Querying Large Knowledge Graphs over Triple Pattern Fragments: An Empirical Study

Lars Heling(✉), Maribel Acosta, Maria Maleshkova, and York Sure-Vetter

Institute AIFB, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{lars.heling,maribel.acosta,maria.maleshkova,york.sure-vetter}@kit.edu

Abstract. Triple Pattern Fragments (TPFs) are a novel interface for accessing data in knowledge graphs on the web. So far, work on performance evaluation and optimization has focused mainly on SPARQL query execution over TPF servers. However, in order to devise querying techniques that efficiently access large knowledge graphs via TPFs, we need to identify and understand the variables that influence the performance of TPF servers on a fine-grained level. In this work, we assess the performance of TPFs by measuring the response time for different requests and analyze how the requests' properties, as well as the TPF server configuration, may impact the performance. For this purpose, we developed the *Triple Pattern Fragment Profiler* to determine the performance of TPF server. The resource is openly available at https://doi.org/10.5281/zenodo.1211621. To this end, we conduct an empirical study over four large knowledge graphs in different server environments and configurations. As part of our analysis, we provide an extensive evaluation of the results and focus on the impact of the variables: triple pattern type, answer cardinality, page size, backend and the environment type on the response time. The results suggest that all variables impact on the measured response time and allow for deriving suggestions for TPF server configurations and query optimization.

1 Introduction

Accompanied by the proliferation of knowledge graphs on the web as Linked Data [11], storage, and management solutions are constantly being newly developed or improved in order to support the necessity for accessing knowledge graphs online. A variety of interfaces to access and query RDF knowledge graphs have been proposed, including SPARQL endpoints and Triple Pattern Fragments (TPFs) [14]. Conceptually, the main difference among these interfaces is the expressivity of the requests they are able to handle: endpoints support the execution of SPARQL queries while TPFs are able to evaluate triple patterns. Furthermore, TPFs allow for querying RDF knowledge graphs which may be stored in different sources or backends. The evaluation of a triple pattern

D. Vrandečić et al. (Eds.): ISWC 2018, LNCS 11137, pp. 86–102, 2018.
https://doi.org/10.1007/978-3-030-00668-6_6

against a TPF server produces a sequence of RDF triples that match the given triple pattern; this is called a fragment. In addition, fragments may be partitioned into several fragment pages which contain a fixed maximum number of triples defined as the page size. The page size is configured by the data provider. To retrieve an entire fragment, clients must iterate (or paginate) over the TPF pages.

In order to devise efficient querying techniques over knowledge graphs on the web, it is necessary to understand the factors that impact the performance of the different interfaces, in particular, their response time. The factors that impact the response time of SPARQL endpoints have been extensively studied [2,6,8,10]. Therefore, in this work, we tackle the problem of identifying the variables that impact on the response time of TPFs when querying large RDF knowledge graphs. Our work aims to contribute to a better understanding of the costs related to retrieving data by querying RDF knowledge graphs over TPFs. An array of variables impacting the performance of TPFs have been the subject of previous studies [4,5,13,14], however these evaluations mostly focus on a higher level of query evaluation. Therefore, we provide a fine-grained study on the performance of TPFs and analyze further variables potentially impacting the performance. Concretely, we focus on the following research questions:

RQ1 How does the type of triple pattern impact the response time?
RQ2 What are the effects of the answer cardinality of triple patterns on response time?
RQ3 What is the impact of using different page sizes on performance?
RQ4 How does the response time differ when comparing different backends?
RQ5 What are the effects on response time when TPFs serve as an interface to several knowledge graphs simultaneously?
RQ6 What differences in performance can be observed between querying TPFs in controlled and real-world environments?

We investigate these research questions by first devising a profiler that generates triple patterns from sampling knowledge graphs over TPFs. Our profiler then executes the generated triple patterns and records the performance of the TPF servers. The outcome of the profiler allows for a fine-grained analysis of the factors that may impact on TPF performance. Subsequently, we conduct an empirical study which evaluates the costs of querying four well-known knowledge graphs over TPFs. In summary, we make the following contributions:

- A methodology for generating triple patterns from sampling knowledge graphs over TPFs;
- A fine-grained and extensive evaluation to analyze the impact of several independent variables on the performance of TPFs;
- Finally, we support the reproducibility of our results by providing the raw data as well as the *Triple Pattern Fragment Profiler*[1].

[1] https://github.com/Lars-H/tpf_profiler.

The herein presented evaluation setup was specially designed in order to ensure reuse and open access for the community, as well as reproducibility of the experimental results. The used settings can be replicated, in order to enable the verification of our analysis but also to provide the basis for further experiments and further work by other researchers in the field. The results of our study potentially allow for deriving a relationship between the properties of a requested triple pattern and the corresponding response time. In a subsequent step, the information derived from our analysis may be applied for constructing an empirical cost estimation model to be used in (federated) query engines and help to improve the configuration when setting up TPF servers.

The remainder of this paper is structured as follows. Section 2 presents related work and in Sect. 3 we present our methodology including the sample generation and the TPF profiler. The setup of our study is detailed in Sect. 4. The results of our empirical study are presented and discussed in Sect. 5. Finally, in Sect. 6 we provide our conclusions and an outlook to future work.

2 Related Work

Montoya et al. [8] identified the independent and dependent (or observed) variables that impact on the performance of querying federations of SPARQL endpoints. Following a similar classification, the independent variables that may impact on the performance of TPFs can be grouped into four dimensions: Query, Knowledge Graph (KG), Triple Pattern Fragment Configuration, and Platform. Table 1 summarizes the independent variables studied in the literature and in our work. In this work, we focus on analyzing the dependent variable response time, i. e., the elapsed time between the client contacting the server and the first response arriving. In the following, we position our work with respect to experimental studies that have analyzed the impact of independent variables on the response time (or cost) of querying KGs via SPARQL endpoints or TPF servers.

