SCHEMA-LEVEL INDEX MODELS FOR WEB DATA SEARCH

ANSGAR SCHERP

Data Science and Big Data Analytics
Ulm University, Ulm, Germany
ansgar.scherp@uni-ulm.de

TILL BLUME

GSA R&D
Ernst & Young GmbH WPG, Berlin, Germany
Till.Blume@de.ey.com

Indexing the Web of Data offers many opportunities, in particular, to find and explore data sources. One major design decision when indexing the Web of Data is to find a suitable index model, i.e., how to index and summarize data. Various efforts have been conducted to develop specific index models for a given task. With each index model designed, implemented, and evaluated independently, it remains difficult to judge whether an approach generalizes well to another task, set of queries, or dataset. In this work, we empirically evaluate six representative index models with unique feature combinations. Among them is a new index model incorporating inferencing over RDFS and `owl:sameAs`. We implement all index models for the first time into a single, stream-based framework. We evaluate variations of the index models considering sub-graphs of size 0, 1, and 2 hops on two large, real-world datasets. We evaluate the quality of the indices regarding the compression ratio, summarization ratio, and F1-score denoting the approximation quality of the stream-based index computation. The experiments reveal huge variations in compression ratio, summarization ratio, and approximation quality for different index models, queries, and datasets. However, we observe meaningful correlations in the results that help to determine the right index model for a given task, type of query, and dataset.

Keywords: Graph Summarization; Schema-level Graph Indices; Data Search

1. Introduction

Graph indices are well-established to efficiently manage large heterogeneous graphs like the Web of Data. In general, one can distinguish instance-level indices and schema-level indices for the Web of Data. Instance-level indices focus on finding specific data instances [1, 2, 3], e. g., searching for a specific book by its title such as "Towards a clean air policy". In contrast, schema-level indices (short: *SLI*) support structural queries, e. g., searching for data instances with the property *dct:creator* and RDF type *bibo:book* [4]. An *SLI model* defines how and which combinations of types and properties are indexed, i. e., how data instances are summarized and which queries are supported by the index.

In the past, various SLI models have been developed for different tasks such as data exploration [5, 6, 7, 8], query size estimation [9], vocabulary terms recommendation [10], related entity retrieval [11], data search [4], and others. The task of data search is to find (sub-)graphs on the Web that match a given schema structure. Given a structural query as

shown in Listing 1, an SLI can return URIs of data sources $?ds$ with data instances containing bibliographic metadata.

Listing 1: Structural query to find datasources ?ds containing information about books that have a creator that is an agent.

```
SELECT ?ds WHERE {
    ?ds rdf:type bibo:book ;
        dct:creator foaf:Agent .
}
```

This use of SLIs for data search on the Web of Data is illustrated in Figure 1. In a first step, an SLI is queried to identify relevant data sources, which then in a second step are accessed via HTTP get requests to download the actual data instances. Search systems like LODatio [4], LODeX [5], Loupe [6], and LODatlas [7] rely on SLI to offer a search for relevant data sources or exploration of data sources.
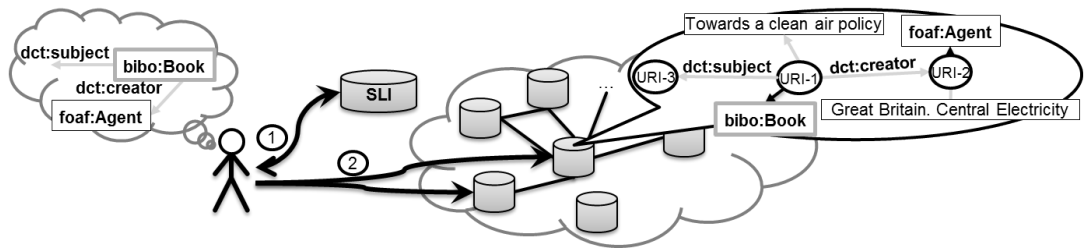

Fig. 1. Finding data sources on the Web using a schema-level index (SLI). A structural query is executed over an SLI to identify relevant data sources (1). Subsequently, the data sources are accessed to retrieve actual data instances (2).

The problem is that all SLI models were designed, implemented, and evaluated for their individual task only, using different queries, datasets, and metrics. Our hypothesis is that there is no SLI model that fits all tasks and that the performance of the specific SLI depends on the specific types of queries and characteristics of the datasets. However, so far only very limited work has been done on understanding the behavior of SLI models in different contexts. With each SLI model evaluated independently in a specific context, it remains difficult to judge whether an approach generalizes well to another task or not. In other words, it is not known which SLI model can be used for which contexts, tasks, and datasets.

This paper presents a systematic comparison of SLI models and shows how different SLI models, queries, and datasets influence the results of querying on the Web of Data [12]. We conduct an extensive empirical evaluation of representative SLI models. To this end, we have for the first time defined and implemented the features of existing SLI models in a common framework available on GitHub. Based on the discussion of related works, we chose six SLI models with unique feature combinations, which were developed for different tasks, to understand and compare their behavior for the data search task. We empirically investigate the behavior of each selected SLI model in three variants, where we index sub-graphs of 0, 1, and 2 hop lengths.

The empirical evaluation consists of two sets of experiments. In a first set of experiments, we analyze the relative size of the computed SLI compared to the original dataset (compression ratio) and the number of schema elements in the SLI compared to the number of data instances in the dataset (summarization ratio). The second set of experiments quantifies the quality of a stream-based computation of the SLI for large datasets obtained from the Web of Data. The stream-based approach is designed to scale to graphs of arbitrary sizes by observing the graph over a stream of edges with fixed window size. Inherently, this approach introduces inaccuracies in the SLI computation by potentially extracting incomplete schema structures due to limited window size [13]. Our experiments show huge variations in compression ratio, summarization ratio, and approximation quality for the SLI models. However, we also observe strong positive and negative correlations between the three metrics. These insights shed light on the behaviors of SLI models for different datasets and queries that help to determine the right SLI model for a given task and dataset.

