Creation, Population and Preprocessing of Experimental Data Sets for Evaluation of Applications for the Semantic Web

György Frivolt, Ján Suchal, Richard Veselý,
Peter Vojtek, Oto Vozár, and Mária Bieliková

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
`Name.Surname@fiit.stuba.sk`

Abstract. In this paper we describe the process of experimental ontology data set creation. Such a semantically enhanced data set is needed in experimental evaluation of applications for the Semantic Web. Our research focuses on various levels of the process of data set creation – data acquisition using wrappers, data preprocessing on the ontology instance level and adjustment of the ontology according to the nature of the evaluation step. Web application aimed at clustering of ontology instances is utilized in the process of experimental evaluation, serving both as an example of an application and visual presentation of the experimental data set to the user.

1 Introduction

Exponentially growing volume of information on the Web forces designers and developers to solve navigation and search problems with novel approaches. Faceted browsing, clustering and graph visualizations are only a few possibilities which can be – or even are – currently used. Nevertheless, in order to evaluate how any of these approaches improve navigation or searching, experimental data sets are always needed. Such data sets can be created either by generating artificial data or – as it is in our case – by acquiring data from existing real data sources. While using data from real data sources seem to be an attractive solution, a creation of such experimental data set comes with problems of its own.

We describe the process of creating an experimental evaluation ontology from existing data sources that represents a generalization of the process that we applied in the domain of scientific publications[1]. This process is influenced by the fact that the ontology is used as an experimental evaluation data set for clustering in an application for the Semantic Web. The major contribution of this work is design and experimental evaluation of a framework dedicated to data acquisition, ontology creation, data preprocessing and clustering.

[1] Project MAPEKUS: `http://mapekus.fiit.stuba.sk/`

1.1 Process of Data Set Development

Our approach exploits real data from existing resources as a basis for data set creation. In the process of creating such a data set firstly, suitable data sources, e.g., from the Web, are identified and data from these sources are retrieved (Section 2). Unfortunately, data acquired from various sources rarely share a perfectly common data model definition (ontology) thus a unified ontology is usually created and mappings between other data models are defined (Section 3).

There exist several methodologies of an ontology building process.

In [1] following reuse processes are combined together: *fusion/merge* and *composition/integration*. Using *fusion/merge* approach, the ontology is built by unifying knowledge from two or more different ontologies. In the *composition/integration* approach each reused ontology is a module of the resulting ontology.

Having the unified data model filled with data from all selected sources a cleaning process needs to be started in order to remove, duplications, inconsistencies or even completely wrong data (Section 3.2). Finally, such cleaned data set can be used to evaluate the semantic web-based application. In this paper, we discuss the usage of a data set to evaluate a semantic web application in the domain of scientific publications which exploits graph clustering methods based on graphs extracted from data set ontology (Section 4).

1.2 Domain for Experimentation

We have used proposed approach for an experimental data set development in our research project in the domain of scientific publications. In this project three major scientific publication portals (ACM, `www.acm.org/`, DBLP, `www.informatik.uni-trier.de/~ley/db/` and Springer, `www.springer.com/`) have been selected as the relevant sources for a large common scientific publication database. Acquisition of metadata from these sources resulted into a large amount of instances of authors, publications, journals, etc. Table 1 shows basic statistics indicating the scale of the acquired data sets.

Table 1. Instance counts of data sets acquired from ACM, DBLP and Springer.

	Data Set		
Instance	ACM	DBLP	Springer
Author	126 589	69 996	57 504
Organization	17 161	—	6 232
Publication	48 854	47 854	35 442
Keyword	49 182	—	—
Reference	454 997	—	—

This database serves as an ontology-based meta-data repository in our publication portal which aims at improving the navigation in such large information spaces by utilizing faceted and visual navigation by exploiting the faceted browser and visual navigation in clusters. While faceted browser operates directly on the base ontology, clustering is done on graphs extracted from this ontology. Figure 1 shows the overall process of data acquisition via wrapping, ontology integration, data cleaning, and finally graph extraction and clustering using as an example its special instance for our research project domain.

