Beyond RDF Links – Exploring the Semantic Web with the Help of Formal Concepts

Markus Kirchberg[1], Erwin Leonardi[1], Yu Shyang Tan[1], Ryan K L Ko[1], Sebastian Link[2], and Bu Sung Lee[1]

[1] Cloud & Security Lab, Hewlett-Packard Laboratories, Singapore
[2] Dept of Computer Science, Auckland University, New Zealand

Abstract. Linked Open Data (LOD) has seen a tremendous rise in popularity. However, with rapid growth, new challenges (e.g., compliance, computational complexity, tool support, usability) arise. The Billion Triple Challenge has been set up to encourage development of approaches to address such short-comings. In this paper, we discuss how Formal Concept Analysis and explorative data visualisation can be applied (at scale) to a random LOD snapshot. By adding a formal concept layer to the Semantic Web, we enable exploration of LOD data-sets, support query refinement, data cleaning, concept clustering, and more.

1 Introduction

The Semantic Web is the most prominent effort to address limitations underlying the design of the World Wide Web (WWW); it aims to bring the WWW to a state in which all its content can also be interpreted by machines. The emphasis is not primarily on putting data on the Web, but rather on creating links in a way that both humans and machines can explore the Web. Accordingly, the term Linked Open Data (LOD) was coined [1]. LOD builds upon standard Web technologies, but extends them to share information in a way that data from different sources can be connected (via RDF links) and queried (via SPARQL).

The Resource Description Framework (RDF) is the basic format of data in the Semantic Web and LOD; it consists of statements of the form: "*Subject S is in some relation (P) to object O.*". Such statements are known as RDF triples (or *triples* for short) that can be serialised in N-Triple format. Recently, the notion of provenance (or *context*) is added in the triples, and hence, "*Given context C, Subject S is in some relation (P) to object O.*". Triples with context are called RDF quads (or *quads*). Quads can be modelled in N-Quads format, an extension of N-Triples with context. Each quad has the following form:

```
<subject S> <predicate P> <object O> <context C> .
```

LOD has seen a tremendous rise in popularity over the past few years; as of August 2011, LOD consists of 295 officially acknowledged data-sets; spans domains such as media, geographic, government (largest wrt. triples), publications

(largest wrt. number of data-sets), cross-domain, life sciences (largest wrt. outlinks) and user-generated content [2]. Similar to traditional hyper-links, RDF links (e.g., triples or quads) support navigation from a data item within one data source to related data items within the same or another sources using a Semantic Web browser. In contrast to the traditional Web, RDF links can also be followed by the crawlers of Semantic Web search engines.

However, LOD's rapid growth, varying data-set compliance wrt. LOD guidelines, complexity of computations over large-scale data-sets, and lack of tools and applications pose significant challenges to the Semantic Web community. How to select the right LOD data-set(s) that contain(s) the data of interest? Even once suitable LOD data-sets have has been found, how to understand their interlinking, schemata (i.e., ontologies), and contents? The annual Billion Triple Challenge (BTC) has been set up to encourage the development of ideas and Semantic Web approaches to address such short-comings (at scale).

1.1 Contribution

In this paper, we discuss how Formal Concept Analysis (FCA) [3], and explorative data visualisation can be applied (at scale) to a random snapshot of the Semantic Web. In order to facilitate efficient and effective storage of and access to more than 2 billion quads (that make up the BTC data-set), we have deployed a Map-Reduce and Column-store-driven framework. This framework also supports RDF data extraction and data format conversion to enable timely formal concept system generation. Parallel FCA algorithms [4] have been extended to support formal concept extraction from quads, concept clustering, and interactive concept lattice visualisation and exploration. Users can specify their (initial) intentions through an interactive user interface; a sub-set of the BTC data-set is selected by specifying constraints / restrictions on quads. Once formal concepts have been extracted (in real-time from quads or based on precomputed formal concept systems), a D3-driven interactive visualisation Web interface allows the users to explore a data set through its embedded formal concepts. While our current approach does not yet include the ability to chain RDF links, run SPARQL queries etc. it is sufficient to demonstrate value-adding benefits in terms of adding a formal concept layer to the Semantic Web.

