SWEEP: a Streaming Web Service to Deduce Basic Graph Patterns from Triple Pattern Fragments

Emmanuel Desmontils, Patricia Serrano-Alvarado and Pascal Molli

LS2N Laboratory - Université de Nantes – France
{firstname.lastname}@univ-nantes.fr

Abstract. The Triple Pattern Fragments (TPF) interface demonstrates how it is possible to publish Linked Data at low-cost while preserving data availability. But, data providers hosting TPF servers are not able to analyze the SPARQL queries they execute because they only receive and evaluate subqueries with one triple pattern. Understanding the executed SPARQL queries is important for data providers for prefetching, benchmarking, auditing, etc. We propose SWEEP, a streaming web service that deduces Basic Graph Patterns (BGPs) of SPARQL queries from a TPF server log. We show that SWEEP is capable of extracting BGPs of SPARQL queries evaluated by a DBpedia's TPF server.

1 Introduction

The Triple Pattern Fragments (TPF) interface demonstrates how it is possible to publish Linked Data at low-cost while preserving data availability [8]. However, data providers hosting TPF servers are not able to analyze the SPARQL queries executed by their clients because they only receive single triple pattern queries.

Understanding the executed SPARQL queries is fundamental for data providers. Mining logs of SPARQL endpoints allows to detect recurrent patterns in queries for prefetching [1], benchmarking [3], auditing [4], etc. It provides the type of queries issued, the complexity and the used resources [2,6]. Such analysis cannot be done on logs of TPF servers because they only contain information about single triple patterns. A Basic Graph Pattern (BGP) of a SPARQL query, that is a set of conjunctive graph patterns, is scattered over the log.

[7] reported statistics from the logs of the DBpedia's TPF server. However, statistics only concern single triple pattern queries and not BGPs. In previous work [5], we proposed an algorithm to extract BGPs of *federated SPARQL queries* from logs of a *federation of SPARQL endpoints*. Here, we address a similar scientific problem but in the context of a single TPF server.

In this demonstration, we present SWEEP, a streaming web service that is able to extract BGPs from logs of TPF servers in real-time. From the stream of single triple pattern queries of a TPF server, SWEEP is capable of extracting BGPs. This allows data providers running TPF servers to better know how their data are used. The demonstration highlights the performances of SWEEP in terms of precision and recall.

2 Motivating example

In Figure 1, two clients, c_1 and c_2, execute concurrently queries Q_1 and Q_2 over the DBpedia's TPF server. Q_1 asks for movies starring Brad Pitt and Q_2 for movies starring Natalie Portman.[1] Both queries have one BGP composed of several triple patterns (tp_n).

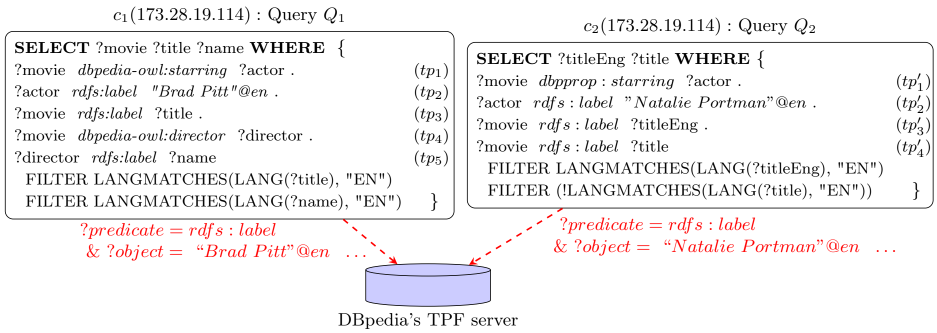

Fig. 1: Concurrent execution of queries Q_1 and Q_2.