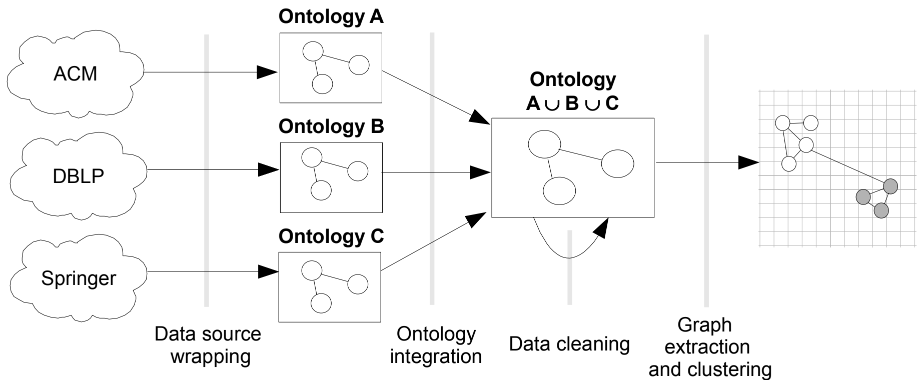

Fig. 1. Overview of the process of data set creation from three various data sources.

2 Data Acquisition

The purpose of the data acquisition stage is to create and populate the experimental data set of the respective domain with domain specific data. For evaluation purposes it is necessary to acquire a large amount of data, ideally from several different data sources, in order to create a statistically relevant sample.

2.1 Acquisition methods

We considered the use of different data acquisition methods ranging from manual approaches through assisted or semi-automatic approaches with only minimum user intervention required to set the specific initialization parameters to fully automatic approaches.

Extraction via web scraping. We employ two types of *web scraping* as suitable means for automatic extraction of huge data sets targeted at predetermined web sources, which contain suitable domain data:

– In case of *wrapper induction* the developer specifies positive and negative examples and the system produces extraction rules [2].
– *Special-purpose automated data gathering* is hand-written and thus custom tailored for a specific purpose – most commonly for the scraping of a particular web site.

Manual data entry. While tedious and not overly effective it is also possible to acquire data manually by data entry workers. This method was considered supplementary and was feasible for small sets of data only, albeit of a high quality. Manual data entry was used in our project scarcely and only when automatic acquisition was not possible.

2.2 Wrapper induction from user examples

The wrapper inducer learns the wrapper from positive examples gained from the user. The problem of wrapper induction, the verification and comparison of wrapping approaches was introduced by Kushmerick [3]. The wrappers are induced from examples gained from the user. The process of learning is outlined as Algorithm 1.

Algorithm 1 Wrapper induction learning process.

```
1: open the web page you wish to wrap
2: repeat
3:     select the example which is an instance of the pattern
4:     the wrapper inducer shows the generalization of the examples
5: until not all instances of the pattern are listed after the generalization
```

We call document information pieces we wish to wrap *document patterns*. A pattern is a coherent part of the document. Patterns are filtered out by filters learnt during the training process. The form and representation of the filter is the matter of the learning method. Patterns are often repeating structures, such as items on web pages listing publications of an author or institution. The patterns can be defined in recursive manner.

We describe a document as a set of elements called subdocuments. The subdocuments have attributes. We distinguish several possible *document representations*. The document representations can differ in a) the way of partitioning the documents into subdocuments, b) the attributes describing the subdocuments and c) the relation among the subdocuments. We considered two types of subdocuments so far in our project:

– *XML representation* – the subdocument is represented as W3C Document Object Model (DOM, `www.w3.org/DOM/`) structure. The elements of the documents are addressable by XPath expression. We operate on the XPath expression of the subdocument.
– *Attributed elements* – the subdocuments are also elements of the DOM-tree. The attributes of the subdocuments are type (name) of the element, style class, depth of the element in the DOM-tree and an index (if the element is included in a table).

Recently wrapping from visual web page representation is being researched [4]. Visual representation concerns the visual features visible by the user browsing a web page. Relying on the HTML structure of the document does not reflect always the web pages' visual features as creators of web pages often do not want the page to be wrapped. Visual representation aims to tackle such situations.

The wrapper inducer is trained on the user examples (subdocuments). The result of the training is a generalization of the examples. We set the following conditions on the generalization: a) each positive example must be in the resulting selection; b) no negative example can be in the resulting selection.

