How Representative Is a SPARQL Benchmark? An Analysis of RDF Triplestore Benchmarks

Muhammad Saleem
AKSW, Leipzig University
saleem@informatik.uni-leipzig.de

Gábor Szárnyas*
MTA-BME Lendület Cyber-Physical Systems Research Group
szarnyas@mit.bme.hu

Felix Conrads
DICE, Paderborn University
felix.conrads@upb.de

Syed Ahmad Chan Bukhari
Department of Pathology, Yale School of Medicine
ahmad.chan@yale.edu

Qaiser Mehmood
Insight Centre for Data Analytics, University of Ireland, Galway
qaiser.mehmood@insight-centre.org

Axel-Cyrille Ngonga Ngomo
DICE, Paderborn University
AKSW, Leipzig University
axel.ngonga@upb.de

ABSTRACT

Triplestores are data management systems for storing and querying RDF data. Over recent years, various benchmarks have been proposed to assess the performance of triplestores across different performance measures. However, choosing the most suitable benchmark for evaluating triplestores in practical settings is not a trivial task. This is because triplestores experience varying workloads when deployed in real applications. We address the problem of determining an appropriate benchmark for a given real-life workload by providing a fine-grained comparative analysis of existing triplestore benchmarks. In particular, we analyze the data and queries provided with the existing triplestore benchmarks in addition to several real-world datasets. Furthermore, we measure the correlation between the query execution time and various SPARQL query features and rank those features based on their significance levels. Our experiments reveal several interesting insights about the design of such benchmarks. With this fine-grained evaluation, we aim to support the design and implementation of more diverse benchmarks. Application developers can use our result to analyze their data and queries and choose a data management system.

ACM Reference Format:
Muhammad Saleem, Gábor Szárnyas, Felix Conrads, Syed Ahmad Chan Bukhari, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2019. How Representative Is a SPARQL Benchmark? An Analysis of RDF Triplestore Benchmarks. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313556

1 INTRODUCTION

The last years have witnessed a significant growth in the use of Linked Data and Semantic Web technologies. This growth has motivated the development of new triplestores with increasingly more efficient RDF storage and SPARQL query processing mechanisms. Accordingly, various benchmarks [2, 5, 8, 11, 14, 16, 18, 24, 27, 30, 31, 33] have been proposed to evaluate the querying performance of these triplestores. However, due to heterogeneity of RDF datasets and SPARQL queries, real-world applications often require customized deployments and experience different workloads when deployed in real environments [22]. Thus, designing a one-fit-all benchmark or selecting the most suitable benchmark for any given use-case and workload is not a straightforward task [12, 21, 24].

This work highlights key features of triplestore benchmarks pertaining to the three main components of benchmarks, i.e., datasets, queries, and performance metrics. State-of-the-art triplestore benchmarks are analyzed and compared against these features. Particularly, we consider triplestore benchmarks that rely on the native capabilities of the triplestores and do not require further reasoning over queries to get complete results. We also analyze the data and query logs of five real-world datasets selected from three different domains so as to provide a comparison with real-world datasets and queries. Our contributions are as follows.

(1) We identify key design features of SPARQL triplestore benchmarks based on our systematic survey on the state-of-the-art.
(2) We provide a detailed comparative analysis of the queries and datasets of 11 representative triplestore querying benchmarks: Train Benchmark [30], FEASIBLE [24], WatDiv [2], DBpedia SPARQL Benchmark (DBPSB) [18], FishMark [5], Bowlogna [11], SP2Bench [11], Berlin SPARQL Benchmark (BSBM) [8], BioBench [33], LDBC Social Network Benchmark Business Intelligence workload (SNB-BI) [31], and LDBC SNB Interactive workload (SNB-INT) [14].
(3) We analyze real data and corresponding user queries of five real-world datasets – DBpedia,[1] Semantic Web Dog Food (SWDF),[2] NCBI Gene,[3] SIDER,[4] DrugBank[5] – and compare them with the selected triplestore benchmarks.
(4) We measure the impact of various SPARQL query features (e.g., result sizes, triple patterns selectivity and number of join vertices) on the overall query execution time and rank these features according to their significance. In addition,

*Also with the Department of Measurement and Information Systems at the Budapest University of Technology and Economics.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313556

[1] DBpedia: http://dbpedia.org
[2] SWDF: https://old.datahub.io/dataset/semantic-web-dog-food
[3] NCBI Gene: https://www.ncbi.nlm.nih.gov/gene
[4] SIDER: http://sideeffects.embl.de/
[5] DrugBank: https://www.drugbank.ca/

we demonstrate that state-of-the-art triplestore benchmarks vary greatly, and highlight their current limitations.

(5) We performed extensive experiments and measure the impact of dataset *structuredness* (a well-known RDF dataset metric [12] formally defined in Section 2.1.1) on the overall query execution time as well as on the result sizes.

The rest of this paper is organized as follows: We provide an overview of the key RDF datasets and SPARQL query features that need to be considered while designing triplestore benchmarks based on a review of the state of the art. We then present a systematic survey of the current benchmarks for triplestores. The subsequent comparison of selected representative SPARQL benchmarks based on key data and query features identified in the survey is followed by a discussion of our results and some concluding remarks. The data and results presented in this evaluation are available online at https://github.com/dice-group/triplestore-benchmarks. The complete results can be reproduced and new benchmarks can be easily compared using the scripts provided at the project page.

2 BENCHMARK DESIGN FEATURES

In general, triplestore benchmarks comprise three main components: (1) a set of RDF datasets, (2) a set of SPARQL queries, and (3) a set of performance metrics. This section presents key features of each of these components that are important to consider in the development of triplestore benchmarks. Most of these features originate from state-of-the-art research contributions pertaining to triplestore benchmarks.

2.1 Datasets

Datasets used in triplestore benchmarks are either synthetic or selected from real-world RDF datasets [24]. The use of real-world RDF datasets is often regarded as useful to perform evaluation close to real-world settings [18]. Synthetic datasets are useful to test the scalability of systems based on datasets of varying sizes. Synthetic dataset generators are utilized to produce datasets of varying sizes which can often be optimized to reflect the characteristics of real-world datasets [12]. Previous works [12, 20] highlighted two key measures for selecting such datasets for triplestores benchmarking: (1) Dataset Structuredness, (2) Relationship Specialty. However, observations from the literature (see e.g., [12, 23]) suggest that other features such as varying number of triples, number of resources, number of properties, number of objects, number of classes, diversity in literal values, average properties and instances per class, average indegrees and outdegrees as well as their distribution across resources should also be considered.

2.1.1 Dataset Structuredness: Duan et al. [12] combine many of the aforementioned dataset features into a single composite metric called dataset *structuredness* or *coherence*. This metric measures how well a dataset's classes (i.e., `rdf:type`) are covered by the different instances of the dataset. The structuredness value for any given dataset lies between $[0, 1]$, where 0 stands for lowest possible structure and 1 points to a highest possible structured dataset. They conclude that synthetic datasets are highly structured while real-world datasets have structuredness values ranging from low to high, covering the whole structuredness spectrum. Formally, dataset structuredness is defined in the form of class coverage. The coverage of a class C, denoted by $CV(C)$, is defined as follows [12]:

Definition 2.1 (Class Coverage). Let D be a dataset. Moreover, let $P(C)$ denote the set of distinct properties of class C, and $I(C)$ denote the set of distinct instances of the class C. Let $I(p, C)$ count the number of entities for which property p has its value set in the instances of C. Then, the coverage of the class $CV(C)$ is

$$CV(C) = \frac{\sum_{p \in P(C)} I(p, C)}{|P(C)| \cdot |I(C)|}$$

In general, RDF datasets comprise multiple classes with a varying number of instances for different classes. The authors of [12] proposed a mechanism that considers the weighted sum of the coverage $CV(C)$ of individual classes. For each class C, the weighted coverage is defined below.

Definition 2.2 (Weighted Class Coverage). The weighted coverage for a class C denoted by $WTCV(C)$ is calculated as:

$$WTCV(C) = \frac{|P(C)| + |I(C)|}{\sum_{C' \in D} |P(C')| + |I(C')|}$$

By using Definitions 2.1 and 2.2, we are now ready to compute the structuredness of a dataset D.