The remainder of this paper is structured as follows. Subsequently, we discuss the features of the schema-level index (SLI) models reported in the literature. We formalize how data instances are summarized by SLIs in Section 3. In Section 4, we introduce the experimental apparatus, i. e., our framework and the datasets. The results of our experiments are presented in Section 5. Finally, we discuss our main findings, before we conclude.

2. Related Work: Overview of Schema-level Index Models and their Features

Our discussion of related works focuses on different models for schema-level indices (SLI). As introduced above, SLIs support the execution of structural queries over the Web of Data. Structural queries such as the example in Listing 1 consist of a combination of types and properties.

In the past, various SLI models were defined, which capture different schema structures and are defined using different theoretical approaches [14]. Table 1 provides an overview of SLI models reported in the literature and the specific features they support. These features are the use of property sets [9, 11, 13, 8, 5, 6, 10, 15], use of type sets [13, 8, 5, 6, 10, 15], use of neighbor information [11, 13, 8, 5, 6, 10, 15], use of path information [9, 11, 13, 8, 5, 6, 15], use of k-bisimulation [15], use of incoming property sets [9, 15], use of OR combination of feature [15], use of transitively co-occurring (related) properties [15], and use of inferred information from RDF Schema properties [8, 15]. In addition, we propose inferencing over `owl:sameAs` as a new feature. Below, we discuss the SLI models presented in Table 1 from top to bottom. We present their task, schema structure, and feature combination.

Characteristic Sets [9], illustrated in Figure 2a, was developed to optimize cardinality estimations for join-queries in RDF databases. Formally, the SLI model is defined as sets of data instances using a first-order-logic expression over triples. Characteristic Sets summarize data instances along common incoming properties (the **incoming property set**) and outgoing properties (the **property set**). Related to Characteristic Sets is the SLI model SemSets [11], illustrated in Figure 2b, which was developed to discover semantically similar sets of data instances in knowledge graphs to improve keyword-based ad-hoc retrieval. SemSets is defined using set operators. The SemSets model summarizes data instances that share the same outgoing properties, which are linked to a common resource. Thus, the model uses property sets and include **neighbor information**.

Table 1. Nine index models (left column) and their features (top row). Features marked with X are fully supported, (X) are partially supported, and - are not supported.

Index Model / Feature	Property sets	Type sets	Neighbor information	Path information	k-bisimulation	Incoming property set	OR combination	Related properties	RDF Schema	SameAs
Characteristic Sets [9]	X	-	-	X	-	X	-	-	-	-
SemSets [11]	X	-	X	X	-	-	-	-	-	-
Weak Property Clique [15]	X	-	-	X	X	X	X	X	X	-
ABSTAT [8]	X	X	X	X	-	-	-	-	(X)	-
LODex [5]	X	X	X	X	-	-	-	-	-	-
Loupe [6]	X	X	X	X	-	-	-	-	-	-
SchemEX [13]	X	X	X	X	-	-	-	-	-	-
TermPicker [10]	X	X	X	-	-	-	-	-	-	-
SchemEX+U+I	X	X	X	X	-	-	-	-	X	X

Goasdoué et al. [15] introduce compact graph summaries for RDF graphs. They propose so-called Property Cliques formalized as equivalence relations, which can capture different schema structures. Property Cliques summarize data instances based on the co-occurrence of **related properties**. In their work, two properties are related, if they co-occur in a data instance. Thus, also pointed out by the authors, the relationship of being related is transitive [15]. Furthermore, Goasdoué et al.use property sets and incoming property sets. They propose so-called *strong equivalence* relations, where the related incoming *and* outgoing properties are the same, and so-called *weak equivalence* relations, where the incoming *or* the outgoing properties are the same (the **OR combination**). They also support **RDF Schema** inferencing to compute their summaries. The weak equivalence version of the Property Cliques is the only index model that uses transitively co-occurring (related) properties and summarizes instances based on the same property sets, or the same incoming property sets, or both (OR combination). In the following, we call this variant of the compact graph summaries *Weak Property Clique* (illustrated in Figure 2c).
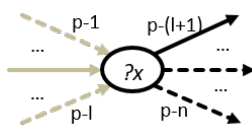
ABSTAT [8], LODeX [5], Loupe [6], and SchemEX [13] summarize data instances based on a common set of RDF types (the **type sets**) and properties linking to resources with the same type set (illustrated in Figure 2d). Thus, they use neighbor information in combination with type sets. ABSTAT, LODeX, and Loupe were developed to explore datasets. ABSTAT's schema structure has no formal definition and is only informally defined in a textual description. In terms of the SLI model features, ABSTAT [8] can use inferencing over the RDFS type hierarchies to compute so-called minimal patterns. A minimal pattern is the set of most specific RDF types associated with the instances. All super-types of these most specific types are removed. This feature is also supported by Goasdoué et al. [15], who additionally take
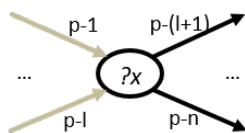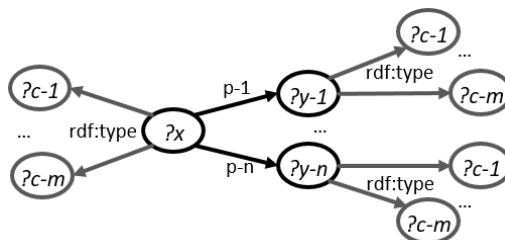
(a) Characteristic Sets [9] summarize data instances based on a common set of incoming and outgoing properties.
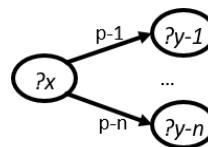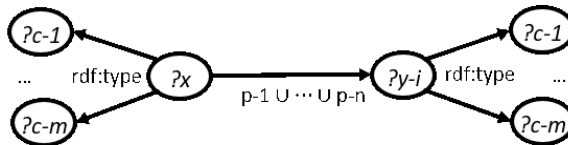

(b) SemSets [11] summarize data instances based on a common set of properties linked to the same resources.