Query. This dimension includes variables with regard to the structure of the query. Most of the works have studied the impact of the query shape with different types of joins (e. g., subject-subject, subject-object, etc.) on the performance of endpoints [2,8] and TPFs [1,13,14], as well as the effect of specifying SPARQL query operators with different complexity [2,6,8] on the response time of endpoints. Nonetheless, little attention has been paid to studying the impact of individual triple patterns and their instantiations [2,14] or answer size [1] on server performance. In this work, we conduct a fine-grained study of independent variables in the query dimension at the level of triple patterns to better understand the behavior of TPFs when evaluating different types of requests.

Knowledge Graph (KG). The variables in this dimension characterize the KG or the data, including the number of statements in the KG, the distribution of nodes and relationships, and partitioning or replication of the data. Analogous to other studies [2,6,8], in our work we study the response time of servers when querying real-world KGs with different sizes and data distributions.

Table 1. Comparison of empirical studies that analyze the impact of independent variables on the response time of SPARQL endpoints or TPFs.

Independent variables	Endpoints				Triple Pattern Fragments					
	[8]	[6]	[2]	[10]	[13]	[1]	[14]	[4]	[5]	Our Work
Query										
Query shape	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗
Query complexity	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
Triple pattern type	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
Number of constants	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓
Answer cardinality	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓
Knowledge Graph (KG)										
KG size	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓
Data frequency distribution	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓
KG partitioning/replication	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Triple Pattern Fragment Configuration										
Backend type	–	–	–	–	✗	✗	✗	✗	✗	✓
Page size	–	–	–	–	✗	✗	✗	✗	✗	✓
Pagination	–	–	–	–	✗	✗	✗	✗	✗	✓
Relation KGs/TPF instance	–	–	–	–	✗	✗	✗	✗	✗	✓
Platform										
Server workload	✗	✗	✓	✗	✓	✗	✓	✓	✓	✓
Network delays	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓
Caching	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗
Data serialization	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Hardware configuration	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗

Triple Pattern Fragment Configuration. This dimension focuses on the variables that are particular to the TPFs. Based on the current definition and implementations of TPFs [13,14] we identify four variables in this dimension: backend type, page size, pagination, relation KGs/TPF instance. Current implementations of TPF servers support different backends including HDT files [3], SPARQL endpoints, and RDF documents.[2] Another important feature of TPFs is that they partition the result of evaluating a triple pattern into fragment pages; each page contains a fixed maximum number of triples. In this work, we will focus on studying the behavior of TPF servers with configurations, suitable for querying large knowledge graph. Another variable in this dimension is pagination, i. e., the cost of iterating the fragment pages to completely evaluate a triple pattern. Lastly, the KGs/TPF variable captures the correspondence of the KGs accessed via an instance of a TPF server. From all the independent

[2] https://github.com/LinkedDataFragments/Server.js.

variables listed in the TPF configuration dimension, only the work by Hartig and Buil-Aranda [5] investigated the impact of varying the page size on cache hits but not on response time. In contrast, our work studies the variables pagination and the relation KGs/TPF by studying the cost of dereferencing different pages of a given fragment and quantifying the cost of querying TPFs when KGs are accessed individually or simultaneously through a TPF instance, respectively.

Platform. The platform dimension as defined by Montoya et al. [8] comprises the variables that describe the computing infrastructure. In this dimension, the most studied variables are: server workload [2,4,5,13,14], network delays [1,2,8], and caching [2,4,5]. In our work, we study the impact of server workload and network delays when performing requests in two different environments: a controlled environment with one client and no network delays (best-case scenario), and a real-world environment when querying public TPF servers.

3 Our Approach

We devise an approach to measure the response time of Triple Pattern Fragment (TPF) servers under different conditions. Our approach is independent of the underlying knowledge graph (KG) and TPF server configuration and can be applied to conduct empirical studies on any KG accessible via TPF servers. The main components of our approach are the **Triple Pattern Sampling Component** and the **Evaluation Component**. In the following, we describe the sampling component and then present how the samples are used to capture the performance of TPFs.

The goal of the proposed Triple Pattern Sampling Component is generating random triple patterns with different characteristics to evaluate the performance of TPF servers under different conditions. The input of the sampling method is a KG G composed of a set of RDF triples accessible via a TPF server, and a sample size m. The output of this method is a sample S, with S corresponding to a set of triple patterns such that the evaluation of a triple pattern $tp \in S$ over G produces answers, i.e., $[[tp]]_G \neq \emptyset$. The core idea of our sampling method is to select a set of RDF triples in a given KG and derive triple patterns by replacing constants (RDF terms) with existential variables. Assuming R the set of RDF terms – IRIs, literals, and blank nodes – and V the universe of variables [9], the triple patterns in the generated sample are of the following 2^3 types: $\{\langle r1, r2, r3\rangle$, $\langle v1, v2, v3\rangle$, $\langle v1, v2, r3\rangle$, $\langle v1, r2, v3\rangle$, $\langle r1, v2, v3\rangle$, $\langle r1, v2, r3\rangle$, $\langle r1, r2, v3\rangle$, $\langle v1, r2, r3\rangle\}$ with $r1, r2, r3 \in R$ and $v1, v2, v3 \in V$.

