Journal of Data Intelligence and Predictive Systems (JDIPS) **2025, 1(1): 1-8**
https://www.longmanpublishers.com/

Index Models at the Schema Level for Web Data Search

Farhad
Al-Mustafa International University, Qom, Iran

Abstract: Finding and examining data sources is one of the many options provided by indexing the Web of Data. Selecting an appropriate index model—that is, how to index and summarize data—is a key design choice when indexing the Web of Data. Several attempts have been made to create particular index models for a particular goal. It is still challenging to determine whether a method generalizes effectively to a different task, collection of queries, or dataset because each index model is created, put into use, and assessed separately. Six typical index models with distinct feature combinations are empirically evaluated in this paper. One of these is a novel index model that combines owl:sameAs and inferencing over RDFS. For the first time, we combine all index models into a single stream-based architecture. Using two sizable, real-world datasets, we assess versions of the index models taking into account sub-graphs of size 0, 1, and 2 hops. We assess the indices' quality in terms of the F1-score, which indicates the approximation quality of the stream-based index computation, the compression ratio, and the summarization ratio. For various index architectures, queries, and datasets, the trials show significant differences in the approximation quality, summarization ratio, and compression ratio. Nonetheless, we find significant connections in the outcomes that aid in selecting the appropriate index model for a certain task, query type, and dataset.
Keywords: Graph Summarization; Schema-level Graph Indices; Data Search.

1. Introduction

Large heterogeneous graphs, such as the Web of Data, can be effectively managed with the help of well-established graph indices. For the Web of Data, instance-level and schema-level indexes may generally be distinguished. Finding individual data instances is the main goal of instance-level indices [1, 2, 3]. For example, one can look for a book by title, like "Towards a clean air policy." Schema-level indices, or SLIs for short, facilitate structural queries, such as looking for data instances with the RDF type bibo:book and the property dct:creator [4]. An SLI model specifies which types and property combinations are indexed, how data instances are compiled, and which queries the index supports.
Data exploration [5, 6, 7, 8], query size estimate [9], vocabulary phrase suggestion [10], related entity retrieval [11], data search [4], and other tasks have all previously been addressed by a variety of SLI models. Finding (sub-)graphs on the Web that correspond to a specified schema structure is the goal of data search.

Figure 1 shows how SLIs are used to search for data on the Web of Data. To obtain the actual data instances, a SLI is first queried to find pertinent data sources, which are then accessible using HTTP get requests in a subsequent stage. To provide a search for pertinent data sources or an investigation of data sources, search systems such as LODatio [4], LODeX [5], Loupe [6], and LODatlas [7] rely on SLI.

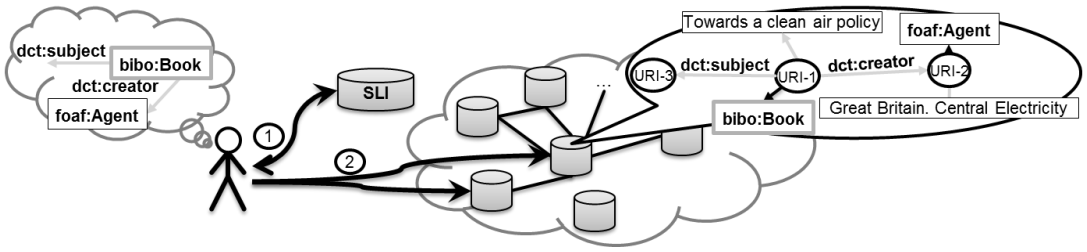

Fig. 1. Finding data sources on the Web using a schema-level index (SLI). A structural query is executed over an SLI to identify relevant data sources (1). Subsequently, the data sources are accessed to retrieve actual data instances (2).