(c) Weak Property Cliques [15] summarize data instances based on at least one related incoming property and/or at least one related outgoing property.

(d) SchemEX [13], ABSTAT [8], LODeX [5], and Loupe [6] summarize data instances based on a common type set and a common set of properties linked to resources sharing the same type set.

(e) TermPicker [10] summarizes data instances based on a common type set, a common property set, and a common type set linked over all properties.

Fig. 2. Illustration of different schema structures captured by different schema-level index models reported in the literature.

sub-properties as well as RDFS domain and range into account. Similarly, the SLI model Loupe is also only informally specified, which—for some aspects—leaves room for interpretations. The Loupe model supports three so-called Class, Property, and Triple Inspector that allow different types of queries. The Class Inspector considers only type sets, the Property Inspector only property sets, and the Triple Inspector a combination of type sets and property sets. The Triple Inspector extracts triples of the form $< subjectType, predicate, objectType >$ from the input graph. Thus, the schema structure captured by Triple Inspector is equivalent to the SLI models ABSTAT and LODeX. In contrast to ABSTAT and Loupe, the SLI model of LODeX computes clusters over the RDF types and selects a representative RDF type per cluster. SchemEX was evaluated for the data search task on snapshots of the Web of Data using a stream-based schema extraction approach. SchemEX was defined as stratified 1-bisimulation [16].

TermPicker [10], illustrated in Figure 2e, was developed to make data-driven recommendations of vocabulary terms. TermPicker summarizes data instances based on a common type set, a common property set, and a common type set of all property-linked resources. Thus,

in contrast to, e. g., SchemEX, TermPicker does not take **path information** into account, i. e., it does not matter via which properties these data instances are connected. Therefore, in order to model TermPicker one needs to be able to aggregate the information of neighboring data instances.

Furthermore, the index model SchemEX+U+I (short for: SchemEX with **U**nions of sameAs instances and RDFS **I**nferencing) extents the common schema structure of ABSTAT, LODeX, Loupe, and SchemEX, by providing support for inferencing over `owl:sameAs` as well as using all RDFS information, i. e., types, properties, domain, and range.

Bisimulation is an approach to determine whether two graphs are equivalent by considering the traversal over a graph as operating on state transition systems and defines an equivalence relation over states [16]. Two states are equivalent (or bisimilar) if they change into equivalent states with the same type of transition. A stratified k-bisimulation reduces the path length to k hops as it is defined with $k = 1$ in SchemEX [13]. Tran et al. [17] model k-bisimulation as height parameterization that can be applied on SLI. Thus, we decided to use k-bisimulation as a feature rather than a separate index model. Tran et al. determine that the parameter is typically $k \in 0, 1, 2$.

In summary, from the discussion of the related works, we can state that there exists a variety of SLI models that capture different schema structures and are suitable for different tasks. SLI models are designed, implemented, and evaluated independently for their specific tasks.

3. Summarizing Data Instances with Schema-level Indices

We define the foundational concepts of data graph, data instances, and summarization of instances using schema-level indices (SLI). A **RDF data graph** G is defined as $G \subseteq V_U \cup V_B \times P \times (V_U \cup V_B \cup L)$, where V_U denotes the set of URIs, V_U the set of blank nodes, $P \subseteq V_U$ the set of properties, and L the set of literals. Furthermore, there exists a subset $V_C \subseteq V_U \cup V_B$ that contains all RDF classes. A triple is a statement about a resource $s \in V_U \cup V_B$ in the form of a subject-predicate-object expression $(s, p, o) \in G$.

We define a **data instance** $I_s \subseteq G$ to be a set of triples, where each triple shares a common subject URI s. We call the set of all $c \in V_C$ with $(s, rdf{:}type, c) \in I_s$ the type set of I_s. Furthermore, we call the set of all $p \in P$ with $(s, p, o) \in I_s$ the property set of I_s. In our example in Figure 3, we visualize the type sets of the instances as squares above the corresponding URI. The remaining triples are visualized as directed edge, e. g., $(s\text{-}1, p\text{-}1, o\text{-}1)$.

SLI summarize data instances I_s based on their common schema. The specific features used to compute the schema of a data instance are defined in the SLI model. For example, Characteristic Sets uses property sets and incoming property sets of each instance only, while most other index models also support to take RDF types into account. For a detailed discussion of the features in SLI models, please refer to Section 2. Data instances with the same schema can be uniquely identified by their common schema. This unique schema identifier, which summarizes the data instances, is called the **schema element**. Schema elements are stored as keys in the SLI and can be used, e. g., to retrieve all summarized data instances. In Figure 3, the schema-level index contains one schema element (SE-1).

For tasks such as searching in the Web of Data, one is interested in the data sources of the summarized instances (compare the motivating illustration in Figure 1). To model where
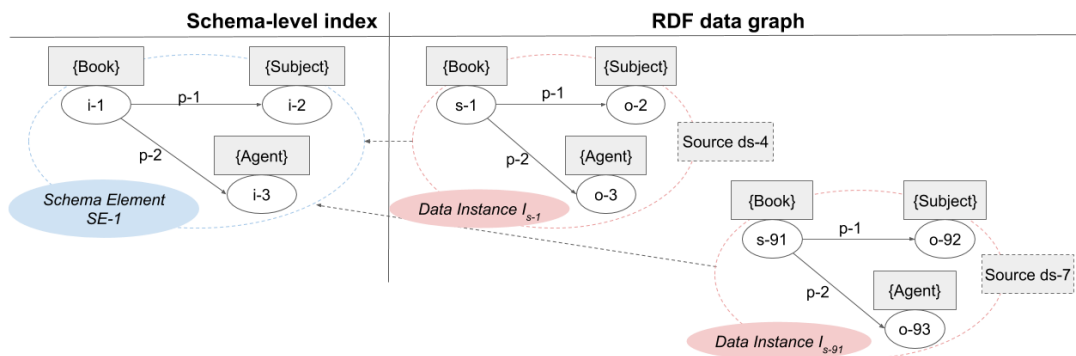
Fig. 3. Computing the SchemEX index for a decentralized RDF data graph. Both data instances I_{s-1} and I_{s-91} (right) are summarized by the same schema element SE-1 (left).

