SchemEX—Web-Scale Indexed Schema Extraction of Linked Open Data

(BTC Submission)

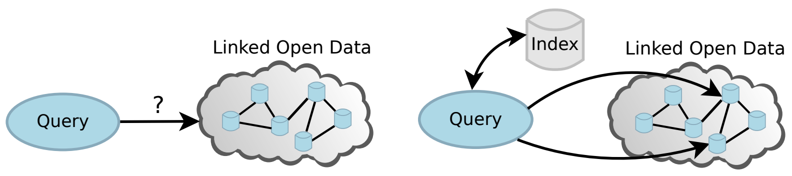
Mathias Konrath, Thomas Gottron, and Ansgar Scherp

WeST – Institute for Web Science and Technologies
University of Koblenz-Landau, 56070 Koblenz, Germany
{mkonrath,gottron,scherp}@uni-koblenz.de

Abstract. We present SchemEX, an approach and tool for web-scale, real-time indexing and schema extraction of Linked Open Data (LOD) at linear runtime complexity. As we cannot assume that a complete retrieval of the LOD cloud on a local machine is feasible, we follow a stream-based approach that makes no assumption about how the RDF triples are retrieved from the web by a data crawler. We show the applicability of our approach by applying SchemEX to the Billion Triple Challenge Dataset 2011 and a smaller dataset with 11M triples.

1 Introduction

Linked Open Data (LOD) [5] is the collective term for publishing open data using the Resource Description Framework (RDF). Data is provided by different, connected data sources building a huge web-scale RDF graph. The LOD graph does not provide a single federated interface to perform graph queries as shown in Figure 1(a). A complete retrieval of this huge RDF graph with currently more than 31 billion triples (`http://lod-cloud.net/`) to perfom queries locally is not feasible. Thus, one of the challenges when working with LOD is the lack of a concise summary or description of what kind of data can be found in which data source and how these data sources are connected. Such a summary is desirable for solving many tasks in common LOD scenarios, like searching, browsing, exploring or querying the LOD graph. Typical tasks in these scenarios are, e.g., to find data sources that contain instances with certain properties, to detect which data sources are interlinked and to support the execution of distributed queries. To address this issue some datasets provide a voiD



(a) (b)

Fig. 1. Index usage for identifying relevant data sources

description [1], containing metadata information such as a SPARQL endpoint location (`void:sparqlEndpoint`), example resources (`void:exampleResource`), and compositional relationships between different parts of the data source using `dcterms:hasPart` and `dcterms:isPartOf`. However, to date only a small fraction of datasets are publishing voiD descriptions, mostly triple store datasets such as DBPedia. Further, voiD does not contain explicit schema information and is not sufficient to support all of the above tasks. Therefore, a more precise index structure is desirable that allows to search for data sources that contain instances with certain properties or to identify relevant data sources for a given query.

Our solution to this problem is to extract a concise schema from the LOD graph with a suitable structure to be used as an index. In our context, schema extraction means to abstract RDF instances to RDF schema concepts that represent instances with the same properties. For using this schema as an index, each schema concept maps to data sources that contain instances with corresponding properties. This allows to search for data sources that contain instances with certain properties or to find relevant data sources for given queries. Such a pre-processed schema-level index cannot process queries itself. However, it supports identifying the relevant data sources as shown in Figure 1(b), thereby operating as service provider to find data sources.

In this paper, we introduce an enhanced index structure called SchemEX that leverages RDF typings and links for creating a graph-based web-scale schema-index. SchemEX uses a fixed-window approach for schema extraction and index building operating on a stream of RDF triples. This allows to index without persistently storing the data and an effective integration with a Linked Data crawler such as LDSpider [6]. To demonstrate the scalability of SchemEX and the capability to handle web-scale data in web-quality the full Billion Triple Challenge 2011 dataset was processed. Additionally, a crawl of 11M triples has been executed with LDSpider used for a detailed qualitative evaluation.