Definition 2.3 (Dataset Structuredness). The overall structuredness or coherence of a dataset D denoted by $CH(D)$ is defined as

$$CH(D) = \sum_{C \in D} CV(C) \cdot WTCV(C)$$

2.1.2 Relationship Specialty. In datasets, some attributes are more common and associated with many resources. In addition, some attributes are multi-valued, e.g., a person can have more than one cellphone number or professional skill. The number of occurrences of a predicate associated with each resource in the dataset provides useful information on the graph structure of an RDF dataset, and makes some resources distinguishable from others [20]. In real datasets, this kind of relationship specialty is commonplace. For example, several million people can like the same movie. Likewise, a research paper can be cited in several hundred of other publications. Qiao et al. [20] suggest that synthetic datasets are limited in how they reflect this relationship specialty. This is either due to the simulation of uniform relationship patterns for all resources, or a random relationship generation process. The relationship specialty of a *relationship predicate* is defined as follows:

Definition 2.4 (Predicate Relationship Specialty). Let d be the distribution that records the number of occurrences of a *relationship predicate* r associated with each resource and μ is the mean and σ is the standard deviation of d. The specialty value of r denoted as $\kappa(r)$ is defined as the Pearson's Kurtosis value of the distribution d.

$$\kappa(r) = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \cdot \frac{\sum_{x_i \in d}(x_i - \mu)^4}{\sigma^4} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

Where n is the number of available values, i.e., sample size. The relationship specialty of a dataset is defined in the form of a weighted sum of specialty values of all relationship predicates:

Definition 2.5 (Dataset Relationship Specialty). The relationship specialty of dataset D denoted by $RS(D)$ is calculated as follows:

$$RS(D) = \sum_{r_i \in R} \frac{|T(r_i)| \cdot \kappa(r_i)}{\sum_{r_j \in R} |T(r_j)|}$$

where $|T(r_i)|$ is the number of triples in the dataset having predicate r_i, $\kappa(r_i)$ is the specialty value of relationship predicate r_i.

The dataset structuredness and relationship specialty directly affect the result size, the number of intermediate results, and the selectivities of the triple patterns of the given SPARQL query. Therefore, they are important dataset design features to be considered during the generation of benchmarks [12, 20, 24].

2.2 SPARQL Queries

The literature about SPARQL Queries [2, 15, 23, 24, 26] suggests that a SPARQL querying benchmark should vary the queries with respect to various features such as *query characteristics*: number of triple patterns, number of projection variables, result set sizes, query execution time, number of BGPs, number of join vertices, mean join vertex degree, mean triple pattern selectivities, BGP-restricted and join-restricted triple pattern selectivities, join vertex types, and highly used SPARQL clauses (e.g., `LIMIT`, `OPTIONAL`, `ORDER BY`, `DISTINCT`, `UNION`, `FILTER`, `REGEX`). All of these features have a direct impact on the runtime performance of triplestores. We assume that the reader is familiar with the basic concepts of SPARQL, including the notions of a triple pattern, a basic graph pattern (BGP), and projection variables.[6] In the following, we define the remaining SPARQL features formally, i.e., the number of join vertices, mean join vertex degree, join vertex types, triple pattern selectivities, BGP-restricted and join-restricted triple pattern selectivities.

We represent any basic graph pattern (BGP) of a given SPARQL query as a *directed hypergraph* (DH) [25], a generalization of a directed graph in which a hyperedge can join any number of vertices. In our specific case, every hyperedge captures a triple pattern. The subject of the triple becomes the source vertex of a hyperedge and the predicate and object of the triple pattern become the target vertices. For instance, the query (Figure 1) shows the hypergraph representation of a SPARQL query. Unlike a common SPARQL representation where the subject and object of the triple pattern are connected by an edge, our hypergraph-based representation contains nodes for all three components of the triple patterns. As a result, we can capture joins that involve predicates of triple patterns. Formally, our hypergraph representation is defined as follows:

Definition 2.6 (Directed hypergraph of a BGP). The hypergraph representation of a BGP B is a directed hypergraph $HG = (V, E)$ whose vertices are all the components of all triple patterns in B, i.e., $V = \bigcup_{(s,p,o) \in B} \{s, p, o\}$, and that contains a hyperedge $(S, T) \in E$ for every triple pattern $(s, p, o) \in B$ such that $S = \{s\}$ and $T = (p, o)$.

The representation of a complete SPARQL query as a DH is the union of the representations of the query's BGPs. Based on the DH representation of SPARQL queries, we can define the following features of SPARQL queries:

[6] See https://www.w3.org/TR/sparql11-query/ for the corresponding definitions.

```
SELECT DISTINCT * WHERE {
 ?drug :drugType :smallMolecule .
 ?enzyme :xSubstrate ?compound  .
 ?chemReac :equation ?chemEq    .
 ?drug :description ?drugDesc    .
 ?drug :keggCompoundId ?compound .
 ?chemReac :xEnzyme ?enzyme      .
 ?chemReac :title ?reacTitle     }
```

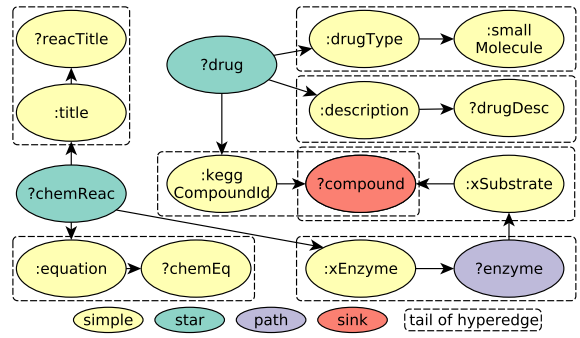


Figure 1: Directed hypergraph representation of a SPARQL query. Prefixes are ignored for simplicity.

Definition 2.7 (Join Vertex). For every vertex $v \in V$ in such a hypergraph we write $E_{\text{in}}(v)$ and $E_{\text{out}}(v)$ to denote the set of incoming and outgoing edges, respectively; i.e., $E_{\text{in}}(v) = \{(S, T) \in E \mid v \in T\}$ and $E_{\text{out}}(v) = \{(S, T) \in E \mid v \in S\}$. If $|E_{\text{in}}(v)| + |E_{\text{out}}(v)| > 1$, we call v a *join vertex*.

Definition 2.8 (Join Vertex Degree). Based on the DH representation of the queries the join vertex degree of a vertex v is $\mathcal{JVD}(v) = |E_{in}(v)| + |E_{out}(v)|$, where $E_{in}(v)$ resp. $E_{out}(v)$ is the set of incoming resp. outgoing edges of v.

Definition 2.9 (Join Vertex Types). A vertex $v \in V$ can be of type *star*, *path*, *hybrid*, or *sink* if this vertex participates in at least one join. A *star* vertex has more than one outgoing edge and no incoming edges. A *path* vertex has exactly one incoming and one outgoing edge. A *hybrid* vertex has either more than one incoming and at least one outgoing edge or more than one outgoing and at least one incoming edge. A *sink* vertex has more than one incoming edge and no outgoing edge. A vertex that does not participate in joins is *simple*.

Definition 2.10 (Triple Pattern Selectivity). Let tp_i be a triple pattern of a SPARQL query Q and D be a dataset. Furthermore, let N be the total number of triples in D and $Card(tp_i, D)$ be the cardinality of tp_i w.r.t. D, i.e., total number of triples in D that matches tp_i, then the selectivity of tp_i w.r.t. D denoted by $Sel(tp_i, D) = Card(tp_i, D)/N$.

Definition 2.11 (BGP-Restricted Triple Pattern Selectivity). Consider a Basic Graph Pattern BGP and a triple pattern tp_i belonging to BGP, let $R(tp_i, D)$ be the set of distinct solution mappings (i.e., result set) of executing tp_i over dataset D and $R(BGP, D)$ be the set of distinct solution mappings of executing BGP over dataset D. Then the BGP-restricted triple pattern selectivity denoted by $Sel_{BGP\text{-}Restricted}(tp_i, D)$ is the fraction of distinct solution mappings in $R(tp_i, D)$ that are compatible (as per standard SPARQL semantics [3]) with a solution mapping in $R(BGP, D)$ [2]. Formally, if Ω and Ω' denote the sets underlying the (bag) query results $R(tp_i, D)$ and $R(BGP, D)$, respectively, then

$$Sel_{BGP\text{-}R.}(tp_i, D) = \frac{|\{\mu \in \Omega \mid \exists \mu' \in \Omega' : \mu \text{ and } \mu' \text{ are compatible}\}|}{|\Omega|}$$

Definition 2.12 (Join-Restricted Triple Pattern Selectivity). Consider a join vertex x in the DH representation of a *BGP*. Let BGP' belonging to *BGP* be the set of triple patterns that are incidents to x. Furthermore, let tp_i belonging to BGP' be a triple pattern and $R(tp_i, D)$ be the set of distinct solution mappings of executing tp_i over dataset D and $R(BGP', D)$ be the set of distinct solution mappings of executing BGP' over dataset D. Then the *x-restricted* triple pattern selectivity denoted by $Sel_{\mathcal{J}Vx\text{-}Restricted}(tp_i, D)$, is the fraction of distinct solution mappings in $R(tp_i, D)$ that are compatible with a solution mapping in $R(BGP', D)$ [2]. Formally, if Ω and Ω' denote the sets underlying the (bag) query results $R(tp_i, D)$ and $R(BGP', D)$, respectively, then

$$Sel_{\mathcal{J}Vx\text{-}R.}(tp_i, D) = \frac{|\{\mu \in \Omega \mid \exists \mu' \in \Omega' : \mu \text{ and } \mu' \text{ are compatible}\}|}{|\Omega|}$$

All of the above important query features were collected from the previous works [2, 15, 23, 24] in triplestores benchmarking. Finally, we combine all these important query features into a single composite metric called the *Diversity Score* of the benchmark queries, defined as follows.

