Similar Structures inside RDF-Graphs

Anas Alzogbi and Georg Lausen

Institut für Informatik, Universität Freiburg
Georges-Köhler-Allee, Geb. 051, 79110 Freiburg
alzoghba|lausen@informatik.uni-freiburg.de

ABSTRACT

RDF is the common data model to publish structured data on the Web. RDF data sets are given as subject-predicate-object triples and typically are represented as directed edge-labeled graphs. To make the information represented by such graphs comprehensible, RDF-schema (RDFS) provides concepts to define a class-structure as part of the given RDF-graph and thus supports a more abstract view on the data set. In this paper we follow a different approach and propose to make an RDF graph more comprehensible by reducing its size by partitioning to discover subgraphs which are similar with respect to their structure. The methods applied to derive a partition are based on bisimulation and agglomerative clustering. We demonstrate the usefulness of the approach by applying it on several synthetic and one real world RDF datasets.

1. INTRODUCTION

RDF (*Resource Description Framework*) and RDFS (*RDF Schema*) constitute the standard model for data publishing and interchange on the web [1]. There is an increasing number of RDF data sets being published on the web. An impressive example demonstrating the wide range of available data sets is given by the LOD (*Linked Open Data*) cloud [2], which is formed out of a remarkable number of interlinked RDF data sets. To make these data sets useful in practice, meta-information can be assigned describing the characteristics and structure of a data set at hand in appropriate detail. As a bridge between the publishers and users of RDF data, W3C proposes the VoID-vocabulary (*Vocabulary of Interlinked Datasets*) [3]. However, the VoID-vocabularies intention mostly is to make only statistical and linking information about a data set explicit and not its structure in some detail. A tool to generate voiD-annotations automatically which, to a certain extend, takes also structure into account is described in [4].

RDF provides a means of asserting facts about certain objects in the form of $(subject, predicate, object)$ *triples*, respectively (s, p, o)-triples. The intended semantics is that *subject* is related to *object* via *predicate*. RDF data most intuitively is represented by an edge-labeled directed *RDF graph* G, where each (s, p, o)-triple gives rise to an edge directed from s to the o carrying as label p. We write $G = (V, L, E)$, where V is the set of nodes, E the set of edges and L an arc-labeling function. In this paper we will concentrate on the question how the size of a given RDF graph can be reduced to a considerably smaller reduced RDF graph G^r without losing too much of the original inherent structure. This goal is related to several lines of research. Size reduction can be achieved by extracting a schema of an RDF graph. In certain cases this task may be eased when RDF data already contains RDFS type information or even is linked to an expressive ontology. However, as the intention of RDF is to provide an open minimally constraining way for representing information, the quality of an existing typing remains questionable. Examples for work related to schema extraction are [5, 6, 7, 8]. Another line of research is motivated by the construction of a path-index to make the evaluation of structure-aware queries on an RDF graph more efficient (e.g. [9, 10, 11, 12, 13]). Moreover, schema extraction is related to the techniques developed for ontology alignment, as matching two ontologies may contain the subtask of reducing the size of the input ontologies (cf. e.g. [14, 15]).

To reduce the size of a RDF graph $G = (V, L, E)$ the approach presented in this paper is based on finding a partition $\mathcal{P} = \{B_1, \ldots, B_n\}$ of the set of nodes V, where the elements of $\mathcal{P}$ are the nodes of the reduced graph G^r. Edges $B \xrightarrow{a} B'$ are introduced in G^r whenever there exists an edge $p \xrightarrow{a} p'$ in G, where $p \in B$ and $p' \in B'$. The ultimate goal of our approach is to group only such nodes p, p' into a common block B of $\mathcal{P}$ whenever they are *similar*. The notions of similarity we shall use take the paths originating at p and p' into account and will be defined later in the paper. We evaluate our method on different well-known RDF datasets and will demonstrate the possible trade-off between strength of similarity and size of reduction.