2 The SchemEX Approach

Several approaches and algorithms have been developed for extracting schemata information from general semi- or unstructured graphs [3,7]. In the context of Linked Open Data and Semantic Web, different approaches for schema extraction and building indexes have been developed. For example, [4] uses a QTree structure to identify relevant data sources for a given query. This is done by adding triples to corresponding buckets in the QTree. Queries are answered by finding relevant buckets for each query triple pattern and detecting overlaps between them. An approach for creating voiD descriptions [2] from web-scale datasets uses the Map-Reduce paradigm for identifying voiD datasets and links between them. The created voiD descriptions can be used for supporting distributed queries. Another approach is to use n-bisimulation for building a structure index [8]. Bisimulation defines an equivalence relation on instances based on equal sets of edge labels as the resources share the same outgoing RDF properties. The equivalence classes induced by this relation build the index structures that are mapped to individual instances. In this setting, bisimulation does not regard RDF typings that are usually existing and available in LOD.

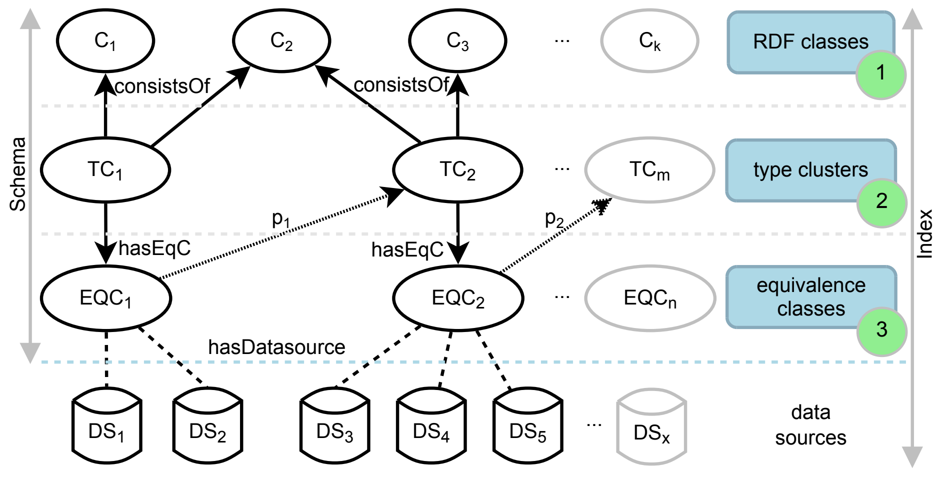


Fig. 2. Enhanced index structure with two additional layers leveraging RDF typings

While our index structure is based on equivalence classes induced by bisimulation, it extends previous approaches under several aspects. On one hand, our index structure adds two further schema layers to leverage RDF typings. On the other hand, we use these additional layers to adapt the bisimulation itself to effectively incorporate RDF type information into the equivalence classes. Thereby, the index is extended to support a wider range of query types and improve and refine the results of type-selecting queries.

An example of the enhanced index structure underlying SchemEX is shown in Figure 2. The index consists of three schema layers that reference the data source layer. The entries in the schema layer capture different schema information targeted at different types of queries. The schema information attached to the entries is defined by class and link patterns observed in the RDF instances. The index then maps the elements of the layers to those data sources that actually provided the RDF data complying with these patterns.

The first layer captures schema information defined by the set Γ of individual *RDF classes* (e.g. `foaf:Person`) found in the dataset, indicated by C_1, C_2, and C_3 in Figure 2. This allows for supporting queries that select instances of a specified, single RDF class. Formally, we can map the elements C_j in this layer to data sources containing instances of those classes.

The second layer describes *type clusters*. Each type cluster TC is an element in the power set $\mathcal{P}(\Gamma)$ over the RDF classes in layer 1. To be more efficient, we use only those type clusters in the index that are actually observed in the data. The type clusters are mapped to those data sources providing instances that belong to a precise set of RDF classes, such as TC_1 and TC_2 denoted in Figure 2.

The third layer corresponds to *equivalence classes* extracted by bisimulation. Here, the type clusters are additionally partitioned into equivalence classes defined by outgoing RDF links and the type cluster of the target instances. Literals are derived and mapped to corresponding XML schema data types in the schema. This way also properties with literals can be processed. These equivalence classes allow to query over RDF properties with and without constraints on the RDF classes of subject and object. The advantage of this approach over a complete bisimulation is that it avoids the very high number of equivalence classes that was observed in [8].