data instances have been found on the Web of Data, we define the data source D_s of an instance I_s as the set of URIs ds that identify documents on the Web that contain triple (s, p, o) about instance I_s. Common RDF Crawlers provide this information using the notion of quads (s, p, o, ds) [18]. For data search, this means that the SLI does not need to store URIs of the summarized data instances directly, but merely needs to store the information about the URIs to the data sources where these instances have been observed. For example, the data instances in Figure 3 can be found in ds-4 and ds-7. Thus, intuitively, a SLI for data search is a lookup table that contains information about schema structures of data instances on the Web and where they are located. For another task such as cardinality estimation one would lookup the number of summarized data instances instead of their location.

In the Web of Data exists the idea of inferencing implicit information, e. g., using RDFS or `owl:sameAs`. For example, the semantics of a triple $(p\text{-}1, \text{rdfs:subPropertyOf}, p\text{-}2) \in G$ states that for any instance I_s with $(s, p\text{-}1, o)$, we can infer the additional triple $(s, p\text{-}2, o)$. In the following, we assume that inferencing over RDF or `owl:sameAs` adds the respective types and properties to the corresponding type sets and property sets of each instance.

4. Experimental Study

We implement all features from the related work (see Section 2) into a single framework. Our framework allows flexibly combining these features to define all of the existing as well as new SLI models.

4.1. *Choice of Schema-level Index Models*

We select six representative index models based on our analysis in Section 2. These SLI models are Characteristic Sets, Weak Property Clique, SemSets, TermPicker, SchemEX, and SchemEX+U+I. We select Characteristic Sets, Weak Property Clique, SemSets, TermPicker since they provide unique feature combinations. Furthermore, we select SchemEX since the SLI model shares the same base schema structure with LODex, Loupe, and ABSTAT (see Table 1). Finally, we select SchemEX+U+I since it uses `owl:sameAs` and full RDFS reasoning. This covers also the RDFS type hierarchy inferencing of ABSTAT and the RDFS reasoning

provided by the compact graph summaries. For each of the six selected SLI models, we apply the k-height parameterization feature proposed by Tran et al. [17]. Reasonable values for the height parameterization k are 0, 1, and 2 [17]. Thus, in total, we compare 18 unique index models.

4.2. *Datasets*

We use two datasets crawled from the Web of Data. The first dataset is called *TimBL-11M*. It contains about 11 million triples crawls in a breadth-first search starting from a single seed URI, the FOAF profile of Tim Berners-Lee [13]. The TimBL-11M dataset is a graph with $673k$ data instances distributed over $18k$ data sources. Each instance has on average 13 outgoing properties (standard deviation (SD): 275), 3 incoming properties (SD: 848), 3 types (SD: 40), and is defined in 3 data sources (SD: 40). The latter means that, on overage, triples with a common subject URI can be found in 3 distinct data sources, which makes the challenge for data search specifically hard. Overall, $3,919$ unique properties and $2,738$ unique RDF types appear in the TimBL-11M dataset.

The second dataset *DyLDO-127M* contains $127M$ triples [19]. The Dynamic Linked Data Observatory (DyLDO) provides regular snapshots from the Web of Data. In contrast to TimBL-11M, which uses only a single seed URI, the DyLDO dataset is populated by crawling in breadth-first search about $95k$ representative seed URIs of the Web of Data [19]. The crawling of these seed URIs is restricted to a crawling depth of two hops. We use the first snapshot of the DyLDO dataset, as it is with $127M$ triples the larged one. The DyLDO-127M dataset contains $7M$ data instances distributed over $154k$ data sources. Each instance has on average 17 outgoing properties (SD: 6503), 7 incoming properties (SD: 635), a single type (SD: 17), and is defined in 2 data sources (SD: 17). Overall, $15k$ unique properties and $31k$ unique types appear in the DyLDO-127M dataset.

Both datasets are reasonably large with $11M$ triples and $127M$ triples, while still allowing to compute a gold standard for the stream-based index computation. A gold standard is created by loading the entire dataset into the main memory and computing the indices with no window size limit. No pre-processing was conducted on the datasets, except removing triples that do not follow the W3C standards.

5. Experiment 1: Compression and Summarization Ratio

5.1. *Description*

We evaluate the index size for the selected indices over the two datasets mentioned above. The size of an index refers to the number of triples when stored as an RDF graph. We compare the number of triples in the index to the number of triples in the dataset (**compression ratio**). Furthermore, we compare the number of schema elements in the index to the number of data instances in the dataset (**summarization ratio**). This ratio gives an idea of how well the defined schema structure can summarize data instances on the Web of Data. For the compression and summarization ratios, we use exact indices. This means we loaded the complete data graph into the main memory before we started the index computation process.

Table 2. Results from the analysis of the **compression ratio** and **summarization ratio** of the six selected SLI models (with height parameter $k \in \{0, 1, 2\}$). $\#t$ is the number of triples in millions (M) in the SLI and in brackets below the ratio compared to the number of triples in the dataset (**compression ratio**). $\#e$ is the number of schema elements in thousands (T) in the SLI and in brackets below the ratio compared to the number of instances in the dataset (**summarization ratio**). As datasets, we use the TimBL-11M (top) and DyLDO-127M datasets (bottom).

