teepee - A Triple Pattern Fragment Profiler Visualization Tool

Lars Heling and Maribel Acosta

Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
{heling|acosta}@kit.edu

Abstract. Triple Pattern Fragments (TPFs) is a low-cost interface for querying knowledge graphs on the web. In this work, we present *teepee*, a visualization tool for analyzing the profiling results of TPF performance. Moreover, *teepee* allows for comparing the performance of different configurations of TPF servers. Attendees will observe how the execution of requests against TPFs is impacted by i) the type and cardinality of the requested triple patterns, and ii) network delays and server workload. The demo is available at `http://km.aifb.kit.edu/services/teepee/`.

1 Introduction

Triple Pattern Fragments [2] is a web querying interface that supports the execution of triple patterns against knowledge graphs (KGs) while guaranteeing high availability. In order to devise efficient querying techniques on top of TPFs, it is essential to empirically identify the variables that have an impact on their performance. In this work, we present *teepee*, a tool for visualizing and analyzing the performance of TPF servers in terms of response time.

teepee relies on the TPF Profiler [1] to assess the performance of TPF servers at a fine-grained level. The empirical results obtained by the TPF Profiler are aggregated and visualized with *teepee*. In this demonstration, the attendees will be able to observe with *teepee* the effects of different factors on the response time of TPF servers. We describe two demonstrations of use cases. The first use case shows the impact of triple patterns properties. The second use case compares the performance of querying local vs. remote TPF servers to analyze the effect of network delays and server workload. In addition, as part of the *teepee* demo, we have included further use cases that showcase the impact of factors from several dimensions using KGs with different characteristics and TPF servers with varying configurations, including page size, backend type, and network delays. In summary, the contributions of this work are:

- The online demo of *teepee*[1], which includes a basic setup of the TPF Profiler with a limited configuration for the sake of performance.
- The source code of *teepee*, which is available as part of the TPF Profiler repository[2] and can be adjusted and executed locally.

[1] `http://km.aifb.kit.edu/services/teepee/`

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

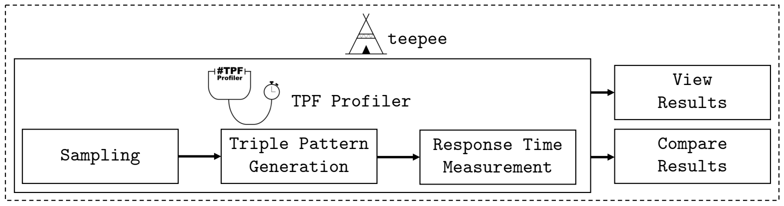


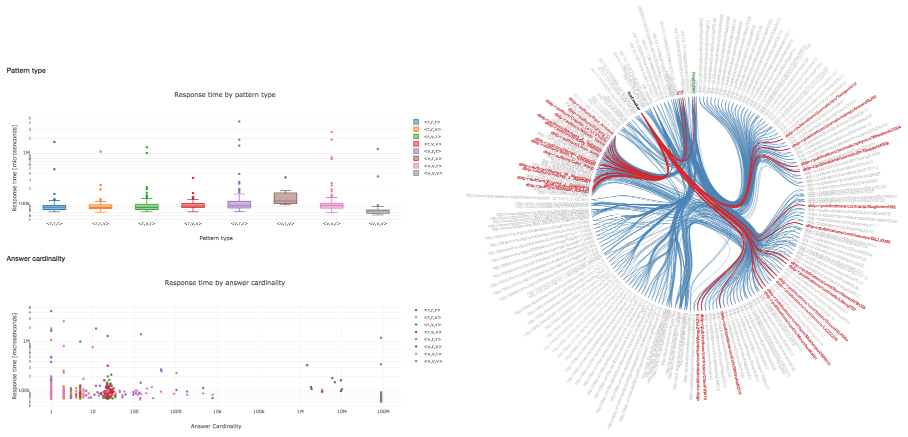
Fig. 1: *teepee*: Overview of the architecture for visualizing TPF performance.

2 Our Approach: *teepee*

teepee includes three components for visualizing the performance results of TPFs (cf. Figure 1): TPF Profiler, View Results, and Compare Results.
TPF Profiler [1]. The first component of *teepee* invokes the TPF Profiler. Given the URI of a TPF server and a sample size, the TPF Profiler collects performance measurements when submitting different requests to the server in three steps. First, the TPF Profiler randomly selects a sample of RDF triples from the KG via the TPF server. In the second step, a set of triple patterns is generated based on the sampled triples by replacing RDF terms with variables. In the third step, the TPF Profiler executes the triple patterns previously generated sample against the TPF server and record the measured response time per request.
View Results. After the measurement is completed, the results generated by the profiler can be viewed. The visualizations when viewing the results are separated into two major section. In the first section, visualization of the response time with respect to the triple pattern type and answer cardinality are shown. In the second section, the sample used for the measurement is visualized.
Compare Results. For the comparison of different measurements, *teepee* provides an A/B comparison visualization view. The view allows for selecting two results and visualizes the results next to each other. This facilitates, for instance, a direct comparison of different TPF server configurations.