We proposed and implemented two learning strategies for generalization of the document pattern.

Simple XPath learning strategy. The strategy generalizes only from positive examples and operates on the XML document representation. The process of learning starts with an empty filter. The examples are XPath expressions. The algorithm of the learning process is outlined as Algorithm 2.

Algorithm 2 Process of generalization of the XPath expressions in simple XPath learning strategy.

```
1: Generalization ← receive the first example from the user
2: while the training process is not terminated do
3:     NewExample ← a new example from the user
4:     update Generalization with NewExample
5: end while
```

Updating the generalization is realized by one of the following actions: a) *index removal* – both XPath expressions are compared element by element from the left side of the expressions. If any index on this path differs this index is removed from the generalization; b) *XPath truncation* – if any tag in the XPath expressions differ, the tag and the path after the tag is truncated.

Attribute selection learning strategy. We apply machine learning methods on the attributes (listed in description of the document representation) of the subdocument. The strategy is able to learn from positive and negative examples. The "positivity" of an example is used as the *next* attribute of the examples. If any of the attribute is missing, its value is set to special value *!missing* and is used in the process of classification [2].

3 Data Preprocessing

Data preprocessing includes methods and procedures which transform the data obtained in the process of data acquisition to a form which is more appropriate for data processing and experimentation.

3.1 Data Integration

In the first phase of data preprocessing it is necessary to integrate data from different sources and thus different models into the one. It includes:

– *Unified data model definition* – data model should be strong enough to represent all instances and their relations from all input sources. For this purpose the domain ontology for publications metadata was created based on generalizations of relevant parts of source data models.
– *Data mapping definition* – for every data source a definition of data mapping between source data model and unified data model is required. In our case, this transformation is executed already in wrappers.

3.2 Data cleaning

This part of the process is aimed at cleaning of inconsistencies, which are present in data because of:

– *Inconsistencies in the source data model* – inconsistencies and other flaws like data duplicities, incomplete or wrong data can be transferred from source model.
– *Inconsistencies due the source integration* – because data are integrated from many sources, duplicities of instances for example authors or publications may occur.
– *Inconsistencies created in the wrapping process* – there is a chance that duplicates are created during the wrapping process, for example not all links to the authors can be followed and checked due to high increase of the process duration.

Single-pass instance cleaning. This phase is designed to correct data flaws in scope of one instance like:

– correcting the format of some data types, e.g., names, which should start with capital letter;
– separating data fields, e.g., separating first names and surname;
– filtering of instances with insufficient data to work with them;
– filtering of data fields, e.g., conjunctions from key terms.

The instance cleaning is realized as a set of filters, one for each particular task. This solution is based on pipes and filters architecture. Whole process can be realized by one pass through all instances and therefore has linear time complexity. It is effective to realize it directly after acquisition of each instance using a wrapper yet before storing it because the processor is only lightly loaded while downloading data from sources.

3.3 Duplicates Identification

The aim of this phase is to identify instances that are describing identical entity (e.g., author or publication). As the acquisition process is based on the semantics of the domain and extracts data together with their meaning to common domain ontology no approaches for comparing concepts and discovering a mapping between them [5] are needed.

We combine two methods, comparing data itself (in terms of ontology data type properties) and working with relations between data (object type properties), which was already described in [6].

The overall similarity of the instances is computed from similarity of their properties that are not empty. For each property of an instance we use weighting method with positive and negative weight. This is more general than simple weighting, which in many cases is not sufficient. For example, consider the country property of two authors – if the similarity is low it means that the authors are not likely the same person (need for strong weight), but if it is high, it does not mean, the authors likely are the same person (need for low weight). The parameters are three values, positive weight p, negative weight n, and threshold t, which determines where to use positive weight and where to use negative. Overall similarity $S \in < 0, 1 >$ is calculated as:

$$S = \frac{\sum_{i=1}^{n} F_i\left(s_i\right) + n_i}{\sum_{i=1}^{n} p_i - n_i} \tag{1}$$

where n is number of properties, s_i represents the similarity of i-th properties ($s_i \in < 0, 1 >$), p_i stands for positive weight, n_i for negative (values between 0 and 100) and F_i represents step function, which is calculated as

$$F_i(x) = \begin{cases} p_i x & \text{if } x \geq t_i \\ -n_i\left(1 - x\right) & \text{if } x < t_i \end{cases} \tag{2}$$

where i is the index of property, p_i is positive weight, n_i negative and t_i stands for threshold of i-th property (value between 0 and 1).