Definition 2.13 (Queries Diversity Score). Let μ_i be the mean and σ_i the standard deviation of a given distribution w.r.t. the ith feature of the said distribution. The overall diversity score *DS* of the queries is the average coefficient of variation of all the query features k analyzed in the queries of benchmark B:

$$DS = \frac{1}{k} \sum_{i=1}^{k} \frac{\sigma_i(B)}{\mu_i(B)}$$

2.3 Performance Metrics

Based on the previous triplestores benchmarks and performance evaluations [2, 5, 8, 10, 11, 14, 16, 18, 24, 27, 30, 31, 33] the performance metrics for such comparisons can be categorized as:

- **Query Processing Related**: The performance metrics in this category are related to the query processing capabilities of the triplestores. The query execution time is the central performance metric in this category. However, reporting the execution time for individual queries might not be feasible due to the large number of queries in the given benchmark. To this end, Query Mix per Hour (QMpH) and Queries per Second (QpS) are regarded as central performance measures to test the querying capabilities of the triplestores [8, 18, 24]. In addition, the query processing overhead in terms of the CPU and memory usage is important to measure during the query executions [27]. This also includes the number of intermediate results, the number of disk/memory swaps, etc.
- **Data Storage Related:** Triplestores need to load the given RDF data and mostly create indexes before they are ready for query executions. In this regard, the data loading time, the storage space acquired, and the index size are important performance metrics in this category [8, 11, 27, 33].
- **Result Set Related:** Two systems can only be compared if they produce exactly the same results. Therefore, result set correctness and completeness is important metrics to be considered in the triplestores evaluations [8, 24, 27, 33].
- **Parallelism with/without Updates:** Some of the aforementioned triplestores performance evaluations [8, 10, 33] also measured the parallel query processing capabilities of the triplestores by simulating workloads from multiple querying agents with and without dataset updates.

We analyzed state-of-the-art existing SPARQL triplestore benchmarks across all of the above mentioned dataset and query features as well as the performance metrics. The results are presented in Section 4.

3 SYSTEMATIC SURVEY

In this section, we present a systematic survey carried out to collect triplestore benchmarks and their selection criteria for further analysis. We conducted a public survey[7] through various relevant W3C Linked Open Data mailing list[8] and Semantic Web mailing list[9] with a request to participate email. We received 14 responses[10] regarding SPARQL triplestore benchmarks. Moreover, we used Google Scholar to retrieve published research work relating to the design of triplestore benchmarks and/or their performance evaluation. Initially, we selected 40 relevant papers[11] and evaluated them against our designed inclusion criteria. In our inclusion criteria we mandated that (1) the benchmark target the query runtime performance evaluation of triplestores, (2) both RDF data and SPARQL queries of the benchmark are publicly available or can be generated (3) the queries must not require reasoning to retrieve the complete results. After manual evaluation, we found 11 benchmarks (7 with synthetic and 4 with real data) that fulfilled our requirements. The sections below provide details of the selected benchmarks.

3.1 Synthetic Triplestore Benchmarks

The *Train Benchmark* (TrainBench) [30] uses a data generator that produces railway networks in increasing sizes and serializes them in different formats, including RDF. *The Waterloo SPARQL Diversity Test Suite* (WatDiv) [2] provides a synthetic data generator that produces RDF data with a tunable structuredness value and a query generator. The queries are generated from different query templates. *SP2Bench* [27] mirrors vital characteristics (such as power law distributions or Gaussian curves) of the data in the DBLP bibliographic database. *The Berlin SPARQL Benchmark* (BSBM) [8] uses query templates to generate any number of SPARQL queries for benchmarking, covering multiple use cases such as explore, update, and business intelligence. *Bowlogna* [11] models a real-world setting derived from the Bologna process and offers mostly analytic queries reflecting data-intensive user needs. The *LDBC Social Network Benchmark* (SNB) defines two workloads: (1) the *Interactive* workload (SNB-INT) measures the evaluation of graph patterns in a localized scope (e.g., in the neighborhood of a person), with the graph being continuously updated [14], and (2) the *Business Intelligence* workload (SNB-BI) focuses on queries that mix complex graph pattern matching with aggregations, touching on a significant portion of the graph [31], without any updates. Note that these

[7] Survey: https://goo.gl/R59uoM
[8] public-lod@w3.org
[9] semantic-web@w3.org
[10] Responses: https://goo.gl/JdivN3
[11] SPARQL benchmarking studies: https://goo.gl/cCu85z

two workloads are regarded as two separate triplestore benchmarks based on the same dataset.

3.2 Triplestore Benchmarks Using Real Data

FEASIBLE [24] is a cluster-based SPARQL benchmark generator, which is able to synthesize customizable benchmarks from the query logs of SPARQL endpoints. The *DBpedia SPARQL Benchmark* (DBPSB) [18] is another cluster-based approach that generates benchmark queries from DBpedia query logs, but employs different clustering techniques than FEASIBLE. The *FishMark* [5] dataset is obtained from FishBase[12] and provided in both RDF and SQL versions. The SPARQL queries were obtained from logs of web-based FishBase application. *BioBench* [33] evaluates the performance of RDF triplestores with the real biological datasets and queries from five different real-world RDF datasets[13], i.e., Cell, Allie, PDBJ, DDBJ, and UniProt. Due to the size of the datasets, we were only able to analyze the combined data and queries of the first three.

3.3 Selected Real-World Datasets

As mentioned before, we aimed to analyze the data and queries of real-world datasets and compare them to those of the benchmark datasets and queries. The selection criteria for the real-world datasets were: (1) The RDF datasets must be publicly available, (2) the real queries posted by users of the datasets via SPARQL endpoints should be available. We were able to get real log queries of the Bio2RDF datasets,[14] DBpedia, and Semantic Web Dog Food. Our goal was to select real-world datasets from different domains. Hence, we selected DBpedia[15] and SWDF and three datasets – NCBIGene, Sider, DrugBank from Bio2RDF. The selection of the three Bio2RDF datasets was based on a recommendation from domain experts. The well-known DBpedia dataset is the RDF version of Wikipedia. The SWDF represents the publication from Semantic Web and Linked Data as RDF. NCBIGene provides genetic information from a wide range of species. SIDER contains information on marketed medicines and their recorded side-effects. DrugBank knowledge base contains information about drugs, their composition and their interactions.

Table 1 shows statistics from selected datasets of the benchmarks and real-world datasets. More advanced statistics will be presented in the next section. The table also shows the number of SPARQL queries of the datasets included in the corresponding benchmark or query log. It is important to mention that we only selected SPARQL SELECT queries for analysis. This is because we wanted to analyze the triplestore benchmarks for their query runtime performance and most of these benchmarks only contain SELECT queries [24]. For the synthetic benchmarks that include data generators, we chose the datasets used in the evaluation of the original paper that were comparable in size to the datasets of other synthetic benchmarks. For template-based query generators such as WatDiv, DBPSB, SNB, we chose one query per available template. For FEASIBLE, we generated a benchmark of 50 queries from DBpedia log to be comparable with a well-known WatDiv benchmark that includes 20 basic testing query templates, and 30 extensions for testing.[16]

	Benchmark	Subjects	Predicates	Objects	Triples	Queries
Synthetic	Bowlogna [11]	2,151k	39	260k	12M	16
	TrainB. [30]	3,355k	16	3,357k	41M	11
	BSBM [8]	9,039k	40	14,966k	100M	20
	SP2Bench [27]	7,002k	5,718	19,347k	49M	14
	WatDiv [2]	5,212k	86	9,753k	108M	50
	SNB [14, 31]	7,193k	40	17,544k	46M	21
Real	FishMark [5]	395k	878	1,148k	10M	22
	BioBench [33]	278,007k	299	232,041k	1,451M	39
	FEASIBLE [24]	18,425k	39,672	65,184k	232M	50
	DBPSB [18]	18,425k	39,672	65,184k	232M	25
Datasets	DBpedia3.5.1	18,425k	39,672	65,184k	232M	35,803
	SWDF	36k	185	95k	0.3M	71,406
	NCBIGene	15,614k	57	82,688k	159M	3,644
	SIDER	1,222k	39	5,952k	17M	26,048
	DrugBank	316k	105	1,828k	3M	50,877

Table 1: High-level statistics of the data and queries used on our evaluation. Both SNB-BI and SNB-INT use the same dataset and are therefore named as SNB for simplicity.