The issue is that every SLI model was created, put into practice, and assessed specifically for that task utilizing various queries, datasets, and metrics. We hypothesize that no SLI model is universally applicable and that the performance of a given SLI is contingent upon the particular query types and dataset properties. However, there hasn't been much research done to date on how SLI models behave in various settings. It is still challenging to determine whether a method generalizes successfully to another task because each SLI model is assessed separately in a particular setting. Stated differently, it is unknown which SLI model is appropriate for specific activities, datasets, and circumstances.

The outcomes of querying on the Web of Data are influenced by various SLI models, queries, and datasets, as demonstrated in this paper's thorough comparison of SLI models [12]. We do a comprehensive empirical analysis of SLI models that are representative. To do this, we have for the first time outlined and put into practice the characteristics of pre-existing SLI models in a shared framework that is accessible on GitHub. We selected six SLI models with distinct feature combinations that were created for various tasks based on the discussion of related studies in order to comprehend and contrast how they behaved for the data search task. In three variations, we index sub-graphs of 0, 1, and 2 hop lengths to empirically examine the behavior of each chosen SLI model.

There are two sets of experiments in the empirical evaluation. In a first set of experiments, we examine the number of schema elements in the SLI in relation to the number of data instances in the dataset (summarization ratio) and the relative size of the calculated SLI in relation to the original dataset (compression ratio). The quality of a stream-based computation of the SLI for sizable datasets retrieved from the Web of Data is measured in the second set of studies. The stream-based method observes the graph across a stream of edges with a constant window size, allowing it to scale to graphs of any size. Due to the small window size, this method may extract partial schema structures, which inherently adds errors into the SLI calculation [13]. Our tests reveal significant differences in the SLI models' approximation quality, summarization ratio, and compression ratio. Strong positive and negative connections between the three indicators are also evident, though. The behaviors of SLI models for various datasets and queries are clarified by these insights, which aid in selecting the best SLI model for a particular task and dataset.

This is how the rest of the paper is organized. We then go over the characteristics of the schema-level index (SLI) models that have been documented in the literature. In Section 3, we codify the way SLIs summarize data instances. We present the experimental setup, including the datasets and our methodology, in Section 4. Section 5 presents the findings from our experiments. Before we wrap up, we finally go over our key results.

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

Different models for schema-level indices (SLI) are the main topic of our discussion of related research. SLIs facilitate the execution of structural searches throughout the Web of Data, as previously mentioned. Types and properties are combined in structural inquiries, like the one shown in Listing 1. Different SLI models that represent distinct schema structures and are built using different theoretical techniques have been described in the past [14]. An overview of SLI models and the particular capabilities they support that have been documented in the literature is given in Table 1. Property sets [9, 11, 13, 8, 5, 6, 10, 15], type sets [13, 8, 5, 6, 10, 15], neighbor information [11, 13, 8, 5, 6, 10, 15], path information [9, 11, 13, 8, 5, 6, 15], k-bisimulation [15], incoming property sets [9, 15], OR combination of features [15], transitively co-occurring (related) properties [15], and inferred information from RDF Schema properties [8, 15] are some of these features. Furthermore, as a novel feature, we suggest inferencing over owl:sameAs. From top to bottom, we go over the SLI models shown in Table 1 below. We introduce their feature combination, task, and schema structure.

To optimize cardinality estimations for join queries in RDF databases, Characteristic Sets [9], as shown in Figure 2a, were created. Formally, a first-order-logic statement over triples is used to define the SLI model as sets of data instances. Data instances are summarized using characteristic sets along common outgoing properties (the property set) and incoming properties (the incoming property set). The SLI model SemSets [11], which is related to Characteristic collections, was created to find semantically similar collections of data instances in knowledge graphs in order to enhance keyword-based ad hoc retrieval. It is depicted in Figure 2b. Set operators are used to define SemSets. Data instances that are connected to a shared resource and have similar outgoing properties are summarized by the SemSets model. As a result, the model incorporates neighbor information and property sets.