Technically, our approach consists of a bisimulation-step [16] followed by an agglomerative clustering-step [17]. Similar attempts have been conducted before, e.g. [5, 13, 11]; however in [5] for similarity only depth 1 is considered and [13, 11] do not combine bisimulation and clustering. The contribution of the current paper is a discussion of the interplay of bisimulation and agglomerative clustering with the goal to reduce the size of an RDF graph while keeping its structure as much as possible. The paper is organized as follows. In Section 2 we demonstrate the usefulness of bisimulation for characterizing an RDF graph and in Section 3 we continue with discussing agglomerative clustering. Section 4 presents our experiments and Section 5 concludes the paper.

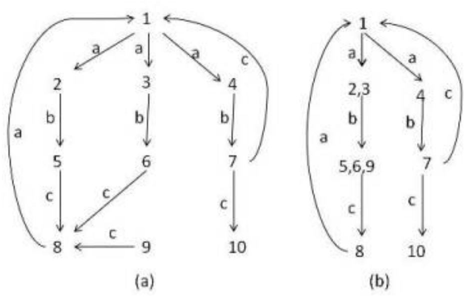
2. SIMILAR STRUCTURES BY BISIMULATION

Bisimulation allows us to define sets of nodes of an RDF graph which cannot be distinguished by looking at the sequence of predicates of their paths. The following definitions are taken from [18] with only minor modifications to adapt them to RDF graphs. Let $G = (V, L, E)$ be an RDF graph. A binary relation $R \subseteq V \times V$ is a strong bisimulation if for all $p, q \in V$, such that pRq, the following two conditions hold. (1) If $p \xrightarrow{a} p'$, then $\exists q' \in V : q \xrightarrow{a} q' \wedge p'Rq'$. (2) If $q \xrightarrow{a} q'$, then $\exists p' \in V : p \xrightarrow{a} p' \wedge p'Rq'$. If a strong bisimulation R exists such that pRq, then p and q are strongly bisimilar.

The problem we are interested in is to find the equivalent classes of the largest strong bisimulation of the RDF graph, i.e. to find a partition of V such that the corresponding blocks are largest with respect to bisimilarity of their elements. Influenced by [18], the algorithm to compute a strong bisimulation we used is a implementation of the *naive method* [19]; in spite of the worst-case time-complexity of $\mathcal{O}(MN + N^2)$, where N is the number of nodes and M the number of edges in the RDF graph, for our current analysis this was acceptable (see next section).[1] The algorithm starts with putting all subjects in one block and iteratively splits blocks of a partition to derive a new partition until any two members of every block have the same signature with respect to that partition. Again adapting definitions from [18] the signature of a subject s with respect to a certain partition $\mathcal{P}$ is the set of s's outgoing edges to objects in blocks of $\mathcal{P}$: $sig_{\mathcal{P}}(s) = \{(a, B) \mid s \xrightarrow{a} o \text{ and } o \in B \in \mathcal{P}\}$. Let B be a block in some partition of V. B is called *final*, if it also is contained in the partition of the strong bisimulation.

Assume that n iterations are required to compute a strong bisimulation. If only k, $1 \leq k \leq n$, iterations are performed, the blocks of the resulting partition define a k-bisimulation. Intuitively the nodes in a block of a k-bisimulation are equivalent with respect to their paths of length up to k. As demonstrated in Figure 1, the RDF graph shown in (a) has a strong bisimulation (b) and gives rise to the partitions shown in (c). The parameter k in the table indicates the respective partitions for k-bisimulation. It is interesting to see for example, that up to $k = 2$ subjects $\{2, 3, 4\}$ form a block; however after one iteration more this block is split as these subjects are related with objects depending on different predicates. Note further, that certain blocks become final before the final iteration $k = 3$.