4 ANALYSIS OF THE BENCHMARKS

We present a detailed analysis of the datasets, queries, and performance metrics of the selected benchmarks and datasets according to the design features presented in Section 2.

4.1 Datasets

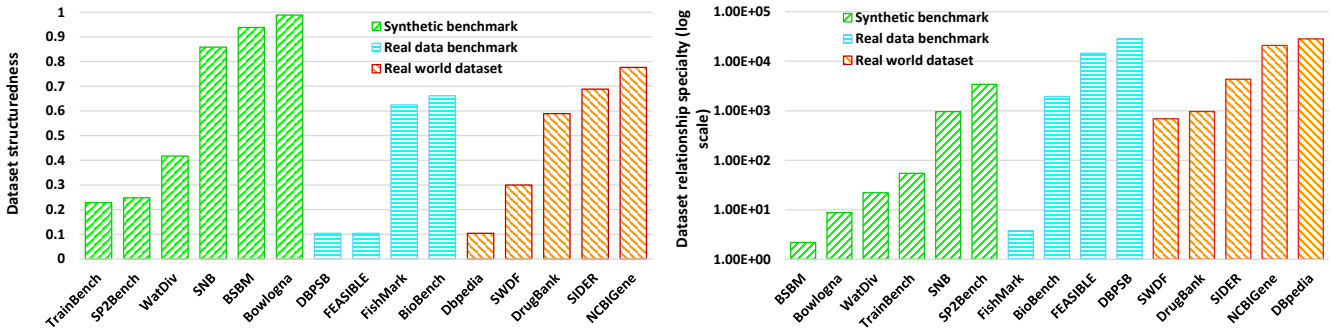
We presents results pertaining to the dataset features of Section 2.1.

4.1.1 Structuredness. Figure 2a shows the structuredness values of the selected benchmarks and real-world datasets. Duan et al. [12] establish that synthetic benchmarks are highly structured while real-world datasets are low structured. This important dataset feature is well-covered in recent synthetic benchmarks such as TrainBench (with a structuredness value of 0.23) and WatDiv, which lets the user generate a benchmark dataset of a desired structuredness value. However, Bowlogna (0.99), BSBM (0.94), and SNB (0.86) have relatively high structuredness values. The average structuredness value of the selected five real-world datasets is 0.49, and 0.65 for the 13 real-world datasets used in LargeRDFBench [23]. Finally, on average, synthetic benchmarks are still more structured than real data benchmarks (0.61 vs. 0.45).

4.1.2 Relationship Specialty. According to [20], relationship specialty in synthetic datasets is limited, i.e., the overall relationship specialty values of synthetic datasets are lower than those of similar real-world datasets. The dataset relationship specialty results presented in Figure 2b mostly confirm this behavior. On average, synthetic benchmarks have a smaller specialty score than real-world datasets (744 vs. 11098). The relationship specialty values of Bowlogna (8.7), BSBM (2.2), and WatDiv (22.0) are on the lower side compared to real-world datasets. The highest specialty value (28282.8) is recorded in the DBpedia dataset.

An important issue is the correlation between structuredness and the relationship specialty of the datasets. To this end, we computed

[12] FishBase: http://fishbase.org/search.php
[13] BioBench: http://kiban.dbcls.jp/togordf/wiki/survey#data
[14] Bio2RDF: http://download.bio2rdf.org/files/release/3/release.html
[15] Version 3.5.1 as used in FEASIBLE and DBPSB
[16] WatDiv query templates: http://dsg.uwaterloo.ca/watdiv/



(a) Structuredness **(b) Relationship Specialty**

Figure 2: Analysis of the datasets of triplestore benchmarks and real-world data.

the Spearman's correlation between the stucturedness and specialty values of all the selected benchmarks and real-world datasets. The correlation of the two measures is −0.5, indicating a moderate inverse relationship. This means that the higher the structuredness, the lower the specialty value. This is because in highly structured datasets, data is generated according to a specific distribution without treating some predicates more particularly (in terms of occurrences) than others.

4.2 Queries

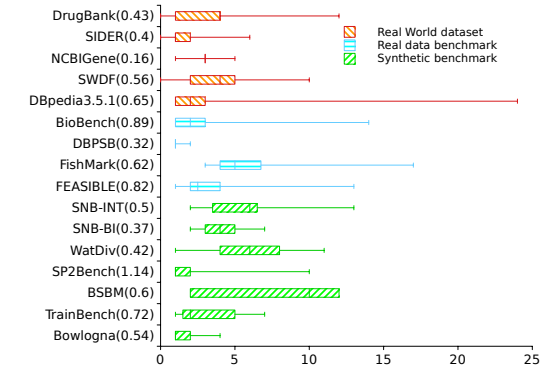
This section presents results pertaining to the query features discussed in Section 2.2. Figure 3 shows the box plot distributions of real-world datasets and benchmark queries across the query features defined in Section 2. The values inside the brackets, e.g., the 0.89 in "BioBench (0.89)", show the diversity score (Definition 2.13) of the benchmark or real-world dataset for the given query feature.

Starting from the number of projection variables (ref. Figure 3a), the NCBIGene dataset has the lowest diversity score (0.16) and SP2Bench has the highest score of 1.14. The mean diversity score (across all benchmarks and real-world datasets) for this feature is 0.59 and hence the diversity scores of DBPSB, SNB-BI, SNB-INT, WatDiv, and Bowlogna are below the average value. Even though the diversity score of BSBM is above average, the distribution shows that the values mostly lie in the second quartile of the box plot. The average diversity score of the number of join vertices (ref. Figure 3b) is 1.39 and hence the diversity scores of the Bowlogna, FishMark, WatDiv, BSBM, TrainBench, BioBench, DBPSB, SNB-BI, and SNB-INT benchmarks are below the average value. It is important to mention that the highest number of join vertices recorded in a query is 51 in the SNB-BI benchmark. The average diversity score of the number of triple patterns (ref. Figure 3c) is 0.75 and hence the diversity scores of the FishMark, Bowlogna, BSBM, and WatDiv benchmarks are below the average value. The average diversity score of the result sizes (ref. Figure 3d) is 11.89 and hence the diversity scores of all benchmarks are below the average value. The average diversity score of the join vertex degree (ref. Figure 3e) is 1.08 and hence the diversity scores of all benchmarks except FEASIBLE are below the average value. The average diversity score of the triple pattern selectivity (ref. Figure 3f) is 3.17 and hence the diversity scores of all benchmarks except FEASIBLE are below the average. The average diversity score of the join-restricted triple pattern selectivity (ref. Figure 3g) is 1.39 and hence the diversity scores of all benchmarks except FEASIBLE and BSBM are below the average value. The average diversity score of the BGP-restricted triple pattern selectivity (ref. Figure 3h) is 4.11 and hence the diversity scores of all benchmarks except WatDiv are below the average value. The average diversity score of the number of BGPs (ref. Figure 3i) is 0.63 and hence the diversity scores of SNB-BI, SNB-INT, BSBM, SP2BENCH, TrainBench, WatDiv, and Bowlogna are below the average value.

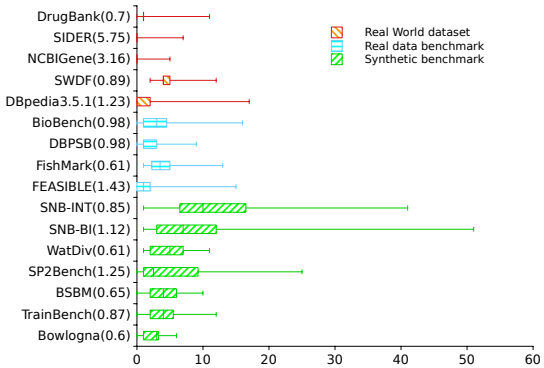
The Linked SPARQL Queries (LSQ) [22] representation stores additional SPARQL features, such as use of `DISTINCT`, `REGEX`, `BIND`, `VALUES`, `HAVING`, `GROUP BY`, `OFFSET`, aggregate functions, `SERVICE`, `OPTIONAL`, `UNION`, property paths, etc. We make a count of all of these SPARQL operators and functions and use it as a single query dimension as number of LSQ features. The average diversity score of the number of LSQ features (ref. Figure 3j) is 0.30, and hence only the diversity scores of SNB and WatDiv are below average value. Finally, the average diversity score of the query runtimes is 12.87 (ref. Figure 3k), and hence the diversity scores of all benchmarks are below average value.