	Index Model	$k = 0$ $\#t$	$\#e$	$k = 1$ $\#t$	$\#e$	$k = 2$ $\#t$	$\#e$
TimBL-11M	**Character -istic Sets**	*na* (*na*)	*na* (*na*)	0.7M (6.5%)	9.6T (1.4%)	1.6M (14.6%)	37.2T (5.5%)
	Weak Prop -erty Clique	*na* (*na*)	*na* (*na*)	1.9M (17.9%)	74 (< 0.1%)	1.1M (9.9%)	50 (< 0.1%)
	SemSets	0.3M (2.9%)	2.8T (0.4%)	7.6M (69.2%)	139.0T (20.6%)	7.6M (69.2%)	139.0T (20.6%)
	SchemEX	0.3M (2.9%)	2.8T (0.4%)	0.8M (6.9%)	12.0T (1.8%)	1.4M (12.5%)	27.7T (4.1%)
	TermPicker	0.3M (2.9%)	2.8T (0.4%)	0.7M (6.5%)	10.8T (1.6%)	1.8M (16.0%)	37.3T (5.5%)
	SchemEX +U+I	0.4M (3.8%)	3.1T (0.5%)	0.8M (7.1%)	11.3T (1.7%)	1.8M (15.9%)	31.0T (4.6%)
DyLDO-127M	**Character -istic Sets**	*na* (*na*)	*na* (*na*)	0.6M (0.5%)	23.0T (0.3%)	2.1M (1.7%)	112.8T (1.6%)
	Weak Prop -erty Clique	*na* (*na*)	*na* (*na*)	14.8M (9.0%)	394 (< 0.1%)	25.1M (19.7%)	102 (< 0.1%)
	SemSets	4.1M (3.2%)	46.6T (0.7%)	45.3M (35.6%)	1733.5T (25.0%)	45.3M (35.6%)	1733.5T (25.0%)
	SchemEX	4.1M (3.2%)	46.6T (0.7%)	15.7M (12.3%)	254.5T (3.6%)	19.8M (15.6%)	431.1T (6.1%)
	TermPicker	4.1M (3.2%)	46.6T (0.7%)	11.1M (8.7%)	238.4T (3.4%)	25.4M (19.9%)	559.1T (7.9%)
	SchemEX +U+I	8.5M (6.7%)	53.0T (0.8%)	19.9M (15.7%)	249.5T (3.5%)	22.9M (18.0%)	466.9T (6.6%)

5.2. *Results*

The results of the experiments regarding the compression ratio and summarization ratio are documented in Table 2. As one can see, there is a huge variety in terms of how well indices compress and summarize the data. For the TimBL-11M dataset, SemSets' compression ratio (with $k = 1$) is about 10 times larger than all other indices except for Weak Property Cliques

(only about 5 times larger). For the DyLDO-127M dataset, SemSets' compression ratio (with $k = 1$) is up to 75 times larger. Additionally, there is no increase in index size from $k = 1$ to $k = 2$, but a more than ten-times increase from $k = 0$ to $k = 1$. A similar increase appears for the summarization ratio. SemSets is the only index that uses neighbor information but not neighbor type sets, i.e., they compare the object URIs o of each (s, p, o) triple. In contrast, the other indices either ignore objects or consider their type sets only. SemSets has a summarization ratio of 20% to 25%, i.e., on average 4 to 5 data instances share the same schema structure. The smallest index, Characteristic Sets, has a summarization ratio of 0.3%, i.e., about 330 data instances share the same schema structure. A notable exception is the Weak Property Clique, which shows the most condensed summarization (summarization ratio of less than 0.1%). However, the combination of either incoming or outgoing related properties in Weak Property Cliques leads to a considerably large compression ratio. Weak Property Clique indices are more than twice the size of Characteristic Sets indices.

When considering the semantics of RDFS and *owl:sameAs* in SchemEX+U+I, the index size increases compared to SchemEX by about 3% more triples. Despite being a larger index in terms of the number of triples, for $k = 1$ fewer schema elements are computed when including the semantics of RDFS and *owl:sameAs*. For $k = 0$ and $k = 2$, SchemEX+U+I requires more schema elements than SchemEX to summarize the data instances.

In summary, including the semantics of *owl:sameAs* and RDF Schema increases the size of the index. However, it can reduce the number of schema elements. Furthermore, using weak equivalences leads to a handful of schema elements with a considerably large size summarizing all data instances.

6. Experiment 2: Stream-based Index Computation

6.1. *Description*

In this experiment, we are interested in how well queries of varying complexity can be supported by the indices if the SLI is computed over a stream of graph edges. Motivated from stream-databases, the idea is to consider the triples in the datasets as a stream that is observed in windows of sizes $1k$, $100k$, and $200k$. This allows us to scale the computation to in principle arbitrary sized input graphs [13]. However, the approach produces approximation errors since only a fraction of the data graph is kept simultaneously in the main memory, while the remainder is not yet known or inaccessible. Thus, we potentially extract incomplete schema structures.

6.2. *Pre-processing vs. On-the-fly Processing in SchemEX+U+I*

Regarding the index model of SchemEX+U+I, we evaluate two variants in this experiment: The RDFS inferencing requires an additional data structure during the computation process, the so-called schema graph [20]. This schema graph is constructed from the triples using RDFS `range`, `domain`, `subClassOf`, or `subPropertyOf`. With the domain, range, and hierarchical types/properties information, we infer additional types and properties for the remaining data instances. In one version called SchemEX+U+oI, the RDFS information is extracted and inferred *on-the-fly*. Here, we construct the schema graph simultaneously to the index

computation. The advantage is that only one pass over the dataset is needed. However, since the schema graph is built while the index is computed information may be missing for the inferencing. In the other version called SchemEX+U+pI, we first extract all RDFS information in a *pre-processing* step to construct the schema graph. The advantage is that the inferencing of triples is conducted on the complete schema graph only. The drawback is that two passes over the dataset are needed.
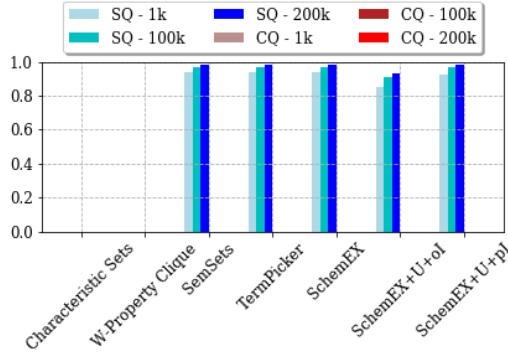
6.3. *Queries*

A central challenge for this experiment is the choice of queries to be executed over the indices. Here, we follow the work by Konrath et al. [13] who conducted a data-driven query generation for the evaluation of approximate graph indices. This means the queries are generated from the actual data instances in the datasets, i.e., their combination of types and properties. We distinguish two types of queries, simple queries (SQ) and complex queries (CQ). Simple queries search for data instances that have a common type set (or in the case of SemSets a common set of objects). Analyses of existing query logs show that most SPARQL queries in search systems are simple queries [21]. In contrast, complex queries search for data instances that match the complete schema structure defined by the specific index model, e.g., include property paths over 2 hops for Characteristic Sets with $k = 2$.