3 Demonstration of Use Cases

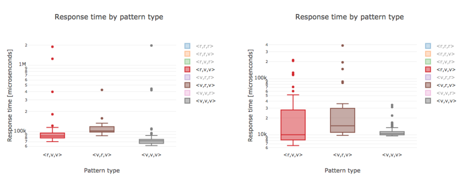
The attendees of the demo will learn how *teepee* can be used to gain insights into TPF server performance using its visualization capabilities. The two use cases can be followed using the online version of *teepee*[1] which includes the examples used in this demonstration. In the online demo, the users are encouraged to generate results themselves by configuring and running the TPF Profiler.
Impact of triple pattern properties on TPF performance. First, we select View Results to retrieve a list of all available results generated by the TPF Profiler. In the list, the examples are listed first. User-generated results are listed below (indicated by an ID starting with "U"). Next, we select View Results of Example 1. The view shows metadata on the top, indicating that the results are for the DBLP KG with a sample size of $m = 100$, page size of 100, HDT



(a) Response Time Visualization (b) Sample Visualization

Fig. 2: **View Results.** (a) Visualization of the response time according to the factors triple pattern type and answer cardinality. (b) Terms co-occurring in triple patterns generated from the sample are connected.

backend and a remote server. Below, the visualizations show the response times for analyzing different factors. As shown in Figure 2a, the response times are visualized as boxplots for the different triple pattern types. Hovering over the boxplot shows detailed information of the results. In the example, we can observe, that the triple pattern type has an impact on the response time as the boxplots vary in shape and position. The pattern type $\langle v, r, v\rangle$ shows the highest median response time. Moreover, the results are visualized according to the answer cardinalities of the triple patterns. The results of the example show a slight increase in the response time for larger answer cardinalities. Additionally, it can be observed that there are no triple patterns with an answer cardinality within the range of 1×10^4 and 1×10^6 answers. Most triple patterns show a cardinality below 50 which may allow for optimizing the server performance by reducing its page size. This assumption can be analyzed further using the A/B comparison visualizations presented in the next use case. Furthermore, information on the sampled triples by the TPF Profiler is provided as well. The number of samples is listed as well as the number of the unique triple patterns generated from the sample. Moreover, the number of distinct subjects, predicates, and objects in the triple patterns are provided allowing to gain an insight into the RDF term distribution. Shown in Figure 2b, all terms in the sample are listed and connected if they commonly occur in the same triple. Furthermore, links for all positions in the triples are added. Hovering over "Subjects" for instance will indicate all terms that occur as subjects in the sample. In the example, the term `foaf:maker` is selected and it can be observed that it occurs in a large number of triples within the sample.


(a) Boxplot A/B comparison

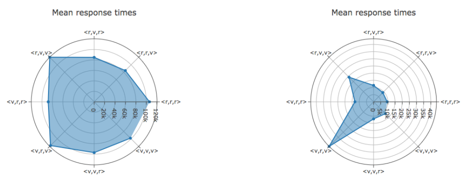

(b) Radar plot A/B comparison

Fig. 3: **Compare Results.** The figure shows an A/B comparison of the response times for the same KG differing in the server location (local vs. remote).

Impact of different servers via A/B comparison. Second, we select Compare Results to visualize the results in an A/B comparison to gain insights into how different TPF server configurations affect response times. Therefore, we select Example 2 and Example 6, which are the results for the DBpedia KG differing in the TPF server environment. The server in Example 2 is a publicly available TPF server hosted remotely at `http://data.linkeddatafragments.org`. The server in Example 6 is hosted locally. In Figure 3a, the response times for a selection of triple pattern types are shown with results for the remote server on the left and for the local server on the right. Two major insights can be gained from this visualization. First, we observe that the response time for the remote server is ≈ 10 times higher than for the local server. This indicates that network delays and server workload are influencing the response time for the remote server. Second, the response time for the pattern type $\langle v, v, v\rangle$ is lower with respect to the other pattern types on the remote server. This result suggests that the publicly available server caches the results for this triple pattern type. This is probably due to the fact that $\langle v, v, v\rangle$ is requested frequently, e.g. each time the web page of the TPF server is loaded. The radar plot in Figure 3b supports the comparison by visualizing various statistics of the response time.

4 Conclusions

This demo presents *teepee*, a tool that combines the TPF Profiler with visualization capabilities to enable analyses of TPF performance. *teepee* allows for gaining a better understanding of how different factors impact on the performance of TPF servers. In two use cases, the attendees are able to analyze and compare empirical results over well-known KGs using *teepee*.

References

1. Heling, L., Acosta, M., Maleshkova, M., Sure-Vetter, Y.: Querying large knowledge graphs over triple pattern fragments: An empirical study. In: ISWC (2018)
2. Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P.: Triple pattern fragments: A low-cost knowledge graph interface for the web. Web Semantics: Science, Services and Agents on the World Wide Web 37, 184–206 (2016)