In summary, FEASIBLE's diversity scores are below the average values in 3 of the 11 features, followed by BioBench with 7/11. These are followed by SP2Bench, TrainBench, and BSBM with 8/11 each. The next is FishMark 9/11 and then Bowlogna, WatDiv, SNB-BI, SNB-INT, and DBPSB with 10/11 each. Figure 3l shows the overall (across all the features, ref. Definition 2.13) diversity scores of the benchmarks and real-world datasets. In the benchmarks category, FEASIBLE produces the most diverse benchmarks (diversity score 2.15), followed by BioBench (1.51), FishMark (1.33), WatDiv (1.32), Bowlogna (1.23), SP2Bench (1.22), BSBM (1.08), DBPSB (1.03), SNB-BI (0.90), SNB-INT (0.863) and TrainBench (0.79).

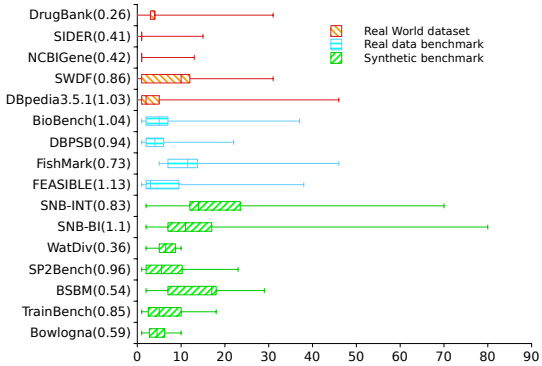
Table 2 shows the percentage coverage of widely used [22] SPARQL clauses and join vertex types for each benchmark and real-world dataset. We highlighted cells for benchmarks that either completely miss or overuse certain SPARQL clauses and join vertex types. TrainBench and WatDiv queries mostly miss the important SPARQL clauses. All of FishMark's queries contain at least one "Star" join node. The distribution of other SPARQL clauses, such as subquery, `BIND`, aggregates, solution modifiers, property paths, and services are provided in the LSQ versions of each of the benchmark queries, available from the project website.



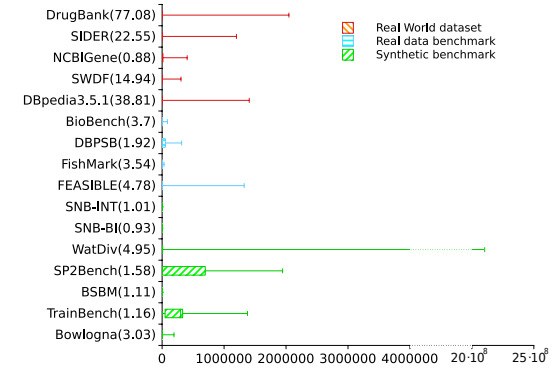
(a) No. of projection variables



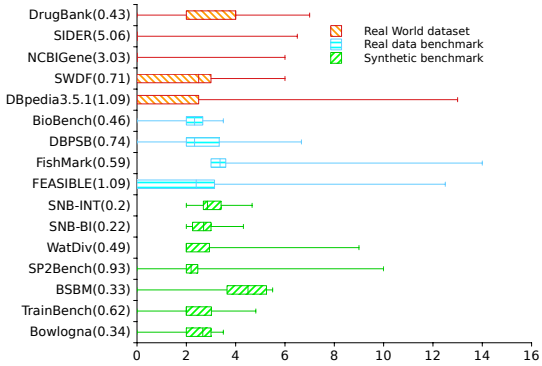
(b) No. of join vertices



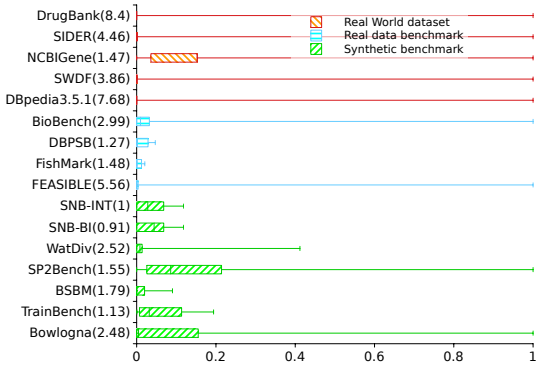
(c) No. of triple patterns



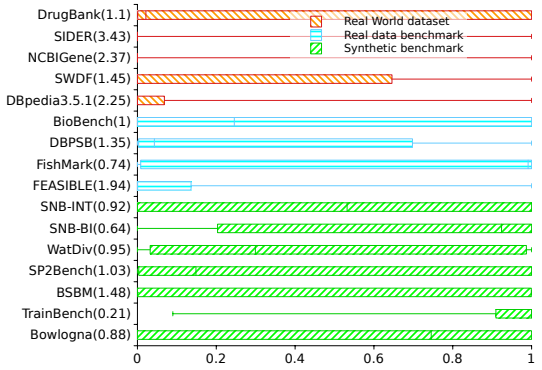
(d) Result size



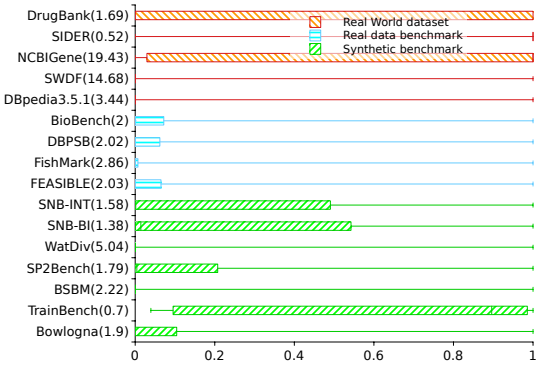
(e) Join vertex degree



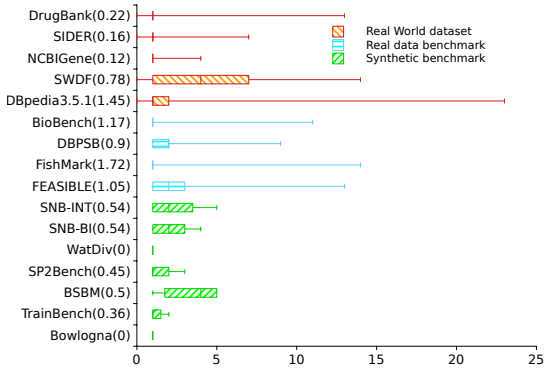
(f) Triple pattern (TP) selectivity



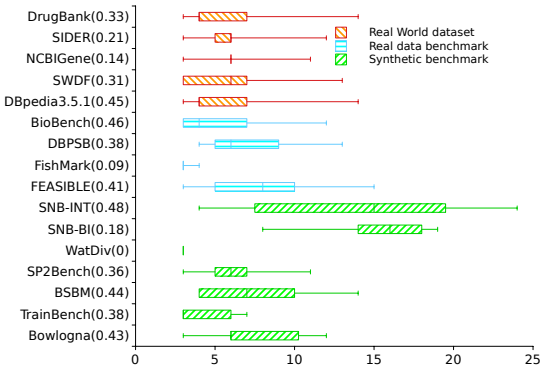
(g) Join-restricted TP selectivity



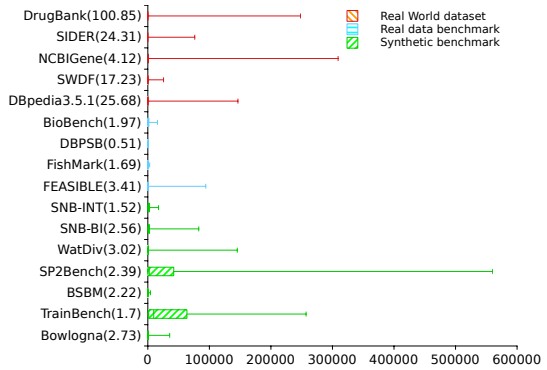
(h) BGP-restricted TP selectivity



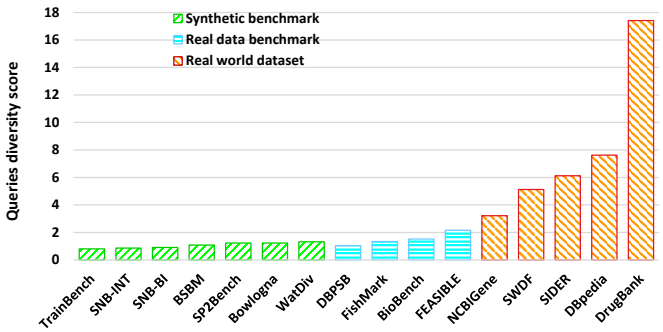
(i) No. of BGPs



(j) No. of LSQ features



(k) Runtimes (ms)



(l) Overall diversity score

Figure 3: Analysis of queries used in triplestore benchmarks and for real-world datasets.