For a given KG G accessible via a TPF server, the sampling component produces a set of RDF triples $G^* \subseteq G$, such that $|G^*| = m$. In order to build G^*, the proposed component performs random sampling over the KG such that G^* contains RDF triples that capture different characteristics of G in terms of data distributions. Furthermore, the sampling component exploits the features of TPFs to construct G^* from G. For instance, when evaluating a triple pattern

tp against G over a TPF server, the server provides the set[3] G_{tp} of RDF triples that match tp as well as metadata that includes an approximation to the total number of answers of tp. Furthermore, TPFs partition G_{tp} into subsets called pages [14]. Let $\mathscr{P}$ be the set of pages of G_{tp}, each page $P_i \in \mathscr{P}$ contains a fixed maximum number of triples, i.e., $|P_i| \leq p_{max}$. To retrieve the complete answer G_{tp} of triple pattern tp, it is necessary to iterate over all the pages in $\mathscr{P}$, where $|\mathscr{P}| = \left\lceil \frac{|G_{tp}|}{p_{max}} \right\rceil$. To generate G^*, the sampling method proceeds as follows:

1. The sampling component evaluates the triple pattern tp with variables in subject, predicate and object position (i. e. $tp = \langle v1, v2, v3 \rangle$) against the KG G via a TPF server. The result of this evaluation is a set of pages $\mathscr{P}$ with RDF triples G_{tp} that match tp.
2. From the set of pages $\mathscr{P}$, the sampling method randomly selects m pages following a random uniform distribution, i.e., all the pages in $\mathscr{P}$ have the same probability of being selected.
3. For each page $P_i \in \mathscr{P}$, the sampling method randomly selects an RDF triple $\langle s, p, o \rangle \in P_i$ following a random uniform distribution and adds it to G^*.
4. From the RDF triples in G^* the triple pattern sample S is generated by replacing the RDF terms with variables: $S = \bigcup_{\langle s,p,o \rangle \in G^*} \{v1, s\} \times \{v2, p\} \times \{v3, o\}$, with $v1, v2, v3 \in V$.

It is important to note that during the sampling of RDF triples, two different RDF triples may lead to the generation of the same triple pattern. For instance, consider the RDF triples $\langle s1, p, o1 \rangle$ and $\langle s2, p, o2 \rangle$ with $s1, s2, p, o1, o2 \in R$, $s1 \neq s2$, and $o1 \neq o2$. In this case, both triples produce the common triple pattern $\langle v1, p, v3 \rangle$ with $v1, v3 \in V$. Nonetheless, according to step 4 of the sampling method, $\langle v1, p, v3 \rangle$ occurs once in S. The reason for restricting S to unique triple patterns is to avoid unwanted caching effects when measuring the performance of TPF servers, which may happen when requesting the same triple pattern several times sequentially. As every triple pattern tp with $|[[tp]]_G| \in [1, |G|]$, $\forall tp \in S$ is unique in the sample, it follows that $m \leq |S| \leq 2^3 m$. This means that the input parameter m of the triple pattern sampling component corresponds to a lower bound on the size of S. Furthermore, we also want to examine the response time for triple patterns which produce no results for the KG (i. e., $[[tp]]_G = \emptyset$). Thus, we extend the sampling component to add m randomly generated patterns to the sample S with $[[tp]]_G = \emptyset$. Patterns that produce empty result sets are generated by randomly selecting triple patterns from the sample and randomly replacing constants with URIs not contained in the KG.

The aforementioned sampling method is implemented in our Triple Pattern Fragment Profiler (see Footnote 1) to conduct the empirical studies. The integration of the sampling methodology in the TPF Profiler to examine the performance of TPF servers is shown in Fig. 1. The required input is the sample size m as well as the URL of the TPF server to access the KG G. Based on this

[3] Although the formal definition specifies that G_{tp} is a sequence of RDF triples [14], for the sake of simplicity, we define G_{tp} as a set of RDF triples.

input, the triple pattern sample set S is generated. Thereafter, in the Evaluation Component, each triple pattern in the sample is requested sequentially at the TPF server and the response time for each request is recorded. A detailed presentation of the implementation and the setup of the experimental studies is given in the following section.

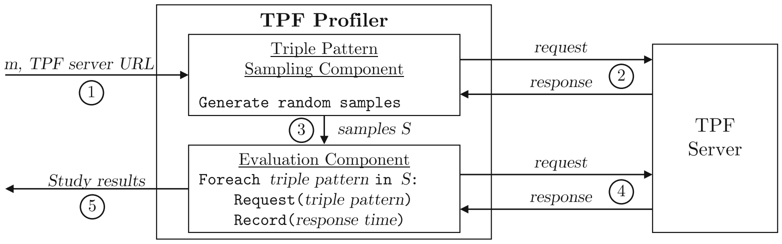


Fig. 1. Overview of the Triple Pattern Fragment Profiler. The numbers indicate the execution sequence.

4 Experimental Settings

We provide a detailed description of the settings used to assess the performance of TPF server in different conditions using the presented approach. This includes environment and implementation, backend and page size, the selection of the knowledge graphs, sample size determination, and reported metrics. To ensure repeatability of our study, the TPF Profiler, as well as the HDT files containing the KGs, are available at https://doi.org/10.5281/zenodo.1211621 under the BSD 3-Clause license. In addition, we provide the raw data and the analysis tools to ensure reproducibility of our experimental results (see Footnote 9) under CC BY 4.0.

Environment and Implementation. We conducted our study in two types of environments: a real-world environment by accessing autonomous TPF servers, and a controlled environment using a dedicated server. As a result, we compare the querying costs in two environments which differ in networking conditions, server workload, and possibly hardware capabilities. In the real-world environment, we accessed the TPF servers available at the official portal of Linked Data Fragments[4]. For the controlled environment, we deployed the TPF server v2.2.3 [12] on a Debian GNU/Linux 8.6 64-bit machine with CPU AMD Opteron 6204 3.3 GHz (4 physical cores) and 32 GB RAM. The TPF profiler is implemented in `Python 2.7.9` and executed on the same server instance to avoid network latencies accessing the TPF server in the controlled environment.