For RDF graphs, Goasdou´e et al. [15] present succinct graph summaries. They suggest "Property Cliques," which are equivalence relations that can represent various schema topologies. Based on the co-occurrence of related properties, Property Cliques provide a summary of data instances. According to their work, if two

attributes appear together in a data instance, they are connected. Consequently, the authors also noted that the link of being related is transitive [15]. Additionally, property sets and inbound property sets are used by Goasdou´e et al. They provide weak equivalence connections, in which either the incoming or the exiting attributes are the same (the OR combination), and strong equivalence relations, in which the related incoming and outgoing properties are the same. To calculate their summaries, they also support RDF Schema inferencing. The only index model that use transitively co-occurring (related) properties and summarizes instances based on the same property sets, the same incoming property sets, or both (OR combination) is the weak equivalence version of the Property Cliques. As seen in Figure 2c, we refer to this variation of the compact graph summary as Weak Property Clique.

Data instances are summarized by ABSTAT [8], LODeX [5], Loupe [6], and SchemEX [13] using a shared set of RDF types (the type sets) and properties that link to resources of the same type (shown in Figure 2d). Therefore, they combine type sets with neighbor information. To investigate datasets, ABSTAT, LODeX, and Loupe were created. The schema structure of ABSTAT is only loosely stated in a textual description and lacks a formal specification. ABSTAT [8] can compute so-called minimum patterns in terms of the SLI model features by inferencing over the RDFS type hierarchies. The collection of the most particular RDF types connected to the instances is known as a minimal pattern. These most specific kinds are eliminated together with all of their supertypes. Goasdou´e et al. [15] also support this functionality, and they also consider RDFS domain and range and sub-properties. Likewise, Loupe's SLI model is only loosely defined, which allows for interpretation in certain areas. Three so-called Inspectors—Class, Property, and Triple Inspector—that handle various query types are supported by the Loupe model. Only type sets are taken into account by the Class Inspector, only property sets by the Property Inspector, and both type sets and property sets by the Triple Inspector. Triples of the format < subjectType, predicate, objectType > are extracted from the input graph by the Triple Inspector. As a result, the SLI models ABSTAT and LODeX are equivalent to the schema structure that Triple Inspector captured. The SLI model of LODeX computes clusters over the RDF types and chooses a representative RDF type per cluster, in contrast to ABSTAT and Loupe. A stream-based schema extraction method was used to assess SchemEX for the data search task on Web of Data snapshots. Stratified 1-bisimulation was the definition of SchemEX [16].

TermPicker [10], which is depicted in Figure 2e, was created to generate data-driven vocabulary term recommendations. A common type set, a common property set, and a common type set of all property-linked resources are the basis on which TermPicker summarizes data instances. As a result, unlike SchemEX, for example, TermPicker ignores path information, meaning that it makes no difference which attributes these data instances are related to. Therefore, the ability to aggregate the knowledge of nearby data instances is necessary in order to model TermPicker.

Additionally, by supporting inferencing over owl:sameAs and utilizing all RDFS information, including types, properties, domain, and range, the index model SchemEX+U+I (short for: SchemEX with Unions of sameAs instances and RDFS Inferencing) expands the common schema structure of ABSTAT, LODeX, Loupe, and SchemEX.

By defining an equivalence relation over states and seeing traversal over a graph as operating on state transition systems, bisimulation provides a method for determining if two graphs are equal [16]. If two states undergo the same kind of transition and transform into equivalent states, they are said to be equivalent (or bisimilar). As defined with k = 1 in SchemEX, a stratified k-bisimulation shortens the path length to k hops [13]. Model k-bisimulation is a height parameterization that can be used on SLI, according to Tran et al. [17]. Therefore, instead of using a distinct index model, we choose to use k-bisimulation as a feature. According to Tran et al., the parameter is usually $k \in 0, 1, 2$.

In conclusion, we can say that there are numerous SLI models that capture various schema patterns and are appropriate for various purposes based on the discussion of related studies. For their particular duties, SLI models are independently developed, put into practice, and assessed.