To decide whether the instances are identical a threshold is applied. If the similarity is above its value the instances are evaluated as identical.

Every instance is compared to every other instance of its class, which leads to quadratic asymptotic time complexity. To shorten the duration of comparing process we use a simple clustering method for authors and divide them into groups by the first letter of their surnames.

Comparison of data type properties. This method is based on comparison of corresponding data type properties of two instances of the same class. For the data comparison there are used several string similarity metrics like QGrams, Monge-Elkan distance, Levenshtein distance and many other[2]. Different method can be specified for each property including so called *composite metric*, which is weighted combination of several metrics (for example QGrams with weight of 0.6 and Monge-Elkan, weight 0.4), where weights are set according to the

[2] Chapman, S.: SimMetrics, `www.dcs.shef.ac.uk/~sam/stringmetrics.html`, Cohen, W.: Record Linkage Tutorial: Distance Metrics for Text, `www.cs.cmu.edu/~wcohen/Matching-2.ppt`

results of the experiments. We also use special metrics for some properties, e.g., names where we consider abbreviations of their parts. Some properties needs to be compared only for identity, e.g., ISBN of the books.

Comparison of object type properties. The principle of this method is to compare the relations to neighboring instances in the ontology. With increasing number of mutual instances the probability that compared instances are the same grows.

For each object property (e.g., authors of two books) we compare their datatype properties to determine how close to each other they are. Each match is included in computing the overall similarity. Thus if two books have three mutual authors, all three partial similarities are counted. The same approach also applies on the authors that are different.

3.4 Duplicates resolving

We identified several possibilities when two instances were identified as identical:

- mark instances as identical via special object property,
- manually resolve (this possibility is appropriate, when small number of duplicates was found),
- delete the instance with less information,
- join instances, take the data type properties with higher length, join non-functional object properties.

3.5 References disambiguation

Aim of this phase is to identify and disambiguate identical references between publications [7]. Parts of references can be recognized using regular expressions and set of experimentally discovered rules. The process of identity identification between two references consists of three steps:

1. *Comparison of titles* – the title similarity is identified using the Levenshtein distance. If the ratio of this value and the length of both strings greater than 5% (value set as a result of experiments with publications metadata) references are considered not to be identical. Otherwise the process continues with the next step.
2. *Comparison of years* – if the years in references are different, references are not identical. If they are equal proceed to step three.
3. *Comparison of authors* – if at least one author is mutual for both references, they are considered to be identical.

4 Data Processing

After the proceprocessing is done, processing and presentation are applied. In our case user navigation in the metadata enhanced data set is employed, e.g. faceted navigation [8] and visual navigation in clusters of data instances. We describe as an example of processing the instance clustering.

4.1 Extracting Graphs from Ontologies

While data instances usually incorporate complex representation of a state-space, clustering all the instances bound together with all types of properties is computationally demanding. In such cases, it is worth utilizing a graph extraction from the ontology.

We represent ontology by the Web Ontology Language (OWL, `www.w3.org/TR/owl-features/`). The OWL representation consists of triplets (RDF based subject-predicate-object construction [9]), which serve as the basis for graph representation of ontology.

Example in Figure 2 shows a scientific publication ontology which consists of instances which belong to classes "Author", "Paper" and "Keyword" and properties between instances are "hasKeyword", "isWrittenBy" and "references", and extraction of all vertices of class "Author" and edges defined by the property "isWrittenBy".

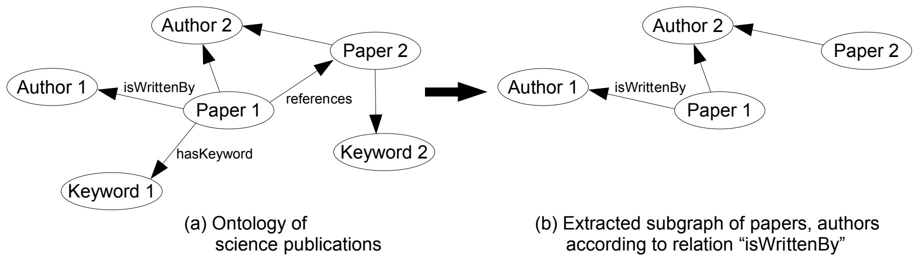

Fig. 2. Extraction of graph of papers and authors from ontology of publications.