Strong bisimulation gives us a rather strong measure of similarity: for any two subjects s, s' in the same block B it is guaranteed that for each path $s \overset{a_1}{\to} s_1 \overset{a_2}{\to} s_2 \overset{a_3}{\to} \ldots$ originating at s there exists a path originating at s' of the same length which carries the same sequence of predicates $s' \overset{a_1}{\to} s'_1 \overset{a_2}{\to} s'_2 \overset{a_3}{\to} \ldots$, where in addition s_i, s'_i are elements of the same block B_i, $i \geq 1$. Therefore, we do not lose structural information when instead of G we consider a reduced graph G^r whose nodes are exactly the blocks of its strong bisimulation and labeled edges between such nodes are drawn whenever an edge exists in the original graph between elements of the respective blocks with the same label (cf. Figure 1). However it has been mentioned in several papers, e.g. [13, 10], that the reduction of the number of nodes strongly depends on the structure of the RDF graph under consideration and therefore a limited notion of similarity based on k-bisimulation seems to be the only viable way.



k	partition	new final blocks
1	$\{1,8\},\{2,3,4\},\{5,6,7,9\},\{10\}$	1
2	$\{1\},\{8\},\{7\},\{2,3,4\},\{5,6,9\},\{10\}$	4
3	$\{1\},\{8\},\{7\},\{2,3\},\{4\},\{5,6,9\},\{10\}$	2

(c)

Figure 1: RDF graph (a), its strong bisimulation indicated by a reduced representation of the graph whose nodes are the blocks of the according partition which is exhibited in (c). The table contains the partitions resulting from k-bisimulation, where $k = 3$ forms a strong bisimulation.

data set	subjects	objects	predicates	edges
SP^{2}Bench250K	50K	100K	61	250K
LUBM2	40K	20K	32	240K
LUBM2R	40K	20K	35	360K
BSBM500K	48K	100K	40	500K
SwDogFood	25K	55K	170	290K

Figure 2: RDF data sets analyzed.

For our analysis we have chosen four benchmarks (SP^{2}Bench [21], LUBM [22], BSBM [23]) and one real dataset (SwDogFood [24]). Figure 2 gives more details; LUBM2R in contrast to LUBM2 contains also the inferable triples. As efficiency of the computation of a bisimulation is not a topic of this paper and benchmarks typically reflect the same structure when varied in their size, the sizes of our datasets are rather modest. In Figure 3 we summarize our results.

The reduction of size of the RDF dataset obtainable by k-bisimulation is of interest. If the number of partitions is small even for large k, then this indicates a homogeneous structure as it might result from an existing typing of the

[1]Recently an efficient external-memory based algorithm for bisimulation has been presented [20]; however this algorithm cannot handle cycles in a graph and therefore can not be applied on RDF datasets.

subjects in the RDF dataset - similar to a typing of a relational database. In contrast, a large number of blocks even for small k indicates structural heterogeneity or absence of typing. For datasets originating from benchmarks SP^{2}Bench, BSBM and LUBM2 we can observe a reduction by 99% or even more which indicates a rather regular underlying graph structure. However for benchmark dataset LUBM2R and real dataset SwDogFood only a reduction of 25% - 60% could be obtained. For these datasets many of the blocks must have only one or two elements. In the next section we shall demonstrate, that also in such cases a considerable reduction of the size of the dataset can be achieved. While this observation clearly underlines the limitations of bisimulation, our analysis however shows that even for rather small k the quality of the blocks is surprisingly high, as most of the blocks are already final with respect to strong bisimulation. For example, for SwDogFood a 3-bisimulation gives us a reduction of the input size by 35% where nearly 95% of the blocks are final, i.e. contained in the partition of strong bisimulation. Thus, when we replace the initial RDF graph G by its reduced version G^r according to the partition resulting from 3-bisimulation, there will be only a small corruption of the initial structure of G.

data set	k	blocks	%subjects	final
SP^{2}Bench250K	1	171	0.004	97
	2	592	0.012	576
*	3	609	0.013	609
LUBM2	1	27	0.001	13
	2	60	0.002	33
	3	124	0.003	93
	4	203	0.005	200
	5	206	0.005	204
	6	209	0.005	208
*	7	210	0.006	210
LUBM2R	1	27	0.001	6
	2	89	0.002	29
	3	734	0.02	373
	4	11884	0.31	10157
	5	16167	0.42	16129
*	6	16208	0.42	16208
BSBM500K	1	49	0.001	31
	2	483	0.01	482
*	3	484	0.01	484
SwDogFood	1	972	0.04	466
	2	10161	0.40	8865
	3	16432	0.65	15657
	4	18604	0.74	18295
	5	19082	0.75	18973
	6	19273	0.76	19243
	7	19302	0.76	19290
	8	19316	0.76	19314
	9	19317	0.76	19316
*	10	19318	0.76	19318

Figure 3: Bisimulation anlysis. For each dataset k indicates the result of k-bisimulation. By * the corresponding row represents a strong bisimulation. Column *blocks* states the number of blocks of the respective k-partition and column *final* the number of blocks inside that partition which are already contained in the partition of strong bisimulation. Column *%subjects* states the quotient of the number of blocks by the total number of subjects of the RDF dataset.