	IP	Time	Asked triple pattern/TPF
1	172...	11:24:19	?predicate=rdfs:label & ?object="Brad Pitt"@en
2	172...	11:24:23	**dbpedia:Brad_Pitt** rdfs:label "Brad Pitt"@en ,
3	172...	11:24:24	?predicate=dbpedia-owl:starring & ?object=**dbpedia:Brad_Pitt**
4	172...	11:24:27	**dbpedia:A_River_Runs_Through_It_(film)** dbpedia-owl:starring dbpedia:Brad_Pitt dbpedia:Troy_(film) dbpedia-owl:starring dbpedia:Brad_Pitt ...
5	172...	11:24:28	?subject=**dbpedia:A_River_Runs_Through_It_(film)** &?predicate=rdfs:label

Table 1: Excerpt of a DBpedia's TPF server log for query Q_1.

The TPF client decomposes the SPARQL queries into a sequence of triple pattern queries partially presented in Table 1. The odd-numbered lines represent received triple pattern queries and the even-numbered ones represent sent triples after evaluation on the RDF graph. Lines 1 and 3, correspond to triple pattern queries for tp_2 and tp_1 of Q_1.[2] We can observe that the object in Line 3, comes from a mapping seen in Line 2. This *injection* of a mapping obtained from a previous triple pattern query, is clearly a *bind join* from tp_2 towards tp_1.

As the TPF server only sees triple pattern queries, the original queries are unknown to the data provider. In this work, we address the following research question: *Can we extract BGPs from a TPF server log?*

The main challenge is to distinguish *similar queries*, that is queries whose triple patterns are the same for the TPF server as tp_1 vs tp'_1. In our example, we aim to extract two BGPs from the TPF server log, one corresponding to Q_1, BGP[1]= $\{tp_1.tp_2.tp_3.tp_4.tp_5\}$ and another corresponding to Q_2, BGP[2]= $\{tp'_1.tp'_2.tp'_3.tp'_4\}$.

[1] These queries come from `http://client.linkeddatafragments.org/`.

[2] TPF clients always rename variables as *"subject"* or *"object"*, regardless of how they are named in the original query.

3 SWEEP

SWEEP uses a TPF server log, as the one of Table 1, composed of an unlimited ordered sequence of execution traces organized by IP-address. It considers a fixed-size window sliding over the TPF server log. Window size can depend on the memory available for the streamed log or on the average of known values used as timeout by TPF clients.

We consider a set G of deduced BPGs. Each time a triple pattern query (tpq_i) arrives, SWEEP creates a new $BPG_j \in G$ or updates an existing one.

Suppose G is empty and SWEEP receives tpq_1 =`{?s p2 toto}` where `?s` produces 2 mappings: $\{c1, c2\}$. As G is empty, SWEEP creates BGP_1 containing tpq_1 with the current time as timestamp, $BGP_1.ts = time()$.

Then, if tpq_2 =`{c1 p1 ?o}` arrives, as $c1$ appears in mappings of a $BGP_j \in G$, SWEEP detects a bind join. This implies updating BGP_1 with the join `{?s p2 toto . ?s p1 ?o}`. If tpq_3 = `{c2 p1 ?o}` arrives, as it is already represented in BGP_1, nothing is done.

If BGP_1 is out the window, i.e., $time() - BGP_1.ts > window$, then it must no longer be updated; it is delivered and removed from the stream.

We run SWEEP with queries proposed by the TPF web client (`http://client.linkeddatafragments.org/`). From 21 queries executed, we obtained 100% of precision and 87% of recall of deduced BGPs when compared to the BGPs of corresponding original queries. SWEEP succeeds in this case because these queries are note very *similar*. Different precision and recall would be produced with a more challenging set of queries.

4 Demo

Figure 2 presents the dashboard of SWEEP available at `http://sweep.priloo.univ-nantes.fr`. It shows the most recent deduced BGPs and original client queries when they are available. Our TPF client, `http://tpf-client-sweep.priloo.univ-nantes.fr`, sends the original client query to SWEEP to be able to calculate precision and recall.