2 Related Work

Formal Concept Analysis (FCA) [3] is a theory of data analysis identifying conceptual structures among data sets. Given a (formal) context, which consists of a set of objects O, a set of attributes A, and an indication of which objects have which attributes, a concept can be defined as a pair (O_i, A_i) such that (1) $O_i \subseteq O$; (2) $A_i \subseteq A$; (3) every object in O_i has every attribute in A_i; (4) for every object in O that is not in O_i, there is an attribute in A_i that the object does not have; and (5) for every attribute in A that is not in A_i, there is an object in O_i that does not have that attribute. We also call O_i the extent and

Number of quad s	2,178,395,469
Number of unique subjects (excl. blank nodes)	13,011
Number of blank nodes appearing in subjects	1,640,600,663
Number of unique predicates	47,133
Number of unique objects (excl. blank nodes and literals)	534,156
Number of literals appearing in objects	581,688,407
Number of blank nodes appearing in objects	514,343,502
Number of unique contexts	6,506

Table 1. BTC 2011 Data-set Statistics

A_i the intent of a context. Concepts (O_i, A_i) for a given context can be partially ordered by inclusion: if (O_i, A_i) and (O_j, A_j) are concepts, we define a partial order $(O_i, A_i) \leq (O_j, A_j)$ if $O_i \subseteq O_j$ (which implies $A_j \subseteq A_i$). Given $(O_i, A_i, \leq)$, we can determine its corresponding concept lattice – a classification system.

From a concept system, we can identify the following properties (encoded in its concept lattice): Set of objects that share common attributes and vice versa (some concept); more general concepts (at the lattice's top) and more specific concepts (at the lattice's bottom); concepts with high/low support (number of objects in a concept); most similar concepts with object(s) added/removed and attribute removed/added (labelled lattice edges); conjunction of concepts (objects with common attributes); disjunction of concepts (attributes for common objects); implications (every object that has attribute a_1 also has attribute a_2; i.e., a_1 implies a_2); concept clustering (grouping of similar concepts); etc. FCA has been applied to many fields including medicine, psychology, databases, information sciences, software re-engineering, civil engineering, etc. Most related to our work are results from Krajca et.al. on parallel FCA algorithms [4] and Map-Reduce-based FCA concept computation [5], and from d'Aquin et.al. on extracting representative questions over a given RDF data-set utilising FCA [6]. While [5] also deployed Hadoop, the open-source Map-Reduce framework, in computing their formal concepts generation algorithm, in our work, Map-Reduce is used extensively throughout the whole work-flow. This enables us to process the entire BTC 2011 raw dataset in reasonable amount of time.

3 The 2011 Billion Triple Challenge Data-set

The BTC 2011 data-set represents a snapshot of LOD; crawled in May/June 2011 and seeded with a random sample of URIs from the BTC 2010 data-set. Table 1 provides a brief overview of key statistics of the given data-set, while Figure 1 depicts the degree of connectivity contained in the random LOD sample.

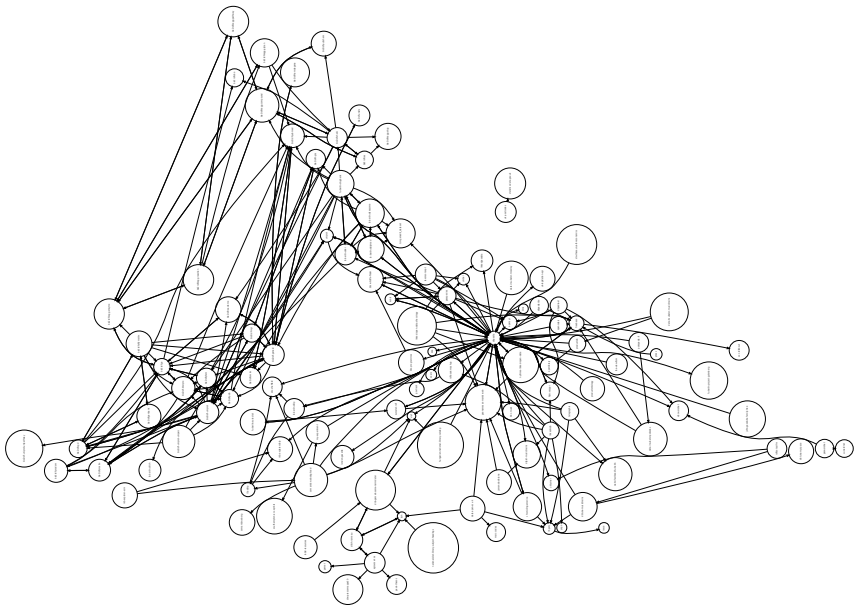

Fig. 1. Connectivity of Name-spaces Covered in the BTC 2011 Data-set.

4 A Framework for Formal Concept Generation and Exploration

Our framework is depicted in Figure 2. In this section, we will briefly explain how this system can be utilised. We will explain how (1) requests are specified and then mapped to quads; (2) different processing pipelines are used to extract subset of the BTC 2011 data set and convert them to a standard FCA format; (3) parallel algorithms are used to compute formal concepts and relationships between them; and (4) visualisation tools can be utilised to enable users to explore the formal concept that corresponds to their original request.