6.4. *Measures*

We execute the simple and complex queries on the SLI computed with fixed window size and on the gold standard SLI. For our data search task, the results of the queries are the two sets D_{gold} and D_{window}, which contain the corresponding data source URIs. Following Konrath et al. [13], the approximation quality is measured by comparing D_{gold} and D_{window} using the F1-score.

6.5. *Results*

Figure 4 shows the approximation quality in terms of F1-score for the selected index models. Table 3 shows the results as a table. For indices with a height parameter $k = 0$, the simple queries and the complex queries are alike. Moreover, Characteristic Sets and Weak Property Cliques do not use type information (or object information). Thus, simple queries are not available for these index models.
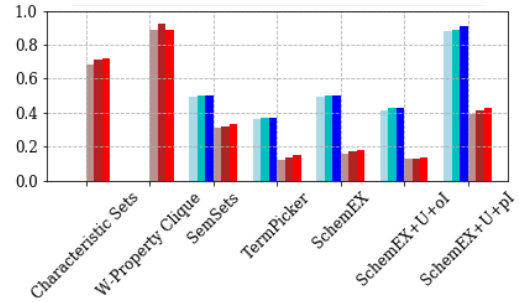
(a) F1-scores for TimBL-11M and $k = 0$.
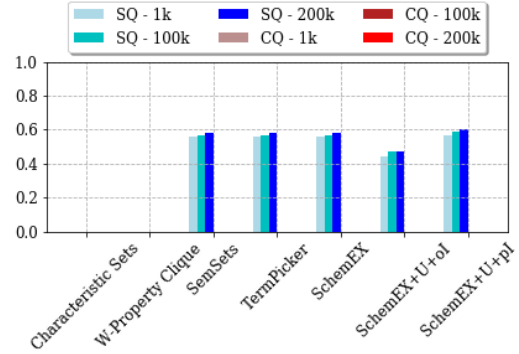
(b) F1-scores for DyLDO-127M and $k = 0$.
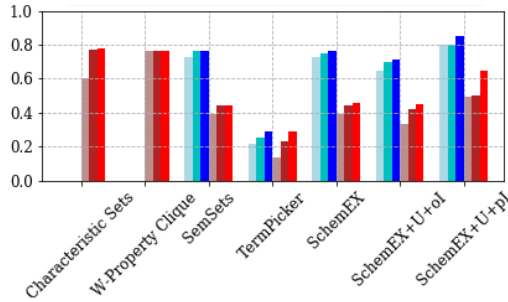
(c) F1-scores for TimBL-11M and $k = 1$.

(d) F1-scores for DyLDO-127M and $k = 1$.
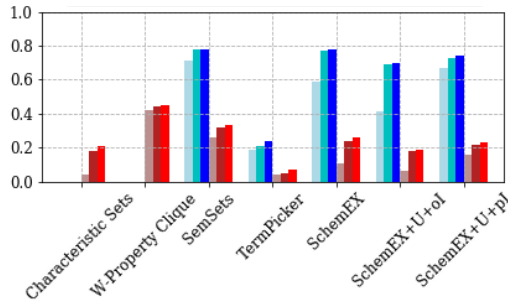
(e) F1-scores for TimBL-11M and $k = 2$.
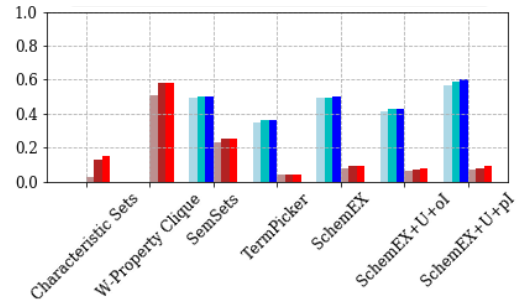
(f) F1-scores for DyLDO-127M and $k = 2$.

Fig. 4. F1-score for simple queries (SQ) and complex queries (CQ) and for window sizes (1k, 100k, 200k). The left column shows the values for the TimBL-11M dataset and the right column the DyLDO-127M dataset, respectively. The influence of the height parameter $k \in \{0, 1, 2\}$ can be seen in the rows from top to bottom.

Table 3. F1-score or not available (na) for simple queries (SQ) and complex queries (CQ), the height parameter $k \in \{0, 1, 2\}$, and for different window sizes (1k, 100k, 200k) on the two datasets TimBL-11M and DyLDO-127M.