The presented index can be built given access to the complete LOD cloud. However, this involves knowledge about the RDF type information for each resources as well as for each referenced resource. Thus, this approach does not scale to an arbitrary amount of data sources. Therefore, we have developed a scalable stream-based approach for schema extraction called SchemEX. The stream-based processing allows to supply the schema extractor directly with a quadruple stream (RDF triple + context/provenance URI) of a LOD crawler. To reach web-scale performance and a moderate memory consumption only a window of the stream with a certain width is observed and buffered by the schema extractor. This approach leverages the characteristic that a crawler traverses the graph from resource to resource via RDF properties so that linked instances are following in a certain distance within the stream. The buffer is implemented as a FIFO-cache (first in, first out) of RDF instances with a fixed size. Each incoming quadruple is parsed and assigned to the subject instance in the cache. If the instance does not exist in the cache yet, an instance object is created and appended to the FIFO-cache. If the cache is full, the oldest instance is removed, its schema information with respect to the three layers is derived and incorporated in the index. Due to this stream-based approach, SchemEX has by design a linear runtime complexity.

The restriction to a certain window size of the data stream typically leads to incomplete results. This happens, if information about a single RDF instance is separated in the stream by triples referring to a number of instances larger than the cache size. In this case, the instance has already been removed from the cache when we see additional information potentially relevant for the schema. This occurs in particular when assigning an instance to an equivalence classes as it involves knowledge about the type clusters of all linked resources. Hence, the essential parameter for this approach is to choose an appropriate cache size value to obtain a certain quality level for the extracted schema.

3 Computing the SchemEX

We considered two datasets for different evaluation purposes. The full BTC dataset is used in Section 3.1 to show the applicability of the SchemEX approach on a web-scale data. A smaller dataset is introduced and used in Section 3.2 to determine the quality of the index and schema extraction process.

3.1 BTC Dataset Schema Extraction

To show that SchemEX can handle web-scale data of realistic web-quality, we have extracted a schema from the Billion Triples Challenge 2011 dataset. We have processed the BTC data in two parts as well as the entire dataset. For our experiments we have applied a cache size of 50K instances. Table 1 shows the statistics of the schema extraction process and the extracted schemata. Each chunk of 10M triples has been processed with an average of ca. 250 seconds using a single Intel Xeon CPU core with 2.93 GHz

	1st billion	2nd billion	full dataset
#triples	1 billion	1 billion	2.17 billion
#instances	187.7M	222.6M	450.0M
#data sources	13.5M	9.5M	24.1M
#type clusters	208.5k	248.5k	448.6k
#equivalence classes	0.97M	1.14M	2.12M
#triples index	29.1M	24.8M	54.7M
Compression ratio	2.91%	2.48%	2.52%
runtime (hh:mm)	6:51	6:05	15:16
average runtime per 10M chunk	247 s	219 s	252 s
standard deviation	80 s	12 s	57 s
#triples/sec.	40.5k	45.6k	39.5k

Table 1. Characteristics of the BTC 2011 dataset (cache size = 50K instances)

and 4 GB of RAM. This allows to process 1 billion triples in <7 hours and the full dataset in approximately 15 hours.

The processing time demonstrates that SchemEX is suitable for real time processing. Using commodity hardware, we have managed to realize a throughput of about 40k triples per second. Thus, SchemEX can easily handle the stream generated by a LOD crawler (we observed LDSpider providing ca. 2k triples per second). Further, our stream-based approach has been proved to be able to handle BTC data and can in principle be extended to an arbitrary amount of data.

Having such a schema for a very large dataset crawled from the web allows us to provide a look-up service for linked data clients to find relevant data sources on the web of data. The schema extracted from the full BTC data is made available on the web at `http://west.uni-koblenz.de/schemex/`.

By applying SchemEX on the BTC dataset, we have demonstrated the effectiveness and efficiency of our schema extraction process. In the following section, we investigate the quality of the extracted index, in particular with respect to the cache size.

3.2 Qualitative Evaluation of SchemEX on the TimBL Dataset

There are different query types that can be answered by the SchemEX index and that target the different schema layers. These query types are of different complexity and are affected to a different degree by incomplete schema information.