4.1 User Requests and Mapping to RDF

Referring to Figure 2 (Step 1), the user first selects an LOD name-space or vocabulary. A name-space selection will then require the user to specify his/her intentions, e.g., a query, as SPO expression (with an optional constraint on C) while a vocabulary selection will require a SOC expression.

Let us assume that the user selects an LOD name-space (vocabulary selection proceeds analogously; only difference being that queries are SOC expressions). The user then simply has to key in (Figure 2 (Step 2)) an expression of the form:

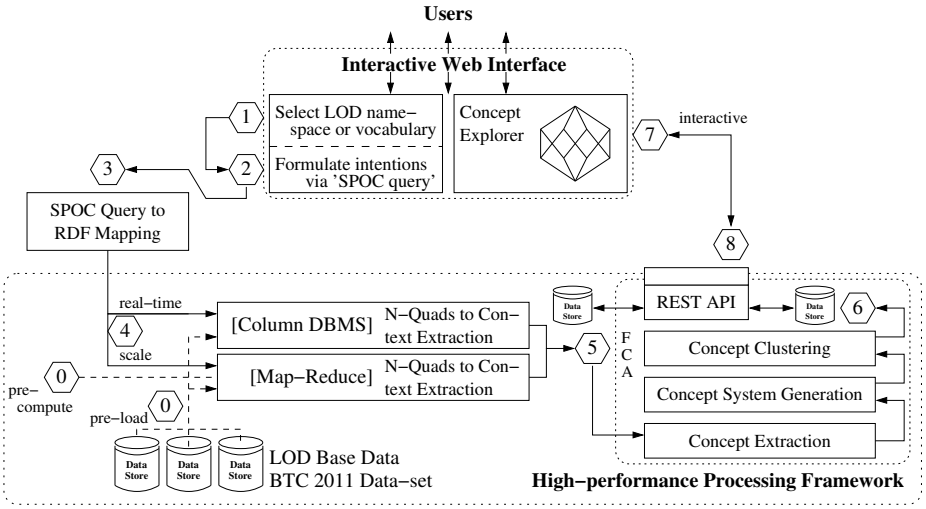

Fig. 2. Framework and Common Work-flow.

Type of Input	**Examples (where c_i is a constraint)**
Complete	$?[SP]?[O]C(c_1)$, $?SP?O$, $?S?[PO]$, $?S(c_1)P(c_2)?O(c_3)$, etc.
Partial w/o Context	$?O?P$, $?P?C$, $?P(c_1)?S$, $?P(c_1)?S(c_2)$, $?S?P$, etc.
Partial w Context	$?P?SC(c_1)$, $?O?PC(c_1)$, $?P?SC(c_1)$, $?S(c_1)?O(c_2)C(c_3)$, etc.

Table 2. Flexible Input Combinations for Name-space Queries

"*List-of-Subjects(optional-Constraint-on-S) List-of-Predicates(optional-Constraint-on-P) List-of-Objects(optional-Constraint-on-O) (optional-List-of-Constraints-on-name-space)*" into the text-boxes provided (LOD base data for the chosen name-space are loaded dynamically providing suggestions as the user is typing). Each expression must contain exactly two '?' symbols indicating the combination of $SPOC$ components to be projected into an FCA context (the mapping is depicted in Figure 2 (Step 3)). There can be many combinations expressed; some examples are shown in Table 2. By default, we consider lists containing multiple elements as disjunctions and an $SPOC$ expressions as a conjunction.

For example, assume the user selects name-space *diseasome* and intends to explore all $S - O$ pairs representing diseases and possible drugs used to fight them. As such, the user would specify a complete request of type $?S(c_1)P?O$: "*?S[disease] P[possibleDrug] ?O[]*" (P only serves as selection criteria).