If you want to test SWEEP with another TPF client, you must specify the address of the DBpedia's TPF server we have setup: `http://tpf-server-sweep.priloo.univ-nantes.fr`. In this case, SWEEP will deduce BGPs but will not be able to calculate precision and recall.

We used, the versions of JavaScript for Node.js of the TPF server and client. The source code is available at `https://github.com/edesmontils/SWEEP`.

5 Conclusion and perspectives

SWEEP demonstrates how it is possible to deduce the BGPs executed by a TPF server. This allows data providers to have a better understanding of the usage of their data.

With SWEEP it would be possible to detect whether clients are executing federated queries over multiple datasets hosted by one TPF server. And if multiple data providers agree on streaming their logs to a shared SWEEP service, they would be able to detect federated queries executed over multiple TPF servers.

INFORMATION

Gap (hh:mm:ss)	Evaluated Queries	BGP	TPQ	Avg Precision	Avg Recall
0:01:30	3 / 3	9	270	1.000	1.000

DEDUCED BGPS

(20 more recents)

	ip	time	bgp	Original query	Precision	Recall
9	127.0.0.1	2017-07-27 18:33:50.460040	?jo0 <http://www.w3.org/2000/01/rdf-schema#label> "Belgium"@en . ?jo1 <http://dbpedia.org/ontology/locationCountry> ?jo0 . ?se18777_7484 <http://dbpedia.org/ontology/developer> ?jo1 .	qsim-WS-127.0.0.1-84_1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT ?software ?company WHERE { ?software dbpedia-owl:developer ?company. ?company dbpedia-owl:locationCountry [rdfs:label "Belgium"@en]. }	1.000	1.000
8	127.0.0.1	2017-07-27 18:33:35.535898	?js0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Book> . ?js0 <http://dbpedia.org/ontology/author> ?oe18782_7064 .	qsim-WS-127.0.0.1-83_1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> SELECT DISTINCT ?book ?author WHERE { ?book rdf:type dbpedia-owl:Book; dbpedia-owl:author ?author. } LIMIT 100	1.000	1.000
7	127.0.0.1	2017-07-27 18:33:15.094616	?js0 <http://dbpedia.org/property/cityServed> <http://dbpedia.org/resource/Italy> . ?js0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Airport> .	qsim-WS-127.0.0.1-82_1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> PREFIX dbpprop: <http://dbpedia.org/property/> SELECT DISTINCT ?entity WHERE { ?entity a dbpedia-owl:Airport; dbpprop:cityServed dbpedia:Italy. }	1.000	1.000
6	127.0.0.1	2017-07-27 18:33:51.133406	?se18779_7052 <http://dbpedia.org/ontology/developer> ?oe18779_7052 .	No query assigned		

Fig. 2: SWEEP dashboard.

References

1. J. Lorey and F. Naumann. Detecting SPARQL Query Templates for Data Prefetching. In *ESWC Conference*, 2013.
2. K. Möller, M. Hausenblas, R. Cyganiak, G. Grimnes, and S. Handschuh. Learning from Linked Open Data Usage: Patterns & Metrics. In *WebSci10:Extending the Frontiers of Society On-Line*, 2010.
3. M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. DBpedia SPARQL Benchmark–Performance Assessment with Real Queries on Real Data. In *ISWC Conference*, 2011.
4. S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards Robustness in Query Auditing. In *VLDB Conference*, 2006.
5. G. Nassopoulos, P. Serrano-Alvarado, P. Molli, and E. Desmontils. FETA: Federated QuEry TrAcking for Linked Data. In *DEXA Conference*, 2016.
6. F. Picalausa and S. Vansummeren. What are Real SPARQL Queries Like? In *SWIM Workshop*, 2011.
7. R. Verborgh, E. Mannens, and R. Van de Walle. Initial Usage Analysis of DBpedia's Triple Pattern Fragments. In *USEWOD Workshop*, 2015.
8. R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. *Journal of Web Semantics*, 37–38, Mar. 2016.