		Distributions of SPARQL Clauses							Distr. of Join Vertex Type				
	Benchmark	DIST	FILT	REG	OPT	UN	LIM	ORD	Star	Path	Sink	Hyb.	N.J.
Synthetic	Bowlogna	6.2	37.5	6.2	0.0	0.0	6.2	6.2	93.7	37.5	62.5	25.0	6.2
	TrainB.	0.0	45.4	0.0	0.0	0.0	0.0	0.0	81.8	27.2	72.7	45.4	18.1
	BSBM	30.0	65.0	0.0	65.0	10.0	45.0	45.0	95.0	60.0	75.0	60.0	5.0
	SP2Bench	42.8	57.1	0.0	21.4	14.2	7.1	14.2	78.5	35.7	50.0	28.5	14.2
	Watdiv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	28.0	64.0	26.0	20.0	0.0
	SNB-BI	0.0	61.9	4.7	52.3	14.2	80.9	100.0	90.4	38.1	80.9	52.3	0.0
	SNB-INT	0.0	47.3	0.0	31.5	15.7	63.15	78.9	94.7	42.1	94.7	84.2	0.0
Real	FEASIBLE	56.0	58.0	22.0	28.0	40.0	42.0	32.0	58.0	18.0	36.0	16.0	30.0
	Fishmark	0.0	0.0	0.0	9.0	0.0	0.0	0.0	100.0	81.8	9.0	72.7	0.0
	DBPSB	100.0	48.0	8.0	32.0	36.0	0.0	0.0	68.0	20.0	32.0	20.0	24.0
	BioBench	28.2	25.6	15.3	7.6	7.6	20.5	10.2	71.7	53.8	43.5	38.4	15.3
Datasets	DBpedia	34.7	38.4	8.2	43.0	14.8	2.2	0.1	39.9	10.7	10.5	1.3	52.2
	SWDF	71.6	2.4	1.0	46.5	55.1	2.4	40.2	65.7	21.9	44.3	54.0	28.7
	NCBIGene	4.3	4.9	0.1	0.6	0.2	90.4	0.1	7.4	2.5	0.4	0.0	89.8
	SIDER	79.3	7.8	0.0	0.0	0.2	79.0	0.0	3.4	0.8	0.5	0.1	96.0
	DrugBank	53.7	2.1	0.2	0.0	2.8	14.4	0.0	89.7	23.2	23.4	0.0	10.0
	Overall	33.8	30.3	4.4	20.4	13.0	26.0	16.5	64.7	33.0	37.8	28.9	26.0

Table 2: Coverage of SPARQL clauses and join vertex types for each benchmark in percentages. SPARQL clauses: `DIST[INCT]`, `FILT[ER]`, `REG[EX]`, `OPT[IONAL]`, `UN[ION]`, `LIM[IT]`, `ORD[ER BY]`. Join vertex types: Star, Path, Sink, Hyb[rid], N[o] J[oin]. Missing ● and overused ● features are highlighted.

4.3 Performance Metrics

This section presents results pertaining to the performance metrics discussed in Section 2.3. Table 3 shows the performance metrics used by the selected benchmarks to compare triplestores. The query runtimes for complete benchmark's queries is the central performance metrics and is used by all of the selected benchmarks. In addition, the QpS and QMpH are commonly used in the query processing category. We found that in general, the processing overhead generated by query executions is not paid much attention as only SP2Bench measures this metric. In the "storage" category, the time taken to load the RDF graph into triplestore is most common. The in-memory/HDD space require to store the dataset and corresponding indexes did not get much attention. The result set correctness and completeness are important metrics to be considered when there are large number of queries in the benchmark and composite metrics such as QpS and QMpH are used. We can see many of the benchmarks do not explicitly check these two metrics. They mostly assume that the results are complete and correct. However, this always might not be the case [23]. Additionally, only BSBM considers the evaluation of triplestores with simultaneous user requests with updates. However, benchmarks execution frameworks such IGUANA [10] can be used to measure the parallel query processing capabilities of triplestores in presence of multiple querying and update agents.

4.4 Impact of Dataset Structuredness

The dataset structuredness has been regarded as one of the most important RDF dataset feature [12]. However, to the best of our knowledge, the impact of dataset structuredness on query runtimes

		Processing			Storage			Result Set		Additional	
	Benchmark	QpS	QMpH	PO	LT	SS	IS	RCm	RCr	MC	DU
Synthetic	Bowlogna	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗
	TrainBench	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓
	BSBM	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
	SP2Bench	✗	✗	✓	✓	✓	✗	✓	✓	✗	✗
	WatDiv	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	SNB-BI	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
	SNB-INT	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓
Real	FEASIBLE	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
	Fishmark	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
	DBPSB	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
	BioBench	✗	✗	✗	✓	✓	✗	✓	✗	✓	✗

Table 3: Metrics used in the selected benchmarks pertaining to query processing, data storage, result set, simultaneous multiple client requests, and dataset updates. *QpS:* Queries per Second, *QMpH:* Queries Mix per Hour, *PO:* Processing Overhead, *LT:* Load Time, *SS:* Storage Space, *IS:* Index Sizes, *RCm:* Result Set Completeness, *RCr:* Result Set Correctness, *MC:* Multiple Clients, *DU:* Dataset Updates.

and result set sizes has not been measured in the literature. To measure this impact, we need to create synthetic datasets of varying structuredness values, all following the same dataset description model or schema and comparably to each others in terms of their sizes. We then need to execute the same set of benchmark queries over the generated datasets and measure the result set sizes and query runtimes. The WatDiv benchmark data generator allows the control of the size of generated datasets as well as the structuredness values of individual entities. However, the task of generating the datasets with exact sizes and structuredness values are difficult to achieve due to the scaling and structuredness factors used to control the size and structuredness of the overall dataset. We generated 10 datasets of varying structuredness values given in Table 4.

We run complete WatDiv queries on the individual datasets by using Virtuso triplestore and measured the result set sizes and runtimes of individual queries. Figure 4 shows the impact of the dataset structuredness values on query runtimes and result set sizes. We can clearly see there is a positive correlation of dataset sturcturedness values and the query runtimes as-well-as result sizes, i.e. the higher the structuredness value of the dataset the higher the result sizes and query runtimes. The result also suggests that there is a direct correlation with the result sizes and the query runtimes.

4.5 Correlation of Query Features vs. Runtimes

In previous sections, we presented the results of some important SPARQL query features that should be considered while designing SPARQL benchmarks. These features were mostly taken from previous works [2, 18, 24] in SPARQL benchmarking. However, an important question to investigate is how significantly these features impact the query runtimes. To this end, we calculated Spearman's correlation of the said query features with the query runtime as shown in Table 5. Note that this table presents combined results obtained from Virtuoso and FUSEKI triplestores. We choose two triplestores as the query planner of the triplestore can greatly affect the query runtimes and hence biased the result towards a particular query planner. The overall results show that the number of projection variables (correlation 0.32) has the highest impact and

	Rank	1	2	3	4	5	6	7	8	9	10
	Feature	PV	JV	TP	Result	JVD	JTPS	TPS	BGPs	LSQ	BTPS
Synthetic	Bowlogna	0.25	0.46	0.25	0.64	0.01	0.32	0.20	NaN	0.09	0.08
	TrainB.	0.76	0.48	0.46	0.85	0.61	0.03	0.67	−0.21	−0.09	−0.66
	BSBM	−0.28	0.17	−0.04	−0.36	−0.11	−0.24	0.21	−0.11	0.38	−0.16
	SP2Bench	0.59	0.38	0.49	0.84	0.34	−0.27	−0.06	0.24	0.24	0.32
	WatDiv	0.00	−0.09	0.05	0.65	0.23	−0.02	0.03	NaN	NaN	0.46
	SNB-BI	0.11	0.41	0.46	0.05	0.23	0.08	−0.20	0.31	−0.16	−0.20
	SNB-INT	−0.10	0.33	0.36	0.07	0.02	−0.08	0.05	0.27	0.20	−0.33
Real	FEASIBLE	0.25	0.23	0.27	0.20	0.33	0.39	0.16	−0.14	−0.12	0.32
	Fishmark	0.69	0.80	0.69	0.56	0.43	0.18	0.12	−0.13	−0.12	0.35
	DBPSB	0.23	0.45	0.42	0.16	0.02	0.11	0.10	0.52	0.31	−0.05
	BioBench	0.27	0.36	0.31	0.46	0.19	0.01	0.14	−0.03	0.12	0.15
Datasets	DBpedia	0.22	−0.04	0.10	0.07	−0.07	0.20	0.00	0.18	0.03	0.01
	SWDF	0.65	0.67	0.74	−0.17	0.67	−0.16	−0.18	0.76	0.45	−0.47
	NCBIGene	0.46	−0.49	−0.50	0.66	−0.49	0.37	0.29	−0.19	−0.13	0.66
	SIDER	0.19	−0.03	−0.03	0.02	−0.03	0.39	0.64	−0.01	−0.05	0.17
	DrugBank	0.81	0.88	0.86	−0.81	0.86	0.30	−0.50	0.06	−0.15	−0.2
	Overall	0.32	0.31	0.30	0.24	0.20	0.11	0.10	0.09	0.06	0.00