4.2 Data Storage, Processing Pipelines and FCA Computation

We have two processing pipelines in our framework: a column-store DBMS-driven pipeline and a Map-Reduce-based pipeline. Once a user request is received, the system decides (based on estimates about data size and processing time) whether to deploy a front-to-end real-time processing approach or utilise pre-computed

```
1) Column-store based Pipeline

Function matrixGenerator(extentQ, intentQ, contextQ)

whereClause <- processContext(contextQ)
getExtentSQL <- constructQuery(extentQ, contextQ)
extentList <- execute(getExtentSQL)
getIntentSQL <- constructQuery(intentQ, contextQ)
intentList <- execute(getIntentSQL)
getExtentIntentRelationSQL <- constructQuery(extentQ,
  intentQ, contextQ)
relation <- execute(getExtentIntentRelationSQL)
initialize matrix
for each e in extentList
  relatedIntent <- relation.getIntent(e)
  matrix <- updateMatrix(matrix, e, relatedIntent)
end for
return matrix

2a) Map-Reduce Pipeline: Context Extraction

function Map (Text inputLine, Context jobConf):
  matchStrings <- getMatchStringForEachTuple(jobConf);
  inputTuples <- splitString(inputLine);
  foreach index in matchStrings
    if(inputTuples[index].match(matchStrings[index])
      EMIT(subject,predicate+object+context);
    endif
  end foreach

function Reduce(Text inputKey, Text values, Context out):
  EMIT(NullWritable, inputKey+values);

2b) Map-Reduce Pipeline: Extent/Intent Extraction

function Map(Text inputLine, Context jobConf):
  extent <- extractExtent(inputLine);
  hash <- generateHash(extent);
  EMIT(extent, hash);

function Reduce(Text extent, Text hash, Context output):
  EMIT(extentIndex, hash+extent);

2c) Map-Reduce Pipeline: FCA-Matrix Generation

function Map(Text inputLine, Context jobConf):
  intentIndexArray <- loadIntentString(intentList);
  extentIndexArray <- loadExtentString(extentList);
  extent <- extractExtent(inputLine);
  intent <- extractIntent(inputLine);
  extentIndex <- extentIndexArray.binarySearch(extent);
  intentIntex <- intentIndexArray.binarySearch(intent);
  EMIT(extentIndex, intentIndex);

function Reduce(Text extentIndex, Iterable<Text>
  intentIndex, Context output):
  forEach e in intentIndex
    append e to List;
  end forEach
  EMIT(extentIndex, List);
```

Fig. 3. Data Extraction–to–FCA MIMI Representation Generation.

concept systems for the selected name-space or vocabulary (Figure 2 (Step 4)). Requests extracting small and medium-sized subsets of the BTC data-set are processed in real-time front-to-end using the DBMS-driven pipeline. The column-store DBMS also hosts LOD base data and query suggestions passed to the user interface. The Map-Reduce pipeline processes large requests as well as pre-computations for all name-spaces and vocabularies. Basic algorithms for both pipelines are shown in Figure 3.

Both processing pipelines generate a MIMI-representation of the formal context that matches the input parameters (e.g., a user request); indexes over extent and intent are stored in a DBMS. Subsequently, an extended version of the parallel recursive algorithm from [4] and customised FCA concept system generation, annotation and clustering scripts (Figure 2 (Step 5)) are run computing formal concepts for a given context, $\leq$ concept relationships, concept support values, concept lattice edges and their annotations, and values necessary for concept clustering; all computational results are stored in a DBMS (Figure 2 (Step 6)).

Considering the *diseases* and *drugs* example, we utilise the DBMS-driven pipeline (as specified selections on *diseasome*result in a small data-set). From the BTC data-set, we extract just over $14,000$ quads forming an FCA extent of size $1,456$ and an intent of size $2,235$; concept computation results in 887 concepts (29 of those have a support ≥ 25 and 5 have a support ≥ 50).

4.3 Formal Concept Visualisation

Formal concepts are typically represented as lattice. Given that the number of formal concepts can explode, we provide a lattice view over concepts and concept clusters. Concept clusters are formed dynamically depending on the number of concepts, concept support values and user actions; typically, we aim to limit

