maximum is given by $d(v_1, v_2, step)$ that gets penalized (divided) by the average size of the clusters where v_1 and v_2 are located. The step in which the comparison is evaluated is also taken into account and divides the whole function:

$$conf(v_1, v_2, step) = \frac{d(v_1, v_2, step)}{\frac{clt\_sz(v_1) + clt\_sz(v_2)}{2/step}} = \frac{(2/step) * d(v_1, v_2, step)}{clt\_sz(v_1) + clt\_sz(v_2)}$$

Where:

$$d(v_1, v_2, step) = \begin{cases} 1 \wedge v_1 = v_2 \wedge (step < 3) \\ \vee \\ levenshtein(v_1, v_2) \wedge (step > 2) \end{cases}$$

$$\forall\ v_1, v_2, step \rightarrow conf(v_1, v_2, step) \in (0, 1]$$

The function $conf(v_1, v_2, step)$ models the confidence producing values close to 1 when the matching is highly reliable which implies: $clt\_sz(v_1) \simeq clt\_sz(v_2) \simeq 1$ and $step = 1$. The function $levenshtein(v_1, v_2)$ returns the similarity of the literals within [0,1][9] therefore $conf(v_1, v_2, step)$ will return lower values (closer to 0) for the steps in which distance based comparisons are performed.

For instance, assuming the comparison v_1=“Cancer Research UK” from Data Set A and v_2=“The Cancer Research Ltd.” from Data Set B and:

- $clt\_size(v_1)$=169 because in Data Set A all the Cancer Research institutions are registered with similar names.
- $clt\_size(v_2)$=3 in Data Set B there are only 3 entities similar to v_2.
- $d(v_1, v_2, step)$=0 for comparison steps 1 and 2 because obviously $v_1 \neq v_2$
- $d(v_1, v_2, step = 3)$=0.8 result comingfrom applying the $levenshtein(v_1, v_2)$.

Therefore the confident ratio is calculated as follows :

$$conf(v_1, v_2, step) = \frac{(2/3) * 0.8}{(169 + 3)} = 0.007$$

The selected example produces a very low confidence factor and shows how the size of the clusters penalize the confidence and that even though the comparison distance (0.8) can be considered high it is not significant enough to raise a positive matching. Furthermore, it is important to notice the granularity of the algorithm since the confidence it is not calculated per RDF property (i.e. name or postcode) but by the presence of similar values within each RDF property. This means that for the studied case the confidence ratio is very low but for other values within the same data attribute the confidence will be higher if the clusters of similar values are smaller or if without being small the comparison is positive on steps 1 or 2.

[9] $levenshtein(v_1, v_2)$ returns factors close to 1 when the literals are similar and close to 0 for the opposite.

Table 2. RDF Literals Matched by Step

$preds_A$ & $preds_B$	(1)	(2)	(3)	(4)
address1 & add2	0.9k 0.94	0.6k 0.77	0.2k 0.33	0 0.0
address1 & add1	1.2k 0.97	1.8k 0.80	0.3k 0.42	0.1k 0.12
postcode & postcode	3.4k 0.68	0.2k 0.40	0 0.0	0 0.0
hasName & name	0.7k 0.65	0.3k 0.50	1.1k 0.38	0.4k 0.21
hasName & add1	0.2 0.98	0.1 0.86	0.8k 0.53	0.1k 0.13

For Data Sets A and B Table 2 represents the RDF literals that has been matched at this phase and shows the average confidence ratio[10]. The links and confidence ratios from table 2 illustrate how RDF predicates that at first place could be chosen as highly reliable are not. For instance the comparison between names from both RDF graphs produces a high number of matchings but with poor average confidence ratio. This effect is produced due to the high number of companies with the same name, it happens with franchises existing several clusters that contains even more than 150 companies with similar names.