3. SIMILAR STRUCTURES BY CLUSTERING

Now we are going to construct new partitions starting from the partition which is the result of k-bisimulation for some RDF graph G with respect to a chosen k. To this end we consider the reduced RDF graph $G^r = (V^r, E^r)$ and compute a similarity matrix S, which contains the pairwise similarity values between each pair of nodes in G^r, i.e. each pair of blocks in the k-bisimulation. The following notion of similarity weakens the strong on bisimulation based form of similarity used so far.

Let v' be a node in G^r, then $T_\sigma(v')$ is the instance tree of G^r which contains all nodes and edges which can be reached when following all possible paths in G^r starting at v' up to length σ. Now, let $T_\sigma(v')$, and $T_\sigma(w')$ be two such instance trees. The intersection $intersect(T_\sigma(v'), T_\sigma(w'))$ of the instance trees is a tree whose root represents v' and w' and further contains all the rooted edge-labeled paths which appear in $T_\sigma(v')$ and as well in $T_\sigma(w')$. Let *size* denote the number of nodes of a tree under consideration. To decide whether or not nodes are to be treated as similar we define the similarity $sim(v', w')$ between nodes (representing blocks of a partiton) v', w' of G^r by

$$sim_\sigma(v', w') = \frac{size(intersect(T_\sigma(v'), T_\sigma(w')))}{(size(T_\sigma(v')) + size(T_\sigma(w')))/2}.$$

The notion of intersection trees is borrowed from [25], where it is introduced as basis for the design of kernel methods used for a learning task for prediction of attributes and links. In contrast to this we use intersection trees as basis for our notion of similarity inside unsupervised clustering. The similarity between each pairs of the internal nodes in $G^r(V^r, E^r)$ is calculated and stored in the similarity matrix S.

A hierarchical clustering method is a procedure for transforming a similarity matrix into a sequence of nested partitions. We use an agglomerative algorithm [17] for the purpose of applying the hierarchical clustering. Before explaining the agglomerative algorithm we will define the concepts of partition and nested partitions.

Let I_{G^r} be the set of internal nodes in G^r defined as: $I_{G^r} = \{v \in V^r : \exists w \in V^r \wedge (v, w) \in E^r\}$. Elements of I_{G^r} are the objects to be clustered. A partition $\mathcal{P}$ over I_{G^r} breaks I_{G^r} into subsets $\{C_1, C_2, \ldots, C_m\}$ under the following conditions: (i) for i and j from 1 to n, where $i \neq j \Rightarrow C_i \cap C_j = \emptyset$, (ii) $C_1 \cup C_2 \cup \ldots \cup C_m = I_{G^r}$. The components of the partition are called clusters. Partition $\mathcal{P}$ is nested into partition $\mathcal{X}$ if every cluster of $\mathcal{P}$ is a subset of a cluster of $\mathcal{X}$.

The agglomerative algorithm for hierarchical clustering is taken from [17] and is called Agglomerative Algorithm for Complete-Link Clustering; it starts by placing each internal node of the graph G^r in an individual cluster, and continues building the threshold graph, which is an undirected, unweighted graph without self-loops or multiple edges. The nodes of the threshold graph are the internal nodes of the graph G^r (I_{G^r}). A threshold graph $G_\tau(V_\tau, E_\tau)$ is defined for a certain similarity value τ by inserting an edge (i, j) between the nodes i and j if the objects i and j at least have a similarity value of τ. That is, $(i, j) \in E_\tau$ if and only if $S[i, j] >= \tau$.