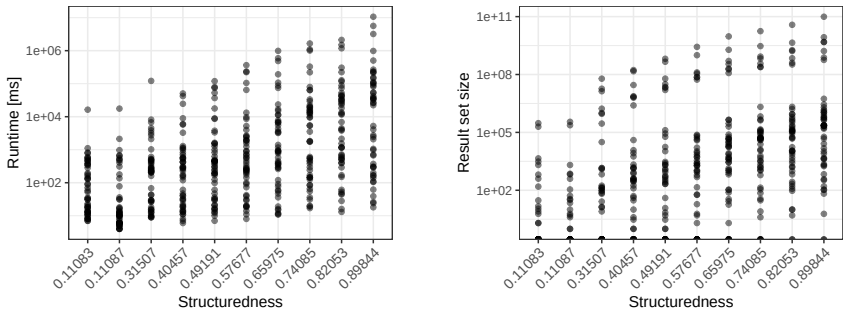
Table 5: Spearman's rank correlation coefficients between query features and query runtimes. PV: Projection Variables, JV: Join Vertices, TP: Triple Patterns, JVD: Join Vertex Degree, JTPS: Join-Restricted Triple Pattern Selectivity, TPS: Triple Pattern Selectivity, BTPS: BGP-Restricted TPS. Correlations and colors (−+): 0.00...0.19 very weak (●●), 0.20...0.39 weak (●●), 0.40...0.59 moderate (●●), 0.60...0.79 strong (●●), 0.80...1.00 very strong (●●).

BGP-restricted triple pattern selectivity (correlation 0.00) has the lowest impact on query runtimes. Yet, the result suggests that there is no single query feature that has a strong or very strong correlation with the query runtimes. This further suggests that the overall query runtime is impacted by a combination of different features.

5 RELATED WORK

Graph structure vs. query performance. The correlation between query runtime and workload metrics were studied in [17]. In addition to standard metrics, the authors introduced composite metrics such as the *absolute difficulty* (logarithm of the search space size), and the *relative difficulty*, which expresses how much worse a query engine does than the theoretical lower bound required by a certain query. The authors generated 12 graphs with different degree distributions along with 25+ queries of different shapes and measured their execution times. Then, they calculated the Kendall's τ rank correlation coefficient for $p < 0.001$ between each metric and the query execution times. The strongest correlation (+0.38) was exhibited by the *absolute difficulty* metric. The goal of gMark [4] is to define a *schema-driven workload generator* that synthesizes graph instances and queries for a given schema. The approach relies on controlling the diversity of the generated graphs and the difficulty of the generated workloads, using a *selectivity estimation* algorithm, which guarantees the selectivity of (certain) generated queries. The flexibility of their approach is demonstrated by generating workloads based on existing RDF benchmarks (SNB, SP2Bench, WatDiv).

Characterization of typed graphs. While this paper focuses on determining the correlation query execution time and graph metrics such as *relationship specialty* and *structuredness* (Sec. 2.1.1), other metrics could also provide valuable insight. In particular, the tools of



(a) Structuredness vs. runtime

(b) Structuredness vs. result set size

Figure 4: Correlation of structuredness value with runtimes and result set sizes. Note that the structuredness values are slighly off from the configured $[0.1, 0.2, \ldots, 1.0]$ values as the WatDiv generator can only target an approximate structuredness value.

DS	SF	Structuredness	Total Triples
1	0.1	0.110,878	103,394,017
2	0.2	0.110,835	103,395,364
3	0.3	0.315,073	105,813,355
4	0.4	0.404,579	99,583,093
5	0.5	0.491,919	98,636,578
6	0.6	0.576,772	102,975,692
7	0.6	0.659,752	102,463,145
8	0.7	0.740,858	107,058,360
9	0.8	0.820,532	98,341,615
10	0.9	0.898,443	108,962,809

Table 4: WatDiv datasets with varying structuredness values. *DS:* Data Set, *SF:* the Structuredness Factor that controls the structuredness values of the entities used in the datasets.

network science are often used to uncover the structural interplay between the nodes of a certain graph [9], which in turn could be used to predict the difficulty of querying such graphs. However, most works in the field only target the characterization of homogeneous, untyped networks, which omit a great deal of valuable information when trying to understand the structure of typed networks such as RDF graphs. Only recent research targeted the understanding of typed graphs, referred to as "multilayer", "multiplex", or "multidimensional" networks. The authors of [7] generalized the degree distribution to take edge types into account and introduced a set of typed connectivity metrics. Extending this work, the authors of [19] defined a set of additional metrics to characterize the effect of types on nodes and edge pairs. Meanwhile, paper [6] introduced variants of the local clustering coefficient that described the ratio of typed triangles in the network. Survey [1] summarizes the state-of-the-art on analyzing multilayer (edge-typed) networks. Typed graph metrics were used successfully in the context of *model-driven engineering* to describe the structure of system models and distinguish real graph models from synthetic ones [29, 32]. However, these were limited to graphs containing at most 1M nodes.

In the fields of *semantic web* and *database engineering*, a set of simple typed graph metrics were proposed in the context for *social network analysis* in [13]. The concept of "meta-path", a path with a given sequence of node/edge types and its related literature were investigated in survey [28]. Duan et al. [12] presented the structuredness values of several real-world datasets and synthetic benchmarks. RBench [20] introduced the relationship specialty metric and compared real datasets with datasets synthesized by data generators. WatDiv [2] introduced the Join-restricted and BGP-restricted triple pattern selectivities as important query features.

Our work. In this paper, we conducted a systematic survey to collect a current list of triplestore benchmarks. We collected important query and dataset features from state-of-the-art [2, 12, 20, 24] and added additional important query features such as the number of projection variables, the number of BGPs, the number of LSQ features, etc. We compared 11 triplestore benchmarks and 5 real-world datasets across the identified important data and query features. We also measured the correlation of the identified query features with the overall query runtime. To the best of our knowledge, there exists no such detailed analysis of triplestore benchmarks.

6 CONCLUSION AND FUTURE WORK

We performed a comprehensive analysis of existing benchmarks by studying synthetic and real-world datasets as well as by employing SPARQL queries with multiple variations. Our evaluation results suggest the following: (1) The dataset structuredness problem is well covered in recent synthetic data generators (e.g., WatDiv, TrainBench). The low relationship specialty problem in synthetic datasets still exists in general and needs to be covered in future synthetic benchmark generation approaches; (2) The FEASIBLE framework employed on DBpedia generated the most diverse benchmark in our evaluation; (3) The SPARQL query features we selected have a weak correlation with query execution time, suggesting that the query runtime is a complex measure affected by multidimensional SPARQL query features. Still, the number of projection variables, join vertices, triple patterns, the result sizes, and the join vertex degree are the top five SPARQL features that most impact the overall query execution time; (4) Synthetic benchmarks often fail to contain important SPARQL clauses such as `DISTINCT`, `FILTER`, `OPTIONAL`, `LIMIT` and `UNION`; (5) The dataset structuredness has a direct correlation with the result sizes and execution times of queries and indirect correlation with dataset specialty. As future work, we endeavour to broaden the scope of our analysis by adding more types of SPARQL query benchmarks (e.g., using reasoning, querying streams) and investigating more typed graph metrics on benchmark data sets.

ACKNOWLEDGMENTS

This work has been supported by the project HOBBIT (GA no. 688227), LIMBO (no. 19F2029I) and OPAL (no. 19F2028A) as well as by Science Foundation Ireland (SFI) under Grant No. SFI/12/RC/2289, MTA-BME Lendület Cyber-Physical Systems Research Group, and the ÚNKP-18-3-III New National Excellence Program of the Ministry of Human Capacities, Hungary. The authors would like to thank János Benjamin Antal for his assistance in reworking and benchmarking the LDBC SNB queries.