Backend and Page Size. We consider different backends as well as page sizes for each backend to provide a detailed insight into how the configuration of the

[4] http://linkeddatafragments.org.

TPF server impact on its performance. To our knowledge, the backend used in the publicly available servers is HDT files and thus is the only backend type in the real-world environment. In the controlled environment, both HDT files as well as SPARQL endpoints are used as backends since they are both suitable for querying large KGs. The SPARQL endpoints are set up using Virtuoso Open Source Edition Version 7.2.4[5]. Since the settings of the real-world server may not be changed, the page size p_{max} is set to 100 answers per page. In the controlled environment, four different page size settings are investigated: $p_{max} = \{100, 500, 1000, 10000\}$. The configuration files for both the TPF server and the Virtuoso SPARQL endpoint are provided in our repository (see Footnote 1).

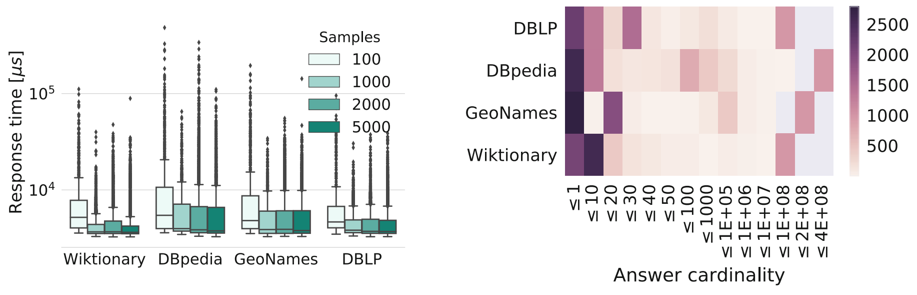
Table 2. Characterization of the knowledge graphs studied in the evaluation. The namespace `ldf` corresponds to http://data.linkeddatafragments.org/.

Knowledge graph	# Triples	# Subjects	# Predicates	# Objects	Server IRI
DBLP	88,150,324	5,125,936	27	36,413,780	`ldf:dblp`
DBpedia	377,367,913	30,458,591	57,465	145,396,686	`ldf:dbpedia2014`
GeoNames	123,020,821	8,345,450	26	42,728,317	`ldf:geonames`
Wiktionary	64,358,375	10,163,240	27	21,554,657	`ldf:wiktionary`

Knowledge Graphs (KGs). Having both a controlled and a real-world environment requires KGs which are both publicly accessible via TPF servers and are available for download to be hosted locally in the controlled environment. Therefore, we selected four well-known KGs available at the official portal of Linked Data Fragments (see Footnote 4) from different knowledge domains: publications (DBLP), geography (GeoNames), linguistics (Wiktionary), and cross-domain (DBpedia). The selected KGs differ in their size (i. e., the number of RDF triples) as well as the number of distinct subjects, predicates and objects (cf. Table 2). As the basis for the controlled environment, we use the identical HDT files as in the real-world environment to ensure comparability. The N-Triples files for Virtuoso are generated from these HDT files using the `hdt2rdf` tool[6] and the characterization in Table 2 is derived from the HDT files using the `hdtInfo` tool (see Footnote 6).

Generated Samples. For our study, we generated samples of triple patterns to be executed over the selected KGs following the sampling method described in Sect. 3. A key aspect of the evaluation settings for sampling is determining an appropriate sample size, i.e., the parameter m. In the following, we describe how we determined m empirically such that m fulfills the conditions: (i) m is large enough to cover a variety of data to represent the general characteristics of the KG, and (ii) m is small enough to efficiently assess the performance of the servers in a feasible fashion. A basic approach to set m is a sample size relative to the size

[5] https://virtuoso.openlinksw.com.
[6] https://github.com/rdfhdt/hdt-cpp.



(a) Sample sizes (b) Sample cardinalities

Fig. 2. Sample properties. On the left, the overall response times with respect to the sample size m are shown. On the right, the distribution of the answer cardinalities for a sample of size $m = 1000$ is shown.

of the KG (e. g., 1% of all triples). However, the major drawback of this approach is that for large datasets, a large number of requests needs to be performed on the TPF to acquire the sample. For instance, a sample size with 1% of DBpedia with ∼400M RDF triples would require approximately 4M requests. Firstly, this is not feasible with respect to the overall run time of the study and, secondly, a larger sample size does not necessarily entail a more representative sample.[7] To verify this, we measured the performance of TPFs in a controlled environment (HDT backend) when varying the sample size $m = \{100, 1000, 2000, 5000\}$ on the studied KGs. The results reported in Fig. 2a reveal that there is no substantial difference in the response times while increasing m (for $m \geq 1000$) in all KGs. Furthermore, we inspected the answer cardinality distribution produced by the triple patterns in the sample obtained with 1000 RDF triples. Figure 2b indicates that the sample generated for each studied KG contains triple patterns with a wide range of answer cardinalities. Therefore, in this study, we set $m = 1000$.

Metrics. The metric to assess the cost of querying TPF server is the response time. In this work, response time t is defined as the elapsed between sending a request and the arrival of its response. Therefore, the time for retrieving the first fragment page of a requested triple pattern is measured. To measure t in our implementation, we use the Python library `requests`[8] v2.18.4. in which the elapsed time between the request and the response merely considers the time until the parsing of the headers is completed and therefore, is not affected by the size of the response's content. We report the measurements of response time in microseconds (μs).

We conducted an extensive experimental study to answer the research questions stated in Sect. 1. At the core of the study is the identification of the independent variables and their effect on TPF performance. Table 3 summarizes the five

[7] Since the likelihood of samples having RDF terms in common increases for larger samples resulting in the same triple pattern derived from the sample.
[8] http://docs.python-requests.org/en/master.