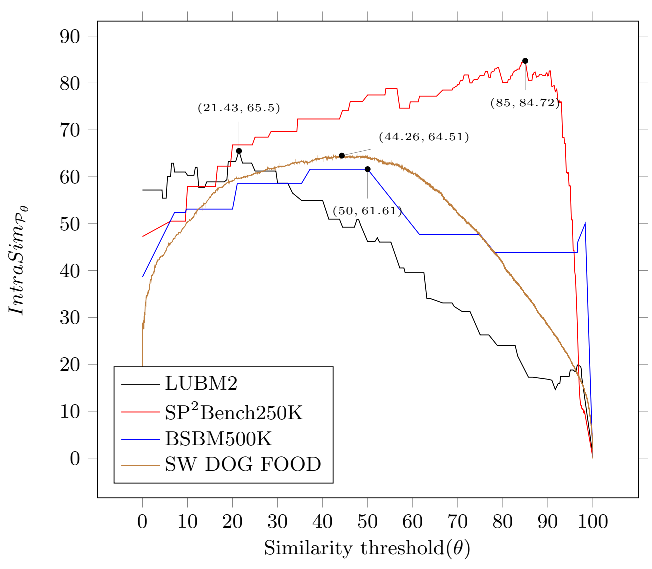


Figure 4: *IntraSim* for each similarity threshold.

Initially the algorithm builds the threshold graph G_∞ (which contains no edges), and then it proceeds by visiting the entries of the similarity matrix S in a descending order over their values $S[i,j]$. For each similarity value $\tau = S[i,j]$, an edge between the nodes i and j is added in the threshold graph G_τ. Constructing the threshold graph in this way, enables us to discover similar objects and to group them into one cluster. Two clusters are considered to be similar if there is an edge connecting each pair of their components, i.e the induced graph formed by their components and all the edges related to these components has a clique. The algorithm continues by visiting all the similarity values until one of the following conditions is met: (i) A given similarity threshold is reached, or (ii) The algorithm visited all the values of the similarity matrix, or (iii) All the nodes are grouped in one cluster. For each similarity value the algorithm constructs the corresponding partition and adds it to the partitions list which is the output of the algorithm.

4. EXPERIMENTAL RESULTS

In this section we present the experimental results obtained by applying bisimulation and agglomerative clustering algorithms successively on the data sets described in Figure 2, where from the two LUBM-versions we considered only LUBM2. We chose $\sigma = k$ as max-value for the definition of similarity of the nodes in G^r, where k is the number of iterations needed for strong bisimulation: SP^{2}Bench:k=3; LUBM2:k=7; BSBM500K:k=3; SwDogFood:k=10. The agglomerative clustering algorithm produces a list of partitions where each partition $\mathcal{P}$ is composed of several clusters, and is induced by a similarity value θ and is identified by the number of clusters ($|\mathcal{P}_\theta|$). In order to evaluate the results, we are interested in two measures: the size of the partition $|\mathcal{P}_\theta|$ which reflects the reduction in the size achieved by the algorithm, and the partition intra-similarity.

The intra-similarity of a partition $\mathcal{P}_\theta$ is the average over the cluster intra-similarity values:

$$IntraSim_{\mathcal{P}_\theta} = \frac{1}{|\mathcal{P}_\theta|} \sum_{c \in \mathcal{P}_\theta} IntraSim_c.$$

The cluster intra-similarity $IntraSim_c$ is calculated for each cluster c of size n as the average of the similarity values among the cluster components:

$$IntraSim_c = \frac{1}{\lambda} \sum_{i<j}^{n} S[c[i], c[j]],$$

where $\lambda = \frac{n(n-1)}{2}$ the number of edges among all the nodes of the induced graph for cluster c. Figure 4 shows the $IntraSim_{\mathcal{P}_\theta}$ values for all the partitions $\mathcal{P}_\theta$: $1 \leq \theta \leq 100$ produced by the algorithm. The initial value of $IntraSim_c$ for each cluster is zero. As the algorithm proceeds, it starts joining clusters with most similar components into new clusters, which results in an increasing $IntraSim_c$ for those new clusters, causing the $IntraSim_{\mathcal{P}_\theta}$ value to raise gradually until reaching the maximum value. Figure 4 depicts this behaviour for all the data sets.