	k	Q	Character-istic Sets			Weak Property Clique			SemSets			SchemEX			TermPicker			SchemEX +U+oI			SchemEX +U+pI		
			1k	100k	200k	1k	100k	200k	1k	100k	200k	1k	100k	200k	1k	100k	200k	1k	100k	200k	1k	100k	200k
TimBL-11M	0	SQ	na	na	na	na	na	na	.94	.97	.98	.94	.97	.98	.94	.97	.98	.85	.91	.93	.92	.97	.98
	1	SQ	na	na	na	na	na	na	.73	.76	.76	.73	.75	.76	.22	.25	.29	.65	.70	.71	.80	.80	.85
	1	CQ	.60	.77	.78	.76	.76	.76	.39	.44	.44	.39	.44	.46	.14	.23	.29	.33	.42	.45	.49	.50	.65
	2	SQ	na	na	na	na	na	na	.71	.78	.78	.59	.77	.78	.19	.21	.24	.41	.69	.70	.67	.73	.74
	2	CQ	.04	.18	.21	.42	.44	.45	.26	.32	.33	.11	.24	.26	.04	.05	.07	.06	.18	.19	.16	.22	.23
DyLDO-127M	0	SQ	na	na	na	na	na	na	.56	.57	.58	.56	.57	.58	.56	.57	.58	.44	.47	.47	.57	.59	.60
	1	SQ	na	na	na	na	na	na	.49	.50	.50	.49	.49	.50	.36	.37	.37	.41	.43	.43	.88	.89	.91
	1	CQ	.68	.71	.72	.89	.92	.89	.31	.32	.33	.16	.17	.18	.12	.14	.15	.13	.13	.14	.39	.41	.43
	2	SQ	na	na	na	na	na	na	.49	.50	.50	.49	.49	.50	.35	.36	.36	.41	.43	.43	.57	.59	.60
	2	CQ	.03	.13	.15	.51	.58	.58	.23	.25	.25	.08	.09	.09	.04	.04	.04	.06	.07	.08	.07	.08	.09

From the results of our experiment, we can state that simple queries consistently show higher F1-scores than complex queries. TermPicker and Weak Property Cliques are the only indices that have a higher F1-score on the DyLDO-127M dataset than on the TimBl-11M dataset. As described in Section 2, TermPicker is the only index not using the path information feature. This restriction is the only difference in the schema structure compared to SchemEX. Still, TermPicker has a 50% lower F1-score than SchemEX on the TimBL-11M dataset.

Regarding the complex queries, the F1-scores of Weak Property Cliques are the highest in the experiments with $k = 1$ and 2. For $k = 2$, Weak Property Cliques have between .12 and .54 higher F1-scores compared to the other indices. SemSets only have a small drop in F1-score from $k = 1$ to $k = 2$. SchemEX+U+oI consistently has a lower F1-score than SchemEX. For SchemEX+U+pI, we extracted the RDF Schema information in a pre-processing step. Compared to SchemEX+U+oI, SchemEX+U+pI has consistently higher F1-scores. Furthermore, for window sizes $100k$ and $200k$, SchemEX+U+pI has higher F1-scores than SchemEX (except for $k = 2$ for TimBL-11M).

We also observe an influence of the characteristics of the crawled dataset on the approximation quality. All indices have on average a .15 lower F1-score on the DyLDO-127M dataset compared to the TimBL-11M dataset. In particular, simple queries achieve much lower F1-scores. On average, simple queries have .25 lower F1-scores and complex queries have .04 lower F1-scores on the DyLDO-127M dataset compared to the TimBL-11M dataset. Furthermore, larger window sizes consistently improve F1-scores. In contrast, on-the-fly inferencing lowered the F1-scores in our experiment compared to no inferencing.

7. Discussion

Key insights from our experiments are: (1) SLI models perform very differently in terms of compression ratio, summarization ratio, and approximation quality depending on the queries as well as the characteristic of the dataset. (2) The approximation quality of an index computed in a stream-based approach depends on three factors: First, we observe an influence of the characteristics of the crawled dataset. Second, simple queries consistently outperform complex queries. Third, a larger window size typically improves the quality only marginally.

Regarding the first insight, we conducted a detailed analysis to understand the relationship between compression ratio and summarization ratio. We computed the Pearson and Spearman correlation coefficient for the $n = 32$ SLI reported in Table 2. Results of the Pearson correlation indicated that there was a significant relationship between compression ratio and summarization ratio, $r(30) = .84, p < .0001$, and as well as for Spearman, $r_s(30) = .64, p < .0001$. Furthermore, there is a significant negative correlation between summarization ratio and approximation quality of a stream-based computation approach. We computed the Pearson and Spearman correlation coefficient for the three cache sizes $1k$, $100k$, and $200k$. We compared the reported F1-scores for the complex queries (for $k = 0$, we used the simple queries) for each cache size (Figure 4) to the summarization ratio of the corresponding gold standard index (Table 2). For all three cache sizes, we found a negative correlation coefficient with $p < .05$ (see Table 4).

From the statistical analysis, we can see that a lower summarization ratio leads to a higher F1-score. This means index structures that summarize well, i.e., summarize many

Table 4. Results of correlation analysis between summarization ratio and approximation quality (F1-score). Pearson and Spearman coefficients and respected p-values for $n = 32$ SLI (Table 2) for three cache sizes with a degree of freedom $df = 30$.

Coefficient	1k		100k		200k	
Pearson	$-.35$	$p < .05$	$-.38$	$p < .04$	$-.38$	$p < .04$
Spearman:	$-.74$	$p < .0001$	$-.74$	$p < .0001$	$-.75$	$p < .0001$

data instances to the same schema element, can be computed with high accuracy in a stream-based approach. When we compute correct schema elements in the stream-based approach, for index models with a low summarization ratio, we assign more data instances to the correct schema element than for index models with a high summarization ratio.

Note that the extreme summarization ratio of Weak Property Cliques also produces the highest F1-scores. This can point to a explanation for the observed correlation. With only a handful of schema elements in the index (see Table 2), it is more likely that a data instance is summarized by the correct schema element. This results in an overall higher F1-score.