4.2 Clustering

The process of clustering utilizes the unsupervised learning schema [10], which is advantageous when no exact taxonomy of data set exists. The disadvantage of involvement of clustering is that created clusters usually do not have attached semantic labels, hence resolving why particular data instances were grouped together can be hard in some cases. An example of graph-based clustering is depicted in Figure 3.

Composing hierarchy of clusters. We represent the information in a space of hierarchical clusters. While many clustering methods were invented over the time [11], almost any clustering method can be simply used when it is properly implemented according to the JUNG library (Java Universal Network/Graph Framework, `jung.sourceforge.net/`), used during the clustering. Hierarchical composition of clusters is implicitly used. Clustering of the input graph is following:

1. Input graph is loaded from relational database and converted to a graph representation using the JUNG library.

2. Clustering parameters are determined, e.g., which clustering method is used, number of cluster layers in the hierarchy, number of vertices in cluster (if the selected clustering method is designed to be parameterized in such way).
3. Layers of the cluster hierarchy are created sequentially from bottom to top.
4. Whole hierarchy of clusters is stored in relational database. With the aim to allow any operations with the clusters, also inverted hierarchy is generated and stored.

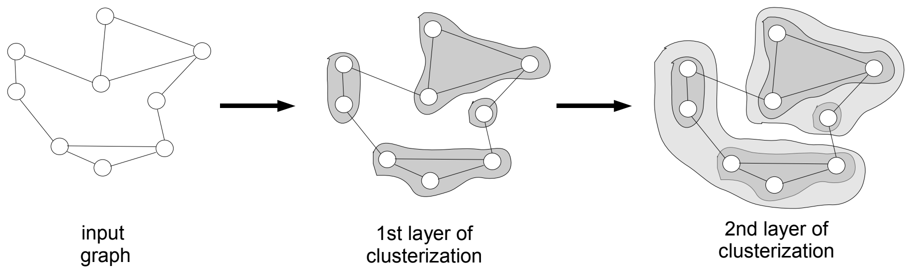

Fig. 3. Example of two layered graph clustering.

5 Evaluation – Duplicate Identification

To measure the effectiveness of duplicity identification process data from DBLP together with artificially created duplicates were used. The instances were randomly selected and the duplication of one instance included:*(i)* Mutation of randomly selected datatype properties with one of these operations: words exchange, letter exchange, word deletion, letter deletion, letter duplication; *(ii)* For each object property there is a chance to be deleted or its reference to be duplicated and its datatype properties mutated like in previous step.

Ten measurements were made for each data set of 1 000, 5 000, 10 000 and 20 000 publications, with 100 injected duplicities. Surname clustering for authors, Levenstein metric for publication titles, special name metric for author given and family names and exact string comparison for ISBN were used. These metrics were weighted for each property to achieve best results. Similarity threshold of 0.8 was used (each pair of instances, that achieved score of 0.8 and greater on the scale from 0 to 1 were considered to be duplicity).

In Table 2 there are columns with number of identified duplicities (column *correct*), wrong identified duplicities so called false positives (column *wrong*) and unidentified duplicities (column *missing*). It contains also statistical measures such as precision, recall and F1 measure. Each cell contains average value and standard deviation.

Only instances that could be identified after clustering are considered in these measures, because clustering is not a part of identification method. Without clustering, the results should be same for all 100 duplicates, but the whole process would be much more time consuming.

Precision is almost constant in each data set. There is a linear growth in number of false positives, but number of compared instances grows also linear and number of comparisons grows quadratic.

Table 2. Results of duplicate identification experiment on various DBLP sample sizes.