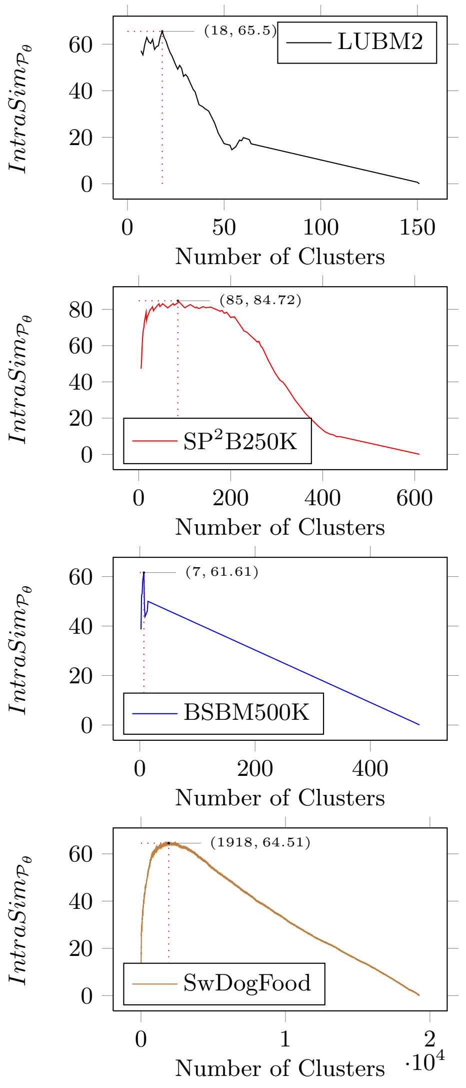


Figure 5: $IntraSim_{\mathcal{P}_\theta}$ and partitions' sizes.

For each dataset, we are interested in the partition which corresponds to the maximum $IntraSim_{\mathcal{P}_\theta}$ value, as such a partition is expected to exhibit clusters where the most similar objects are grouped in one cluster over all the partition's clusters. There is a clear trade-off between the similarity of the components of a cluster and the size of the partition. Figure 5 depicts the relation between $IntraSim_{\mathcal{P}_\theta}$ and the partition size ($|\mathcal{P}_\theta|$). The maximum $IntraSim_{\mathcal{P}_\theta}$ is associated with a relatively small number of clusters, and, if we are interested to reduce the number of clusters, this can be achieved by choosing a smaller similarity threshold while keeping an $IntraSim_{\mathcal{P}_\theta}$ value still close to the maximum. For example, for SP^{2}Bench250K and SwDogFood data sets, where the number of clusters are 85 and 1918, respectively, the corresponding curves in Figure 5 show that we can achieve a much smaller number of clusters when choosing a partition implied by an $IntraSim_{\mathcal{P}_\theta}$ value which still is larger than 50% and thus still close to the maximum.

Finally we will comment on the quality of the obtained partitions. While a detailed analysis is beyond the scope of this paper, a general discussion, however, can be outlined. To decide, whether or not two subjects put into the same cluster are indeed similar, we can look at the type-information associated with them in the data set. For example, in data set SwDogFood, roughly 22K of the 25K subjects have an associated RDF type. Among those types many have been deprecated[2] and we replaced them as suggested. In Table 1 we summarize our analysis. For SwDogFood and SP^{2}Bench we have more clusters than types - this indicates that there might exist interesting subtypes which so far have not been used. For LUBM2 and BSB500K we have more types than clusters, an indication for supertypes. We checked all clusters for possible errors, where we call an assignment of a subject to a cluster an error, when the type of this subject does not conform to any of the types of the majority of the other subjects in the same cluster. For example, for SwDogFood we found a cluster containing 21 subjects of type *ResearchTopic*, while all other 36 subjects where of spatial types *MeetingRoomPlace*, *ConfereneVenuePlace*, *Room* and *SpatialThing*. Similarly, for LUBM2 we found a cluster with two universities, where the other subjects in the cluster were courses.

data set	subjects	RDF types	clusters	errors
SP^{2}Bench	50K	9	85	0
LUBM2	40K	14	6	2
BSB500k	48K	9	7	0
SwDogFood	25K	43	1918	22

Table 1: General analysis

5. CONCLUSION

In this short paper we have presented our first results concerning automatically partitioning an RDF dataset aiming at a reduction of the size. The results of our experiments are encouraging. Future work will aim on the analysis of larger RDF datasets.