Table 3. Overview of the independent variables analyzed per knowledge graph in the evaluation of the experimental study. R and V correspond to RDF terms and variables, respectively. G denotes a knowledge graph and tp a triple pattern.

Independent variable	# Levels	Levels
Environment	2	{controlled, real-world}
Backend	2	{HDT, Virtuoso}
Page size	4	$p_{max} = \{100, 500, 1000, 10000\}$
Triple pattern type	2^3	$(R \cup V) \times (R \cup V) \times (R \cup V)$
Knowledge graph	4	{DBLP, DBpedia, Geonames, Wiktionary}
Answer cardinality	$\|G\|$	$\|[[tp]]_G\| \in [0, \|G\|]$

variables analyzed and their levels with respect to their impact on the response time in the experimental evaluation. The results of our experiments investigating the different levels of these variables yield ≈590000 measurements.

5 Empirical Results

In this section, we present and analyze the results of our experimental studies. Essential to the evaluation of the experimental studies are the methods for analyzing the results. The goal of the analysis is determining the impact of the independent variables on the dependent variable, i. e., the response time. For this purpose, we propose the use of several statistical methods. The method selection primarily depends on the type of the independent variable and dependent variable. For the analysis in our study, the dependent variable response time is continuous. The independent variable, however, is discrete for the answer cardinality and page number and categorical for all others. Therefore, we apply a correlation analysis for the answer cardinality and paginating. For all other independent variables, we report on the significance of the difference between the categories. In our case, the observed variable response time is not normally distributed[9] and thus, we use the non-parametric Kruskal-Wallis test [7] to test our hypothesis. For the sake of brevity, we provide a rather graphical evaluation in this paper. Nevertheless, the complete statistical analysis as well all visualizations of the following evaluation are provided online (see Footnote 9).

Triple Pattern Type. First, we address the research question, whether the triple pattern type has an impact on the response time. In this evaluation, we merely consider the page size $p_{max} = 100$ to allow for the comparability with the real-world environment. In Fig. 3 the results are visualized as a boxplot separated by triple pattern type including `empty` for triple patterns with an empty answer set. The results are listed for the different KGs in both the controlled (with HDT and Virtuoso backend) and real-world environment. Additionally, the mean

[9] https://doi.org/10.5281/zenodo.1211621.

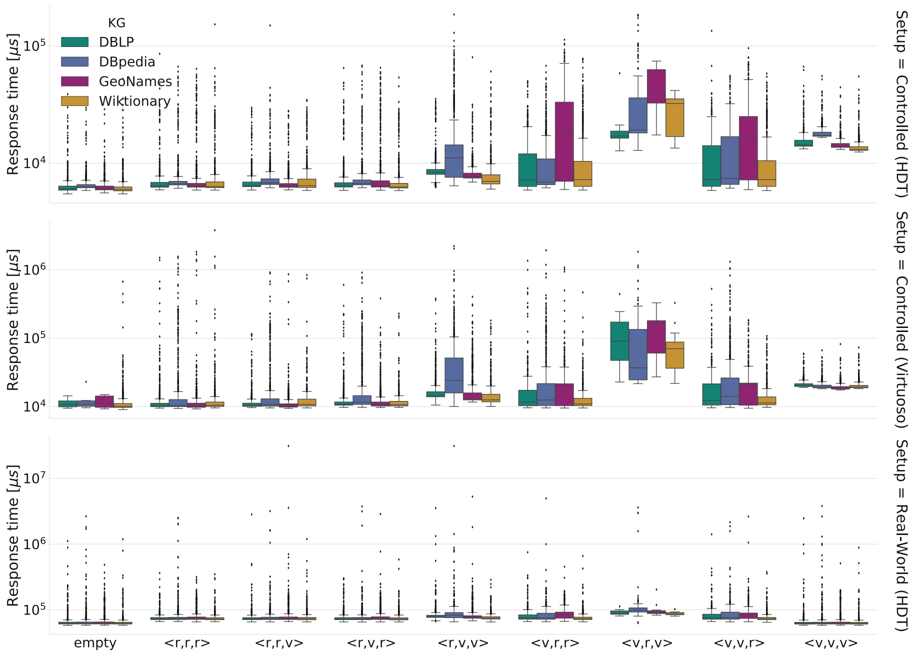


Fig. 3. Boxplot of the response time for the different pattern types.

number of answers per pattern type is listed in Table 4 for the different KGs. The results reveal a difference in response time for the different pattern types. Conducting a Kruskal-Wallis test, we find that the difference between the groups (i. e. pattern types) is statistically significant at a level $\alpha = 0.05$ in both the controlled and real-world environment. In more detail, the response times for some pattern types differ more prominently from the other pattern types. For instance, the triple pattern type $\langle v, r, v\rangle$ shows the highest (median) response times for all KGs in all environments. Note that $\langle v, r, v\rangle$ denotes the triple pattern composed of variable, constant, variable (in that order), but the variables are not necessarily the same. Furthermore, the $\langle v, v, v\rangle$ pattern type yields the second highest response time in the controlled environment (for both HDT and Virtuoso backend), except for DBpedia in which case the $\langle r, v, v\rangle$ yields a higher response time for the Virtuoso backend. Intriguingly, this is not true in the real-world environment, in which the $\langle v, v, v\rangle$ pattern type has one of the lowest response times. This might be due to the fact, that the results for the pattern are cached in the real-world environment as it is requested more frequently. Comparing the backends in the controlled environment, it can be observed that the pattern types have a similar impact on the response. However, the average response time with the Virtuoso backend (21.7 ms) is more than twice as high than for the HDT backend (10.2 ms).

Moreover, compared to the other KGs, the response times for GeoNames are notably higher for the pattern types $\langle v, r, r\rangle$ and $\langle v, v, r\rangle$. Taking the mean number of answers for these pattern types into consideration, we observe that these pattern types also yield the most answers on average. The previous observation may lead to the assumption that merely the higher answer cardinalities are the reason for higher response times.