	Duplicity identification					
Sample size	*Correct*	*Wrong*	*Missing*	*Precision*	*Recall*	*F1*
1 000	53.2 ± 4.23	5.7 ± 7.52	14.1 ± 4.37	0.79 ± 0.06	0.90 ± 0.08	0.84 ± 0.04
5 000	70.6 ± 5.93	6.6 ± 5.95	18.8 ± 7.05	0.79 ± 0.07	0.92 ± 0.05	0.85 ± 0.05
10 000	74.5 ± 6.07	20.7 ± 5.20	17.7 ± 4.19	0.81 ± 0.05	0.78 ± 0.04	0.80 ± 0.04
20 000	81.3 ± 4.38	24.7 ± 5.58	13.9 ± 2.98	0.85 ± 0.03	0.77 ± 0.02	0.81 ± 0.02

Number of duplicates in real data was also measured. Data from each source and data combined from all sources were used (Table 3). Found duplicates were checked manually and over 90% of them were correct.

Table 3. Result of duplicate identification experiment on real data.

Sample size	*ACM*	*DBLP*	*Springer*	*Combined*
1 000	44	5	10	24
2 000	96	7	24	36
3 000	133	25	41	48
4 000	154	26	55	62
5 000	175	29	68	78

Some interesting facts have been discovered in the domain of publications:

- Many duplicities of the publications are present only because of capital letters.
- Chinese names are hard to distinguish, because they are very short.
- There is a problem with various editions and re-editions of books, their title differs only in one number or short word.
- The family members, e.g. brothers writing the same books and having mutual collaborators are hard to distinguish.

6 Conclusions

The data integration process is important when evaluating applications for the Semantic Web on real data sets from various sources. However, only involvement of powerful data acquisition and preprocessing methods can ensure the quality of the resulting data set.

In our research, we focused on integration of the data from various large scale data sources in the domain of scientific publications, from which relevant data was acquired using two different *web scraping* approaches and stored in the ontological repository for further processing and support of navigation.

During the evaluation we also encountered several bottlenecks that currently seriously limit experimenting and widespread deployment of applications for the Semantic Web, mainly the general immaturity of ontological repositories in terms of their processing speed of ontological queries. Having hundred thousands of instances in the ontology, preprocessing realized along with data gathering from their source proved as well-suited approach.

Several of proposed methods for data preprocessing presented in this paper are also suitable for merging various sources of meta-data in particular domain order to provide effective visualization and navigation in data.

Acknowledgment. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104 and the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

1. Pinto, H.S., Peralta, D.: Combining Ontology Engineering Subprocesses to Build a Time Ontology. In: K-CAP'03, ACM Press (2003) 88–95
2. Čerešňa, M.: Interactive Learning of HTML Wrappers Using Attribute Classification. In: Proc. of the First Int. Workshop on Representation and Analysis of Web Space, Prague, Czech Republic (2005) 137–142
3. Kushmerick, N.: Wrapper Induction: Efficiency and expressiveness. Artificial Intelligence **118**(1) (2000) 15–68
4. Simon, K., Lausen, G.: ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. In: CIKM'05, ACM Press (2005) 381–388
5. Weinstein, P.C., Birmingham, W.P.: Comparing Concepts in Differentiated Ontologies. In: KAW'99. (1999)
6. Andrejko, A., Barla, M., Tvarožek, M.: Comparing Ontological Concepts to Evaulate Similarity. In Návrat, P., et al., eds.: Tools For Acquisition, Organisation and Presenting of Information and Knowledge, STU (2006) 71–78
7. Rado, L.: Sharing of Research Results on Portal based on Semantic Web. Master's thesis project report, Bieliková, M. (supervisor), Slovak University of Technology in Bratislava (2007)
8. Tvarožek, M., Bieliková, M.: Adaptive Faceted Browser for Navigation in Open Information Spaces. In: WWW'07, ACM Press (2007) 1311–1312
9. Beckett, D.: Redland RDF Storage and Retrieval. In: SWAD-Europe Workshop on Semantic Web Storage and Retrieval. (2004)
10. Hinton, G., Sejnowski, T.J.: Unsupervised Learning and Map Formation: Foundations of Neural Computation. MIT Press (1999)
11. Frivolt, G., Pok, O.: Comparison of Graph Clustering Approaches. In Bieliková, M., ed.: IIT.SRC 2006, Bratislava, Slovakia (2006) 168–175