6. REFERENCES

[1] "RDF Specification Overview (w3c)," http://www.w3.org/standards/techs/rdf#w3c_all.

[2] "The Linking Open Data cloud diagram," http://richard.cyganiak.de/2007/10/lod/.

[3] "Describing Linked Datasets with the VoID Vocabulary," http://www.w3.org/TR/void/.

[4] C. Böhm, J. Lorey, and F. Naumann, "Creating void descriptions for web-scale data," *J. Web Sem.*, vol. 9, no. 3, 2011.

[5] S. Nestorov, S. Abiteboul, and R. Motwani, "Extracting Schema from Semistructured Data," in *SIGMOD Conference*, 1998.

[6] S. Khatchadourian and M. P. Consens, "ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud," in *ESWC (2)*, 2010.

[7] M. Konrath, T. Gottron, S. Staab, and A. Scherp, "Schemex - efficient Construction of a Data Catalogue by Stream-Based Indexing of Linked Data," *Journal of Web Semantics*, vol. 15, to appear.

[8] M. Kirchberg, E. Leonardi, Y. S. Tan, S. Link, R. K. L. Ko, and B.-S. Lee, "Formal concept discovery in semantic web data," in *ICFCA*, 2012.

[9] T. Milo and D. Suciu, "Index structures for path expressions," in *ICDT*, 1999.

[10] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes, "Exploiting local similarity for indexing paths in graph-structured data," in *ICDE*, 2002.

[11] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth, "Covering indexes for branching path queries," in *SIGMOD Conference*, 2002.

[12] C. Qun, A. Lim, and K. W. Ong, "D(k)-index: An adaptive structural summary for graph-structured data," in *SIGMOD Conference*, 2003.

[13] T. Tran, G. Ladwig, and S. Rudolph, "RDF Data Partitioning and Query Processing Using Structure Indexes," *IEEE Transactions on Knowledge and Data Engineering*, 2012, to appear.

[14] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, and C. T. dos Santos, "Ontology alignment evaluation initiative: Six years of experience," *J. Data Semantics*, vol. 15, 2011.

[15] A. Algergawy, S. Massmann, and E. Rahm, "A clustering-based approach for large-scale ontology matching," in *ADBIS*, 2011.

[16] R. Milner, *Communication and concurrency*, ser. PHI Series in computer science. Prentice Hall, 1989.

[17] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[18] S. Blom and S. Orzan, "A distributed algorithm for strong bisimulation reduction of state spaces," *STTT*, vol. 7, no. 1, 2005.

[19] P. C. Kanellakis and S. A. Smolka, "Ccs expressions, finite state processes, and three problems of equivalence," in *PODC*, 1983.

[20] J. Hellings, G. H. L. Fletcher, and H. J. Haverkort, "Efficient external-memory bisimulation on dags," in *SIGMOD Conference*, 2012.

[21] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "SP^{2}Bench: A SPARQL Performance Benchmark," in *ICDE*, 2009.

[22] Y. Guo, Z. Pan, and J. Heflin, "Lubm: A benchmark for owl knowledge base systems," *J. Web Sem.*, vol. 3, no. 2-3, 2005.

[23] C. Bizer and A. Schultz, "The berlin sparql benchmark," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 2, 2009.

[24] K. Möller, T. Heath, S. Handschuh, and J. Domingue, "Recipes for semantic web dog food - the eswc and iswc metadata projects," in *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, November 2007.

[25] U. Lösch, S. Bloehdorn, and A. Rettinger, "Graph Kernels for RDF Data," in *ESWC*, 2012.

[2]http://data.semanticweb.org/ns/swc/swc_2009-05-09.html