Table 4. Mean answer cardinality of the triple patterns in the sample listed for the different triple pattern types and the different KGs.

	$\langle r, r, r\rangle$	$\langle r, r, v\rangle$	$\langle r, v, r\rangle$	$\langle r, v, v\rangle$	$\langle v, r, r\rangle$	$\langle v, r, v\rangle$	$\langle v, v, r\rangle$	$\langle v, v, v\rangle$
DBLP	1.00E + 00	2.37E + 00	1.19E + 01	2.23E + 01	3.92E + 03	4.18E + 06	3.94E + 03	8.82E + 07
DBpedia	1.00E + 00	4.85E + 00	2.82E + 01	5.55E + 01	2.09E + 03	9.58E + 05	3.50E + 03	3.77E + 08
GeoNames	1.00E + 00	1.06E + 00	8.11E + 00	1.50E + 01	3.36E + 04	5.13E + 06	3.58E + 04	1.23E + 08
Wiktionary	1.00E + 00	5.83E + 00	6.95E + 00	1.32E + 01	4.57E + 04	3.50E + 06	1.18E + 05	6.44E + 07

Answer Cardinality. The response times with respect to the answer cardinality are shown in Fig. 4. The results reveal a similar trend for both the controlled and the real-world environment. There is an increase in the response time up to ≈100 answers and thereafter the response times appear to be rather steady. As the page size in this visualization is $p_{max} = 100$ answers per page, the results suggest that the difference may be related to the page size. To quantify this relation, we report on the correlation coefficient ρ, which allows for measuring the strength and direction of the linear correlation between two variables. Table 5 lists the correlation coefficients for answer cardinality and response time for samples when: (i) the answers fit in one fragment page ($\rho_{\leq p_{max}}$), (ii) pagination is required to dereference the fragment ($\rho_{>p_{max}}$), and (iii) the overall correlation coefficient (ρ). The correlation is reported for the different backends and page sizes in the controlled environment as well as the correlation for the HDT backend and page size $p_{max} = 100$ in the real-world environment.

In Table 5, a weak positive correlation ($\rho \in [0.5, 0.75]$) is highlighted with a light color and a stronger positive correlation ($\rho \in [0.75, 1]$) with a dark color. The results for Virtuoso reveal more often and stronger positive correlations between the answer cardinality and the response time. This indicates that HDT is more efficient in querying the patterns regardless of the answer cardinality. This is also visible in Fig. 4, where there are more outliers with respects to the overall trend for HDT backend in the controlled environment. In contrast, the Virtuoso backend appears to have an increasing lower bound of the response time for higher answer cardinalities. Moreover, it can be observed that the page size also influences the correlation in the controlled environment since the correlation is only present for page size $p_{max} > 100$. In the real-world environment, no correlation can be observed as most correlation indices are close to zero. This suggests that exogenous factors, e.g. network delays and server load, affect the response time, such that the differences induced by the answer cardinality vanish.

Page Size. We measure the performance of TPFs for different page size settings. We report on throughput, i.e., the number of answers produced per time unit.

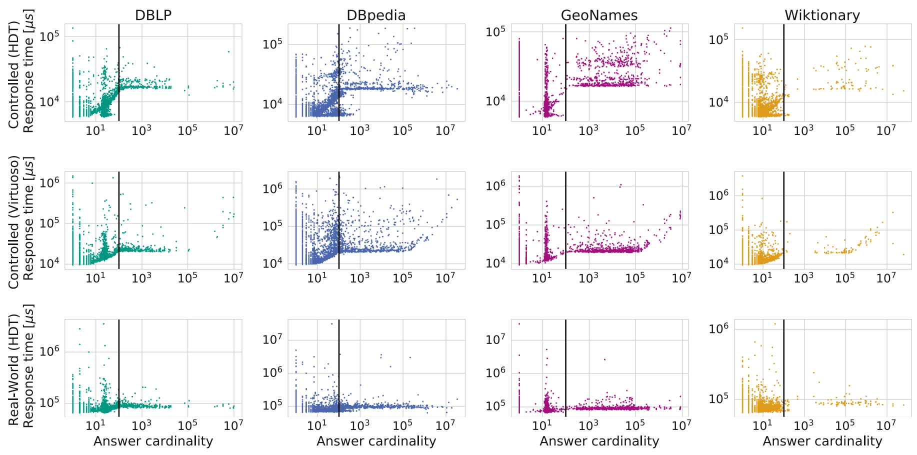


Fig. 4. Response time in the controlled environment and real-world environment with respect to the number of answers for all samples and each KG for all samples except the $\langle v, v, v \rangle$ pattern type. The black line indicates 100 results (page size).

Table 5. Correlation of answer cardinality and response time in the controlled and real-world environment for varying page size and backend. Correlation coefficient for samples whose answers fit in one fragment page ($\rho_{\leq p_{max}}$), when pagination is required to dereference the fragment ($\rho_{> p_{max}}$), and the overall correlation (ρ).