We also observe an influence of the characteristics of how the data has been crawled. First, all indices have on average a .15 lower F1-score on the DyLDO-127M dataset compared to the TimBL-11M dataset. We explain this observation by the different crawling strategies. The TimBL-11M dataset was crawled starting from a single seed URI. In contrast, the DyLDO-127M dataset was crawled using more than $95,000$ seed URIs from 652 unique pay-level domains [19]. Furthermore, the crawling depth is limited to two hops. Because of this difference in the crawling strategy, data instances bare different characteristics in both datasets. First, the DyLDO-127M dataset contains nearly 4-times more unique properties and about 11-times more unique types as the TimBL-11M dataset (see Section). Moreover, the TimBL-11M contains fewer data sources, and data instances are defined in fewer data sources than in the DyLDO-127M dataset. This could be one possible explanation for the overall better performance on the TimBL-11M dataset. The dataset characteristic also influences the size of the index. On average, the compression ratio of indices computed for the TimBL-11M dataset is 14.9% and for the DyLDO-127M dataset, it is 10.5%. Additionally, data instances in the DyLDO-127M dataset have more variety in the number of outgoing properties, but less variety in the number of types. However, the indices using types (SchemEX, TermPicker, SchemEX+U+I) consistently achieve better compression and summarization ratios on the TimBL-11M dataset. The evaluated indices not using types (Characteristic Sets, W-Property Cliques, SemSemts) achieve better compression and summarization ratios on the DyLDO-127M dataset. Thus, the complexity of the combination of type sets and properties seems to be predominately impacted by the number of properties rather than the number of types.

Finally, we observe that inferencing RDF Schema information on-the-fly (SchemEX+U+oI) leads to lower F1-scores than inferencing in a pre-processing step (SchemEX+U+pI). For SchemEX+U+oI, the schema graph information used for inferencing is incomplete until the last triple using an RDFS property is processed. Thus, for SchemEX+U+oI inferencing is another source for approximation errors. However, while including the semantics of `owl:sameAs` and RDFS increases the size of the index, it reduced the number of schema elements in some experiments, i. e., it achieves a better summarization ratio.

8. Conclusion

Our empirical evaluations reveal huge variations in compression ratio, summarization ratio, and approximation quality for different index models, queries, and datasets. This confirms our hypothesis that there is no single schema-level index model that equally fits all tasks and that the performance of the SLI model depends on the specific types of queries and characteristics of the datasets. However, we observed meaningful correlations in the results that help to determine the right index model for a given task, type of query, and dataset.

Data availability and Reproducibility On GitHub, one can find our source code, the queries, and the detailed statistics about the TimBL-11M and DyLDO-127M datasets: `http://github.com/t-blume/fluid-framework`.

References

1. Thanh Tran, Peter Haase, and Rudi Studer. Semantic search - using graph-structured semantic models for supporting the search process. In *ICCS*, pages 48–65. Springer, 2009.
2. Katja Hose, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. Database foundations for scalable RDF processing. In *Reasoning Web. - Int. Summer School*, volume 6848 of LNCS, pages 202–249. Springer, 2011.
3. Yuangui Lei, Victoria S. Uren, and Enrico Motta. SemSearch: A search engine for the semantic web. In *EKAW*, volume 4248, pages 238–245. Springer, 2006.
4. Thomas Gottron, Ansgar Scherp, Bastian Krayer, and Arne Peters. LODatio: using a schemalevel index to support users infinding relevant sources of Linked Data. In K-CAP, pages 105–108. ACM, 2013.
5. Fabio Benedetti, Sonia Bergamaschi, and Laura Po. Exposing the underlying schema of LOD sources. In *Joint IEEE/WIC/ACM WI and IAT*, pages 301–304. IEEE, 2015.
6. Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. Loupe - an online tool for inspecting datasets in the Linked Data cloud. In *ISWC Posters & Demos*, volume 1486. CEUR-WS.org, 2015.
7. Emmanuel Pietriga, Hande Gözükan, Caroline Appert, Marie Destandau, Sejla Cebiric, François Goasdoué, and Ioana Manolescu. Browsing linked data catalogs with LODAtlas. In *ISWC*, volume 11137, pages 137–153. Springer, 2018.
8. Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. ABSTAT: ontology-driven linked data summaries with pattern minimalization. In *ESWC Satellite Events, Revised Selected Papers*, volume 9989, pages 381–395, 2016.
9. Thomas Neumann and Guido Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In ICDE, pages 984–994. IEEE, 2011.
10. Johann Schaible, Thomas Gottron, and Ansgar Scherp. TermPicker: Enabling the reuse of vocabulary terms by exploiting data from the Linked Open Data cloud. In ESWC, volume 9678, pages 101–117. Springer, 2016.
11. Marek Ciglan, Kjetil Nørvåg, Ladislav Hluchý. The SemSets model for ad-hoc semantic list search. In *WWW*, pages 131–140. ACM, 2012.
12. Till Blume and Ansgar Scherp. Indexing data on the web: A comparison of schema-level indices for data search. In *Database and Expert Systems Applications*, Lecture Notes in Computer Science. Springer, 2020.

13. Mathias Konrath, Thomas Gottron, Steffen Staab, and Ansgar Scherp. SchemEX - efficient construction of a data catalogue by stream-based indexing of Linked Data. *J. Web Sem.*, 16:52– 58, 2012.

14. Š. Čebiric, F. Goasdoué , H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika. Summarizing semantic graphs: a survey. *VLDB J.*, 28(3):295–327, 2019.

15. François Goasdoué , Pawel Guzewicz, and Ioana Manolescu. Incremental structural summarization of RDF graphs. In *EDBT*, pages 566–569. OpenProceedings.org, 2019.

16. Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4):15:1–15:41, 2009.

17. Thanh Tran, Günter Ladwig, and Sebastian Rudolph. Managing structured and semi-structured RDF data using structure indexes. *TKDE*, 25(9):2076–2089, 2013.

18. Robert Isele, Jürgen Umbrich, Christian Bizer, and Andreas Harth. LDspider: An open-source crawling framework for the web of linked data. In *ISWC Posters & Demonstrations*, volume 658. CEUR-WS.org, 2010.

19. Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. Observing linked data dynamics. In *ESWC*, volume 7882, pages 213–227. Springer, 2013.

20. T. Blume and A. Scherp. FLuID: A meta model to flexibly define schema-level indices for the web of data. *CoRR*, abs/1908.01528, 2019.

21. Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto , and Pablo de la Fuente. An empirical study of real-world SPARQL queries. *CoRR*, abs/1103.5043, 2011.