The PVE phase produces its output as RDF triples in Turtle[11] format. The output links the RDF resources from both the source and target data sets attaching the calculated confidence ratio:

```
@prefix : <http://users.ecs.soton.ac.uk/ms8/linksB2N#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix datasetA: <http://users.ecs.soton.ac.uk/ms8/linksB2N/datasetA#> .
@prefix datasetB: <http://users.ecs.soton.ac.uk/ms8/linksB2N/datasetB#> .

_:match0001 a :ConfidentMatch;
        :hasConfidence "0.007"^^xsd:float;
        :hasSource datasetA:org002254;
        :hasSourcePredicate datasetA:hasName;
        :hasTarget datasetB:comp000001;
        :hasTargetPredicate datasetB:name
    .
```

The RDF triples generated at this phase act as then main input for the FNC phase which is further explained in the next section.

[10] In table 2 the top number in the cells stands for number of RDF literal matching in 10^3 order and the bottom indicator stands for the average confidence ratio.

[11] http://www.w3.org/TeamSubmission/turtle/ (accessed on 06/2009)

5.4 Filter of Non-confidence Matchings (FNC)

The FNC phase navigates the RDF graphs for all the links matched in PVE phase. The previous phase highlighted similar RDF literals bound through equivalent RDF predicates in different RDF graphs and outputs the results as RDF data. The FNC phase, relying on that, analyzes the topology of the RDF graphs in order to find out the RDF resources that participate in the overlap.

In this phase the main goal is to put together the matchings that refer to the same instances. Figure 9 represents one single matching and shows how at this point two instances *OrgX* and *CompZ* are connected by the PVE output.

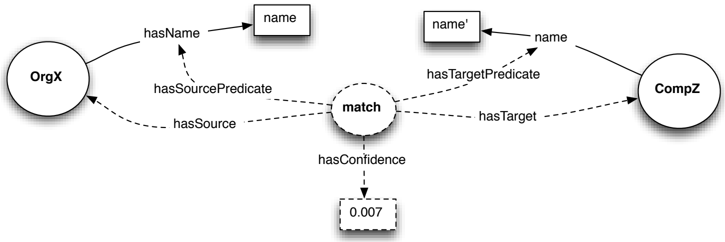


Fig. 9. PVE Single Matching Representation

Nevertheless, Figure 9 shows a local comparison and this view needs to be scaled up to all the matchings that provide additional information about the same mapping.

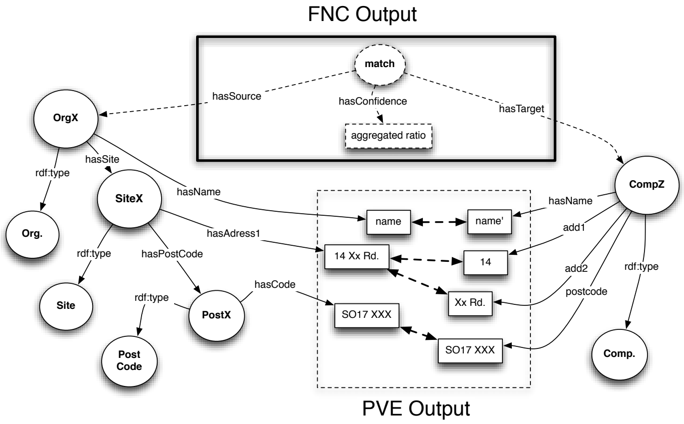


Fig. 10. FNC Output Representation

By integrating the PVE output and the original data sets it is possible to draw a graph that connects resources from data sets A and B. The output of this phase are link propositions with an aggregated confidence equivalent to the sum of all the confidence rations that connect two RDF resources. In that sense, Figure 10 shows all the matchings between *OrgX* and *CompZ* together with matchings between their connected entities.