Environment		Controlled								Real-World
Page Size p_{max}		100		500		1000		10000		100
Backend		HDT	Virt.	HDT	Virt.	HDT	Virt.	HDT	Virt.	HDT
DBLP	$\rho_{\leq p_{max}}$	0.266	0.078	0.141	0.842	0.858	0.913	0.979	0.734	0.039
	$\rho_{> p_{max}}$	0.100	0.441	0.262	0.885	0.229	0.896	0.049	0.727	-0.059
	ρ	0.100	0.166	0.051	0.522	0.269	0.420	0.475	0.451	0.004
DBpedia	$\rho_{\leq p_{max}}$	0.297	0.143	0.301	0.709	0.301	0.918	0.916	0.679	0.020
	$\rho_{> p_{max}}$	-0.014	0.185	0.039	0.735	0.039	0.576	-0.000	0.171	0.000
	ρ	0.036	0.129	0.101	0.525	0.101	0.381	0.130	0.219	0.002
Geo-Names	$\rho_{\leq p_{max}}$	-0.032	0.041	0.130	0.777	0.878	0.963	0.983	0.997	-0.006
	$\rho_{> p_{max}}$	0.169	0.292	0.158	0.890	0.318	0.701	0.354	0.328	0.004
	ρ	0.179	0.127	0.136	0.465	0.238	0.295	0.276	0.253	0.001
Wikt-ionary	$\rho_{\leq p_{max}}$	0.047	0.013	0.207	0.388	0.239	0.302	0.667	0.986	0.071
	$\rho_{> p_{max}}$	-0.027	0.207	-0.019	0.204	-0.063	0.211	-0.263	0.025	-0.077
	ρ	0.059	0.029	0.061	0.281	0.124	0.267	0.193	0.237	0.009

We compute the throughput per fragment page (denoted Θ) when evaluating each triple pattern $tp \in S$ over a KG G as follows:

$$\Theta(tp) := \frac{\min\{|[[tp]]_G|, p_{max}\}}{t(tp)} \; [answers/\mu s], \tag{1}$$

where $|[[tp]]_G|$ is the answer cardinality, p_{max} the page size and $t(tp)$ the response time. Figure 5 shows the mean Θ for all KGs, different backends and page size in the controlled environment. The relative changes in throughput are listed in Table 6. For the HDT backend, the results reveal an increase in throughput for bigger page sizes in most cases. The biggest increase is achieved by setting the TPF page size from 100 to 500 answers per page, in all KGs. When increasing the page size further from 1000 to 10000, the throughput even slightly decreases. Similar results are observable for the Virtuoso backend. In contrast to HDT, however, the throughput is not affected as strongly: it merely improves half as much when increasing the page size from 100 to 500 answers per page. Thus, the improvement is less significant than it is for the HDT backend.

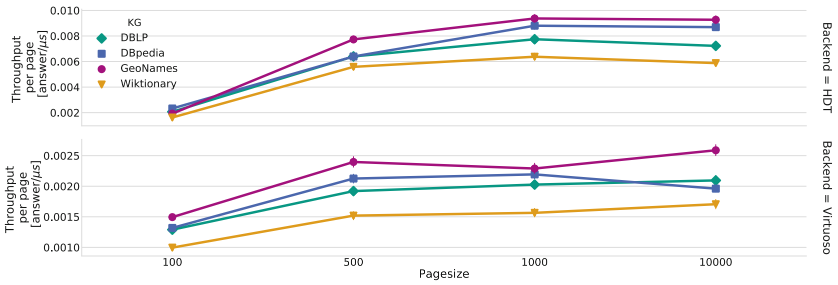


Fig. 5. Mean TPF throughput (Θ) for all KGs, different backends and page sizes.

Table 6. Relative changes in throughput Θ for increasing the TPF page size.

	DBLP		DBpedia		GeoNames		Wiktionary	
	HDT	Virtuoso	HDT	Virtuoso	HDT	Virtuoso	HDT	Virtuoso
From 100 to 500	0.677	0.329	0.636	0.380	0.751	0.376	0.710	0.345
From 500 to 1000	0.174	0.052	0.274	0.031	0.175	−0.047	0.124	0.029
From 1000 to 10000	−0.072	0.033	−0.013	−0.119	−0.011	0.115	−0.085	0.083

Paginating. As the preceding evaluation shows, the page size has an impact on the throughput of TPFs. Moreover, the magnitude of the impact varies for the backend types. The throughput merely considers retrieving at most the first p_{max} answers. However, for $|[[tp]]_G| > p_{max}$ paginating is required to obtain all answers. To examine how the response time varies when paginating, we dereference the first 10000 pages for the $\langle v, v, v\rangle$ pattern. The results are presented in Fig. 6. It can be observed that for the HDT backend, paginating yields a rather constant response time, while a steady increase can be observed for the Virtuoso backend. These observations are supported by the correlation coefficient ρ, which clearly indicates a strong positive correlation between page number and response time for the Virtuoso backend with $\rho = 0.920$ and no correlation for the HDT backend with $\rho = -0.036$. The two previous evaluations suggest that the page size configuration for TPF server needs to consider the backend. For instance, for the Virtuoso backend increasing the page size might be suitable as it increases throughput and paginating is increasingly costly. As the response time for paginating is rather constant for the HDT backend, bigger page sizes may allow for exploiting the increased throughput and further reduce the necessity of paginating. However, the adjustments need to take the answer cardinality distribution of the KG into consideration as well.

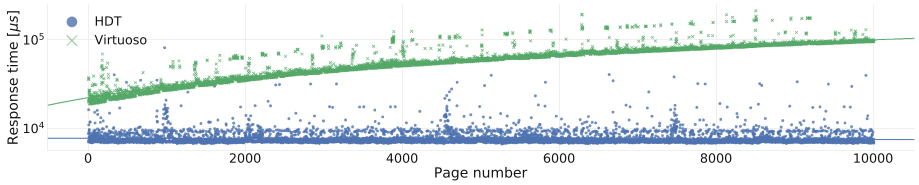


Fig. 6. TPF response time for paginating the first 10000 pages for the triple pattern $\langle v, v, v \rangle$ for both HDT and Virtuoso backend in the controlled environment.