The simplest type of query retrieves all data sources providing instances of a certain type, therefore we refer to them as *RDF-Class* queries. Slightly more information is needed to obtain those data sources providing instances which belong to a certain type cluster (*TC queries*). An extension of these TC queries target a set of RDF classes by considering all type clusters and super-type clusters that contain the RDF classes (*TC+S-TC*). A super-type cluster is a type cluster whose RDF classes contain a subset of classes of another existing type cluster. The next type of query makes use of the third schema layer and addresses equivalence classes. Equivalence classes can be selected just by RDF properties (*EQC*) or in combination with type-selection constraints

(*EQC+TC*). The latter query type is the most complex type and targets all schema layers.

To evaluate index quality, we have constructed a gold standard. Therefore a complete schema regarding the index structure was extracted from a smaller collection of LOD data: the TimBL dataset. This dataset was crawled by Andreas Harth with LDSpider starting at Tim Berner-Lee's FOAF file and aborted at a size of 11M triples. As said in Section 2, our schema can be computed lossless without constraint on the window size in the stream-based approach. Such a lossless schema provides the gold standard to compare with the schema extracted by the SchemEX tool. The gold standard also provides all combinations of RDF classes, type clusters and equivalence classes present in the dataset. We compare all of these combinations in the gold standard with the schema extracted by SchemEX. Thereby we can evaluate recall and precision over the relevant data sources for virtually all possible queries on the dataset.

The gold standard on the TimBL dataset provides a total of ca. 674K instances in the data graph, which are mapped to 2,763 type clusters and ca. 12K equivalence classes. Type clusters are partitioned by an average of 4.33 and up to a maximum of 1,071 equivalence classes of common types like `foaf:Person`.

This evaluation of the quality of SchemEX is performed with respect to both the different query types and variable cache sizes. We ran the SchemEX approach with window sizes from 100 up to 100,000 instances and compared the generated index to the gold standard. Table 2 shows the statistics and values of the schema extraction process and the generated schemata. For each cache size, ten runs in randomized order were done. In addition to the SchemEX-extracted schemata the numbers of the gold standard schema are shown.

	100	**1K**	**10K**	**50K**	**100K**	**Gold/700k**
Runtime	182 s	223 s	192 s	194 s	203 s	376 s
Standard deviation	2.57 s	2.83 s	1.52 s	2.42 s	2.01 s	4.39 s
#triples/sec.	60.4k	49.3k	57.3k	56.7k	54.2k	29.3k
Max. memory consumption	84 MB	136 MB	315 MB	731 MB	874 MB	3393 MB
#type cluster	2772	2751	2749	2757	2761	2763
#equicalence classes	13570	12885	12281	12062	12184	11955
#triples	270871	241187	246396	255751	263916	277695

Table 2. Schema extraction processes and results in dependency of window size (average of 10 runs)

As expected, the window size does not have a strong influence on the runtime performance. Regarding the maximum memory consumption, an increased window size by a factor of 10 results only in a 2 to 3 times higher memory consumption. This can be attributed to observing tendentially more triples about the same instances within a larger window size.

The second part of the table shows values regarding the extracted schemata. The higher the window size, the closer the values are to the gold standard. As this is merely a quantitative analysis, it is not clear that the counted type clusters and equivalence classes are exactly the same.

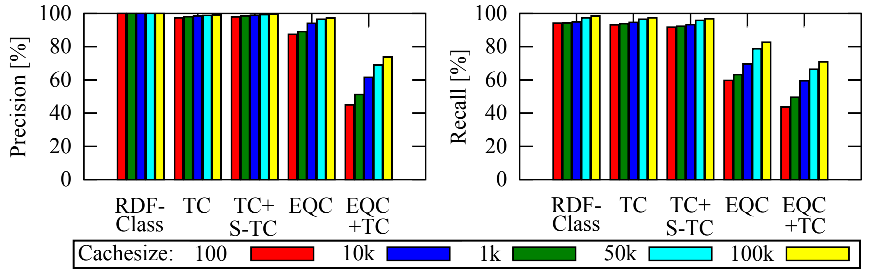


Fig. 3. Precision and recall for different query types in dependency on cache window size

Figure 3 shows precision and recall regarding the five query types introduced above. For each query type the values for different window sizes are shown. For the simpler type selecting queries (RDF-Class, TC and TC+S-TC) both precision (>98%) and recall (>91%) show very good results at any window size. On the more complex EQC and EQC+TC type queries precision and recall are lower. However, with an increasing window size also recall and precision can be boosted to more than 70% for a window size of 100K instances. Based on this evaluation, we chose a window size of 50K instances for processing the BTC dataset.