5.5 Results Overview

In the context of the MBI project, for the data sets A and B, LinksB2N discovered overlaps between Sites and Organisations from A to Companies from B. LinksB2N found that 96 % of the RDF resources in Data Set B were represented in Data Set A, holding a confidence ratio above 0.9 in the third iteration. During the second and third iteration not only the RDF predicates from Table 1 where analyzed but also (*hasAddress2*, *hasAddress3*, *region*) from A and (*phone*, *add3*, *add4*) from B were also included.

LinksB2N flexibility was shown as we discovered that mappings increased from 65%[12] to 74% by adding automatically *hasAddress2* and *phone* from A and B respectively in the second iteration. The third iteration included (*hasAddress3*, *region*) from A and (*add3*, *add4*) from B which increase the overlapping information to 96%.

Besides the overlapping information found, due to the crossed comparison of RDF predicates other evidences were discovered. For instance, some overlaps (2%) existed between *hasTelephone* from A and *add4* from B, which demonstrates how data can contained uncontrolled errors and LinksB2N gets adapted to them and finds unexpected matchings. Other case is that for 17% of the overlaps *region* was overlapped with either *add2*, *add3* or *add4*, this shows how LinksB2N searches for data overlap evidences over a larger set of RDF predicates in order to meet a minimum confidence threshold.

6 CONCLUSIONS

Part of the activities of the MBI project involves the integration of disparate data repositories published in the format of RDF triples. Existing solutions did not meet the needs of the project due to (1) poor performance finding overlapping information when data structure varies and contains errors; and (2) incapability for navigating RDF graphs when the schema is not known. To solve this problem in an automated fashion, we have introduced the LinksB2N algorithm. Throughout the paper it has been shown how LinksB2N (a) identifies overlapping information across data sources, and (b) characterizes these equivalences with a confidence factor; with no user input and without pre-established configuration of the data. These results can support applications to perform

[12] The percentages expressed on this paragraph are the number of RDF resources from Data Set B contained in at least in one RDF graph overlap.

both instance data integration and schema data mapping. Using LinksB2N, the business data pertaining to the MBI project has been successfully integrated and opportunities for improvement have considered include: (a) providing tools to exploit LinksB2N output for data integration scenarios, and (b) developing custom normalisations for well-known terms.

References

1. Harith Alani, Srinandan Dasmahapatra, Nicholas Gibbins, Hugh Glaser, Steve Harris, Yannis Kalfoglou, Kieron O'Hara, and Nigel Shadbolt. Managing reference: Ensuring referential integrity of ontologies for the semantic web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, pages 317–334. Lecture Notes in Artificial Intelligence, 2002.
2. Yigal Arens and Craig A. Knoblock. Sims: Retrieving and integrating information from multiple sources. In *SIGMOD Conference*, pages 562–563, 1993.
3. Gianluca Correndo and Harith Alani. Collaborative support for community data sharing. In *The* 2^{nd} *Workshop on Collective Intelligence in Semantic Web and Social Networks*, December 2008.
4. Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
5. Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri identity management for semantic web data integration and linkage. In 3^{rd} *International Workshop On Scalable Semantic Web Knowledge Base Systems*. Springer, 2007.
6. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.
7. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.
8. Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, and Amit P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.
9. Howard B. Newcombe and James M. Kennedy. Record linkage: making maximum use of the discriminating power of identifying information. *Commun. ACM*, 5(11):563–566, 1962.
10. Alun D. Preece, Kit ying Hui, W. A. Gray, Philippe Marti, Trevor J. M. Bench-Capon, Dean M. Jones, and Zhan Cui. The kraft architecture for knowledge fusion and transformation. *Knowl.-Based Syst.*, 13(2-3):113–120, 2000.
11. Manuel Salvadores, Landong Zuo, S. M. Hazzaz Imtiaz, John Darlington, Nicholas Gibbins, Nigel Shadbolt, and James Dobree. Market blended insight: Modeling propensity to buy with the semantic web. In *International Semantic Web Conference*, pages 777–789, 2008.
12. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In 18^{th} *International World Wide Web Conference 2009*.
13. Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.