KG/TPF Instance Relation. Finally, we examine the results when making all KGs simultaneously available in the TPF server for each backend. The mean response time for all KGs (Multiple KG) and one KG (Single KG) available at a time are presented in Fig. 7. The results show an increase in the TPF response time when making multiple KGs simultaneously available regardless of the backend. Intriguingly, the increase is higher for the smaller KGs and lower for larger KGs for the HDT backend and the opposite holds for the Virtuoso backend. These observations may be due to the index created by the TPF server when making several KGs available simultaneously.

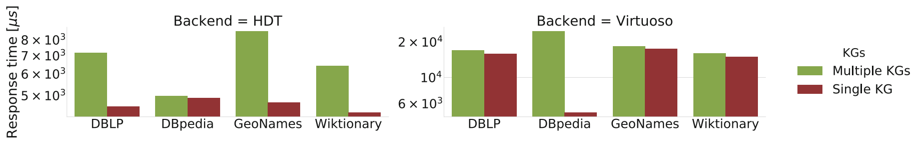


Fig. 7. Relation KGs/TPF instance. Mean response time for both backends with all KGs available and single KGs available at a time.

6 Conclusions and Future Work

We have proposed an approach to assess the cost of querying knowledge graphs (KGs) over Triple Pattern Fragments (TPFs). The presented TPF Profiler includes a sampling component able to generate triple patterns from KGs and an evaluation component to capture the response time for the sampled triple patterns. The results allow for conducting fine-grained analyses identifying factors that impact on server performance. We conducted an empirical study using the proposed approach to evaluate the TPF servers in controlled and real-world environments using diverse well-known KGs and different TPF configurations. Thereafter, we conducted a fine-grained analysis studying the impact of triple pattern type, answer cardinality, backend, page size, environment type, and KGs per TPF server on the response time. To conclude our findings, we answer the research questions stated in Sect. 1.

Answer to RQ1. Our empirical study confirms that the type of triple pattern has a significant impact on the response time of TPF servers regardless of the backend.

Answer to RQ2. Empirical results reveal different behaviors of TPFs depending on the answer cardinality of the triple patterns. For triple patterns that produce answers fitting in one page, answer cardinality is rather positively correlated with response time.

Answer to RQ3. The results of the experimental studies indicate an improved throughput for increasing page sizes. However, the throughput decreases for large page sizes again and the relative improvement is higher for the HDT backend.

Answer to RQ4. Our experimental study reveals significant differences in the response times between different backend types. Overall, the HDT backend outperforms the Virtuoso backend and allows for querying triple patterns more efficiently.

Answer to RQ5. Empirical results suggest that in real-world environments the impact of the analyzed variables on response time is reduced as exogenous factors increasingly affect the response times of TPF servers. In addition, querying autonomous real-world servers can be orders of magnitude more costly.

Answer to RQ6. Our experimental study reveals that accessing multiple KGs through a single TPF server negatively impacts on server performance.

At first sight, the absolute measured impact (in seconds) of some factors on the response time might not appear very high. However, the results in our study report on the response time for retrieving the first page of a fragment. Thus, the response times for paginating to retrieve all answers of a request needs to be considered as well. Moreover, the evaluation of SPARQL queries over TPFs typically requires submitting a large number of requests to the TPF server, thus, leveraging the observations may improve the overall query execution time.

Our future work will focus on integrating more variables into our analysis, studying different variable levels (e. g., different SPARQL endpoint implementations) and gathering additional data for KGs with different characteristics.

Acknowledgments. The authors thank Ruben Verborgh for providing feedback and the KG dumps and Javier Fernández for the fruitful discussions about HDT. This work was carried out with the support of the German Research Foundation (DFG) within the project "Sozial-Raumwissenschaftliche Forschungsdateninfrastruktur (SoRa)".

References

1. Acosta, M., Vidal, M.-E.: Networks of linked data eddies: an adaptive web query processing engine for RDF data. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 111–127. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_7
2. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: ready for action? In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 277–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_18
3. Fernández, J.D., Martínez-Prieto, M.A., Gutierrez, C.: Compact representation of large RDF data sets for publishing and exchange. In: Patel-Schneider, P.F., et al. (eds.) ISWC 2010. LNCS, vol. 6496, pp. 193–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17746-0_13
4. Folz, P., Skaf-Molli, H., Molli, P.: CyCLaDEs: a decentralized cache for triple pattern fragments. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 455–469. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34129-3_28
5. Hartig, O., Buil-Aranda, C.: Bindings-restricted triple pattern fragments. In: Debruyne, C. (ed.) OTM 2016. LNCS, vol. 10033, pp. 762–779. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48472-3_48
6. Kjernsmo, K., Tyssedal, J.S.: Introducing statistical design of experiments to SPARQL endpoint evaluation. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 360–375. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_23
7. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. J. Am. Stat. Assoc. **47**(260), 583–621 (1952)
8. Montoya, G., Vidal, M.-E., Corcho, O., Ruckhaus, E., Buil-Aranda, C.: Benchmarking federated SPARQL query engines: are existing testbeds enough? In: Cudré-Mauroux, P. (ed.) ISWC 2012. LNCS, vol. 7650, pp. 313–324. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35173-0_21
9. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_3
10. Rakhmawati, N.A., Karnstedt, M., Hausenblas, M., Decker, S.: On metrics for measuring fragmentation of federation over SPARQL endpoints. In: WEBIST, pp. 119–126 (2014)
11. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P. (ed.) ISWC 2014. LNCS, vol. 8796, pp. 245–260. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_16
12. Verborgh, R.: Linkeddatafragments/server.js: v2.2.2, May 2017. https://doi.org/10.5281/zenodo.570148
13. Verborgh, R., et al.: Querying datasets on the web with high availability. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 180–196. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_12
14. Verborgh, R., et al.: Triple pattern fragments: a low-cost knowledge graph interface for the web. Web Semant. Sci. Serv. Agents World Wide Web **37**, 184–206 (2016)