Regarding the evaluation results, it has to be considered that even wrongly classified instances are mostly assigned to very similar equivalence classes. Those wrongly classified instances may be returned correctly by EQC and EQC+TC queries as they do not put constraints on all properties of an equivalence class.

4 Conclusion

We have presented an effective and efficient index structure for supporting distributed queries on Linked Open Data. Our SchemEX approach for schema extraction provides a good trade-off between scalability and result quality. Its stream-based processing allows the easy integration with a LOD crawler. Future work will target on evaluating and optimizing crawling strategies for gaining even better results. In addition, other cache strategies could be evaluated for optimization purposes. As the developed index structure cannot deal with instance level queries, extending it in this direction would be of interest. Finally, SchemEX can be integrated with a federated query processing system and direct subqueries to relevant distributed data sources.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the void vocabulary. `http://www.w3.org/TR/void/` (May 2011), visited 30.08.2011
2. Böhm, C., Lorey, J., Naumann, F.: Creating void descriptions for web-scale data. Web Semantics: Science, Services and Agents on the World Wide Web (2011)
3. Goldman, R., Widom, J.: Dataguides: Enabling query formulation and optimization in semistructured databases. In: VLDB. pp. 436–445. Morgan Kaufmann (1997)

4. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data summaries for on-demand queries over linked data. In: WWW. pp. 411–420. ACM (2010)
5. Heath, T., Bizer, C.: Linked Data: Evolving the Web Into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool (2011)
6. Isele, R., Harth, A., Umbrich, J., Bizer, C.: LDspider: An open-source crawling framework for the Web of Linked Data. In: Poster, ISWC2010; Shanghai, Chinam (2010)
7. Nestorov, S., Abiteboul, S., Motwani, R.: Extracting schema from semistructured data. In: Haas, L.M., Tiwary, A. (eds.) SIGMOD Conference. pp. 295–306. ACM Press (1998)
8. Tran, T., Haase, P., Studer, R.: Semantic search - using graph-structured semantic models for supporting the search process. In: ICCS. pp. 48–65. Springer (2009)

Addressing the Billion Triples Challenge Evaluation Requirements

Minimal requirements

Web Scale: SchemEX works on web-scale data. We have demonstrated this by applying our index structure and tool on the Billion Triples Challenge (BTC) 2011 Data Set in different ways. First, we have applied SchemEX on the first and second half of the data set. Finally, it has been applied on the entire data set.

Realistic Web-quality Data: SchemEX works with realistic web-quality data, i.e., it can cope with the specifics of data from the web as it does not make any assumption how and in which order the triples are retrieved for the stream-based indexing and schema extraction. This is shown by applying the SchemEX tool to the BTC data set as well as to a smaller data set of 11M triples crawled from the data web.

Additional Desirable Features

More than Simply Store/Retrieve Large Numbers of Triples: SchemEX can be considered as service provider for discovering data sources that contain specific linked data instances. This is achieved by extracting an index and schema of linked open data at web scale using a stream-based approach.

Scalable in Terms of Amount of Data: Following a stream-based approach, SchemEX can in principle process an arbitrary amount of linked data. Thus, the amount of data that is processed by SchemEX is only limited to how the triples are provided by a linked data crawler.

Scalable in Terms of Distributed Components: In principle, SchemEX can work together with more than one linked data crawler at the same time, i.e., process triples coming from multiple crawlers. Also multiple SchemEX threads are conceivable, extracting a common schema.

Use of Very Large, Mixed Quality Data: For our experiments, we have used parts and the entire BTC data set as well as an own, smaller data set of 11M triples crawled from the web and containing data of different origin and quality.

Function in Real-time: Our SchemEX approach can process $\sim$40k triples per second on a standard single CPU machine with 4 GB RAM. Thus, it is able to serve the data stream crawled by a data crawler such as LDSpider (providing about $\sim$2k triples/s).