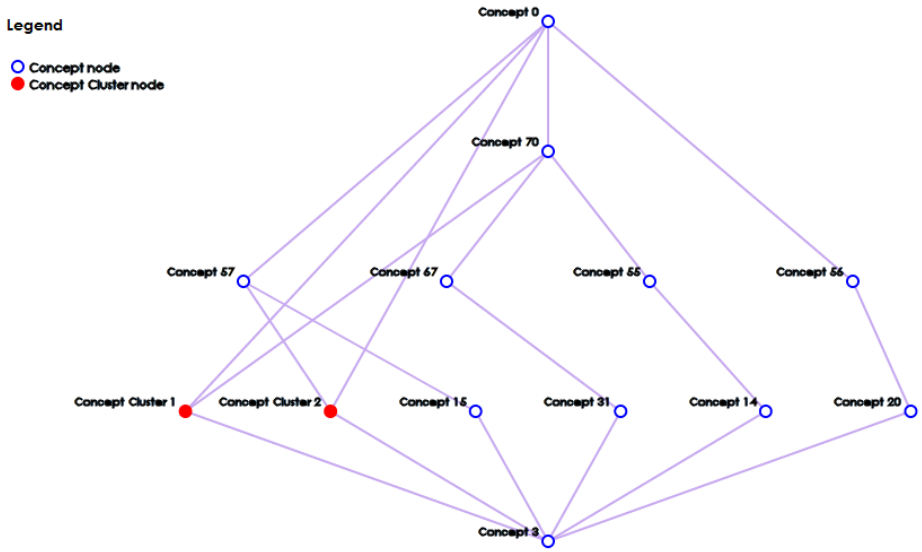

Fig. 4. Visualisation of a Sub-set of the *disease* and *drugs* Example

the number of visible nodes to below 50. Users may zoom into a cluster, trim a branch, view edge annotations etc. (refer to Figure 2 (Steps 7 & 8)). A sample lattice of the *diseases* and *drugs* example is shown in Figure 4.

5 Possible Use-case Scenarios

Users are enabled to specify their (initial) intentions through an interactive user interface. Once formal concepts have been extracted (in real-time from quads or based on pre-computed formal concept systems), a D3-driven interactive visualisation Web interface allows the users to explore a data set through its embedded formal concepts. Adding a formal concept layer to the semantic Web enables interactive exploration of LOD data-sets (via a Formal Concept Explorer), supports query refinement and expansion (e.g., by removing edges / irrelevant concepts from a concept lattice) as well as data cleaning, clustering of similar quad sub-set (by exploring annotated concept lattice edges with minimal changes), defining a new similarity measure between quad sub-sets, and many more.

6 Conclusion

In this paper, we have shown how a formal concept layer can be added to the Semantic Web enabling interactive exploration of LOD data-sets, supports query refinement and expansion, data cleaning, and clustering of similar concepts among other things. Future work includes the provisioning of common Semantic

Web features such as chaining of RDF links and issuing of SPARQL queries as well as extracting RDF Links from the real Semantic Web (instead of a static snapshot as given in the BTC 2011 challenge).

Supplementary information about our work and access to the prototype will be made available via http://203.30.38.230/ (from Oct 17, 2011 onwards).

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int'l Journal on Semantic Web and Information Systems **5**(3) (2009) 1–22
2. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the LOD cloud. Available online; http://www4.wiwiss.fu-berlin.de/lodcloud/state/ (2011)
3. Priss, U.: Formal concept analysis in information science. Annual Review of Information Science and Technology **40**(1) (2006) 521–543
4. Krajca, P., Outrata, J., Vychodil, V.: Parallel recursive algorithm for FCA. In: Proc. of the 6th Int'l Conference on Concept Lattices and Their Applications (CLA). Vol. 433., CEUR WS (2008) 71–82
5. Krajca, P., Vychodil, V.: Distributed algorithm for computing formal concepts using map-reduce framework. In: Proc. of the 8th Int'l Symposium on Intelligent Data Analysis (IDA), Springer-Verlag (2009) 333–344
6. d'Aquin, M., Motta, E.: Extracting relevant questions to an rdf dataset using formal concept analysis. In: Proc. of the 6th Int'l Conference on Knowledge Capture (K-CAP), ACM (2011) 121–128

A BTC Evaluation Criteria

As per ISWC 2011 Semantic Web Challenge specifications, our submissions meets the following minimal requirements and additional desirable features:

- Our prototype uses the entire BTC 2011 data-set. Formal concepts can be computed based on name-spaces, vocabularies or combinations of these.
- Primary focus is on the BTC 2011 data-set; we only utilise some additional LOD data sources and ontologies to capture common knowledge about the LOD cloud (e.g., LOD nodes, name-spaces and vocabularies as per [2]).
- Usability criteria are met by (1) using an interactive Web interface to allow users to specify their intentions; (2) exposing REST-ful APIs to query computational results; and (3) providing a visual, interactive interface to enable explorative navigation of a BTC sub-set matching a user's intentions.
- (add. criteria: beyond simple store & retrieve) Formal concepts are computed for each subset extracted from the BTC data-set; interactive exploration is based on such concepts formed of quads.
- (add. criteria: scalability) We utilise Column-DBMS, Map-Reduce, and parallel FCA algorithms in a HPC cluster setting to allow for scalability.
- (add. criteria: large, mixed quality data set) No prior data cleaning or data massaging has been applied to the BTC data-set.
- (add. criteria: real-time) Whenever feasible, computation takes place in real-time. If a large BTC sub-set is requested, a real-time realisation based on extracting pre-computed sub-concept lattices is provided.