REFERENCES

[1] Alberto Aleta and Yamir Moreno. 2019. Multilayer Networks in a Nutshell. *Annual Review of Condensed Matter Physics* 10, 1 (2019). https://doi.org/10.1146/annurev-conmatphys-031218-013259

[2] Günes Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified Stress Testing of RDF Data Management Systems. In *ISWC*. 197–212. https://doi.org/10.1007/978-3-319-11964-9_13

[3] Marcelo Arenas, Claudio Gutiérrez, and Jorge Pérez. 2009. On the Semantics of SPARQL. In *Semantic Web Information Management - A Model-Based Perspective*. Springer, 281–307. https://doi.org/10.1007/978-3-642-04329-1_13

[4] Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George H. L. Fletcher, Aurélien Lemay, and Nicky Advokaat. 2017. gMark: Schema-Driven Generation of Graphs and Queries. *IEEE Trans. Knowl. Data Eng.* 29, 4 (2017), 856–869. https://doi.org/10.1109/TKDE.2016.2633993

[5] Samantha Bail, Sandra Alkiviadous, Bijan Parsia, David Workman, Mark van Harmelen, Rafael S. Gonçalves, and Cristina Garilao. 2012. FishMark: A Linked Data Application Benchmark. In *Proceedings of the Joint Workshop on Scalable and High-Performance Semantic Web Systems*. 1–15. http://ceur-ws.org/Vol-943/SSWS_HPCSW2012_paper1.pdf

[6] Federico Battiston, Vincenzo Nicosia, and Vito Latora. 2014. Structural measures for multiplex networks. *Phys. Rev. E* 89 (Mar 2014), 032804. Issue 3. https://doi.org/10.1103/PhysRevE.89.032804

[7] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2013. Multidimensional networks: Foundations of structural analysis. *World Wide Web* 16, 5-6 (2013), 567–593. https://doi.org/10.1007/s11280-012-0190-4

[8] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. *Int. J. Semantic Web Inf. Syst.* 5, 2 (2009), 1–24. https://doi.org/10.4018/jswis.2009040101

[9] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. 2006. Complex networks: Structure and dynamics. *Physics Reports* 424, 4 (2006), 175 – 308. https://doi.org/10.1016/j.physrep.2005.10.009

[10] Felix Conrads, Jens Lehmann, Muhammad Saleem, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. 2017. IGUANA: A Generic Framework for Benchmarking the Read-Write Performance of Triple Stores. In *ISWC*. Springer, 48–65. https://doi.org/10.1007/978-3-319-68204-4_5

[11] Gianluca Demartini, Iliya Enchev, Marcin Wylot, Joël Gapany, and Philippe Cudré-Mauroux. 2011. BowlognaBench - Benchmarking RDF Analytics. In *Data-Driven Process Discovery and Analysis SIMPDA*. Springer, 82–102. https://doi.org/10.1007/978-3-642-34044-4_5

[12] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. 2011. Apples and oranges: A comparison of RDF benchmarks and real RDF datasets. In *SIGMOD*. ACM, 145–156. https://doi.org/10.1145/1989323.1989340

[13] Guillaume Erétéo, Michel Buffa, Fabien Gandon, and Olivier Corby. 2009. Analysis of a Real Online Social Network Using Semantic Web Frameworks. In *ISWC*. Springer, 180–195. https://doi.org/10.1007/978-3-642-04930-9_12

[14] Orri Erling, Alex Averbuch, Josep-Lluis Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat-Pérez, Minh-Duc Pham, and Peter A. Boncz. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *SIGMOD*. ACM, 619–630. https://doi.org/10.1145/2723372.2742786

[15] Olaf Görlitz, Matthias Thimm, and Steffen Staab. 2012. SPLODGE: Systematic Generation of SPARQL Benchmark Queries for Linked Open Data. In *ISWC*. Springer, 116–132. https://doi.org/10.1007/978-3-642-35176-1_8

[16] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.* 3, 2-3 (2005), 158–182. https://doi.org/10.1016/j.websem.2005.06.005

[17] Benedek Izsó, Zoltán Szatmári, Gábor Bergmann, Ákos Horváth, and István Ráth. 2013. Towards precise metrics for predicting graph query performance. In *ASE*. 421–431. https://doi.org/10.1109/ASE.2013.6693100

[18] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2011. DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. In *ISWC*. Springer, 454–469. https://doi.org/10.1007/978-3-642-25073-6_29

[19] Vincenzo Nicosia and Vito Latora. 2015. Measuring and modeling correlations in multiplex networks. *Phys. Rev. E* 92 (Sep 2015), 032805. Issue 3. https://doi.org/10.1103/PhysRevE.92.032805

[20] Shi Qiao and Z. Meral Özsoyoglu. 2015. RBench: Application-Specific RDF Benchmarking. In *SIGMOD*. ACM, 1825–1838. https://doi.org/10.1145/2723372.2746479

[21] Shi Qiao and Z. Meral Özsoyoğlu. 2015. One Size Does not Fit All: When to Use Signature-based Pruning to Improve Template Matching for RDF graphs. *arXiv preprint arXiv:1501.07184* (2015). arXiv:1501.07184 http://arxiv.org/abs/1501.07184

[22] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. LSQ: The Linked SPARQL Queries Dataset. In *ISWC*. Springer, 261–269. https://doi.org/10.1007/978-3-319-25010-6_15

[23] Muhammad Saleem, Ali Hasnain, and Axel-Cyrille Ngonga Ngomo. 2018. LargeRDFBench: A billion triples benchmark for SPARQL endpoint federation. *J. Web Sem.* 48 (2018), 85–125. https://doi.org/10.1016/j.websem.2017.12.005

[24] Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. FEASIBLE: A Feature-Based SPARQL Benchmark Generation Framework. In *ISWC*. Springer, 52–69. https://doi.org/10.1007/978-3-319-25007-6_4

[25] Muhammad Saleem, Alexander Potocki, Tommaso Soru, Olaf Hartig, and Axel-Cyrille Ngonga Ngomo. 2018. CostFed: Cost-Based Query Optimization for SPARQL Endpoint Federation. In *SEMANTICS (Procedia Computer Science)*, Vol. 137. Elsevier, 163–174. https://doi.org/10.1016/j.procs.2018.09.016

[26] Muhammad Saleem, Claus Stadler, Qaiser Mehmood, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. 2017. SQCFramework: SPARQL Query Containment Benchmark Generation Framework. In *K-CAP*. 28:1–28:8. https://doi.org/10.1145/3148011.3148017

[27] Michael Schmidt et al. 2009. SP2Bench: A SPARQL Performance Benchmark. In *Semantic Web Information Management - A Model-Based Perspective*. 371–393. https://doi.org/10.1007/978-3-642-04329-1_16

[28] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *IEEE Trans. Knowl. Data Eng.* 29, 1 (2017), 17–37. https://doi.org/10.1109/TKDE.2016.2598561

[29] Gábor Szárnyas et al. 2016. Towards the characterization of realistic models: Evaluation of multidisciplinary graph metrics. In *MODELS*. 87–94. http://dl.acm.org/citation.cfm?id=2976786

[30] Gábor Szárnyas, Benedek Izsó, István Ráth, and Dániel Varró. 2018. The Train Benchmark: Cross-technology performance evaluation of continuous model queries. *Softw. Syst. Model.* 17, 4 (2018), 1365–1393. https://doi.org/10.1007/s10270-016-0571-8

[31] Gábor Szárnyas, Arnau Prat-Pérez, Alex Averbuch, József Marton, Marcus Paradies, Moritz Kaufmann, Orri Erling, Peter A. Boncz, Vlad Haprian, and János Benjamin Antal. 2018. An early look at the LDBC Social Network Benchmark's Business Intelligence workload. In *GRADES-NDA at SIGMOD*. ACM, 9:1–9:11. https://doi.org/10.1145/3210259.3210268

[32] Dániel Varró, Oszkár Semeráth, Gábor Szárnyas, and Ákos Horváth. 2018. Towards the Automated Generation of Consistent, Diverse, Scalable and Realistic Graph Models. In *Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig*. Springer, 285–312. https://doi.org/10.1007/978-3-319-75396-6_16

[33] Hongyan Wu et al. 2014. BioBenchmark Toyama 2012: An evaluation of the performance of triple stores on biological data. *J. Biomedical Semantics* 5 (2014), 32. https://doi.org/10.1186/2041-1480-5-32