3. Summarizing Data Instances with Schema-level Indices

Using schema-level indices (SLI), we describe the fundamental ideas of data graph, data instances, and summarization of instances. $G \subseteq VU \cup VB \times P \times (VU \cup VB \cup L)$ is the definition of an RDF data graph, where VU represents the set of URIs, VU represents the set of blank nodes, $P \subseteq VU$ represents the collection of properties, and L represents the set of literals. Moreover, all RDF classes are contained in the subset $VC \subseteq VU \cup VB$. A triple is a subject-predicate-object expression $(s, p, o) \in G$ that states something about a resource $s \in VU \cup VB$.

A data instance Is ⊆ G is defined as a collection of triples, each of which has a common subject URI s. With (s, rdf:type, c) ∈ Is, we refer to the set of all c ∈ VC as the type set of Is. Additionally, the set of all p ∈ P with (s, p, o) ∈ Is is referred to as the property set of Is.
The type sets of the instances in our case are shown as squares above the relevant URI in Figure 3. The remaining triples, such as (s-1, p-1, o-1), are shown as directed edges. SLI uses a single schema to summarize data entities. The SLI model specifies the particular properties that are used to calculate a data instance's schema. Characteristic Sets, for instance, solely employs the property sets and incoming property sets of each instance, but the majority of other index models additionally allow for the consideration of RDF types. Please see Section 2 for a thorough discussion of the characteristics found in SLI models. The shared schema of data instances allows for their unique identification. The schema element is this distinct schema identifier that provides a summary of the data instances. The SLI stores schema elements as keys that can be used, for example, to obtain all instances of summary data. One schema element (SE-1) is present in the schema-level index in Figure 3.

One is interested in the data sources of the condensed examples for activities like browsing the Web of Data (see Figure 1 for an inspiring example). We define the data source Ds of an instance Is as the collection of URIs ds that identify Web pages that include triple (s, p, o) about instance Is in order to model where data instances have been located on the Web of Data. The concept of quads (s, p, o, ds) is used by common RDF crawlers to offer this information [18]. This means that for data search, the SLI only has to store the information about the URIs of the data sources where these instances have been observed, rather than directly storing the URIs of the summary data instances. For instance, ds-4 and ds-7 contain the data instances shown in Figure 3. A lookup table that provides details on the schema structures and locations of data instances on the Web is, therefore, logically, a SLI for data search. One would seek for the number of occurrences of summarized data rather than their location for another task, such cardinality estimate.

The concept of inferring implicit information, such as through the use of owl:sameAs or RDFS, is present in the Web of Data. For instance, according to the semantics of a triple (p-1, rdfs:subPropertyOf, p-2) ∈ G, we can infer the additional triple (s, p-2, o) for every instance Is with (s, p-1, o). Inferencing over RDF or owl is assumed in the following:The matching types and properties are added to each instance's associated type sets and property sets by sameAs.

4. Experimental Study

According In a single framework, we incorporate every element from the relevant work (see to Section 2). Our framework enables the flexible combination of these aspects to define both new and current SLI models.

4.1. Selecting Schema-level Index Models

Based on our investigation in Section 2, we choose six sample index models. These SLI models include SchemEX, SchemEX+U+I, TermPicker, SemSets, Weak Property Clique, and Characteristic Sets. Because they offer distinctive feature combinations, we choose TermPicker, SemSets, Weak Property Clique, and Characteristic Sets. Additionally, as the SLI model has the same base schema structure as LODex, Loupe, and ABSTAT, we choose SchemEX (see Table 1). Lastly, because SchemEX+U+I employs owl:sameAs and complete RDFS reasoning, we choose it. This also includes the RDFS reasoning offered by the compact graph summaries and the RDFS type hierarchy inferencing of ABSTAT. We use the k-height parameterization feature suggested by Tran et al. [17] for each of the six chosen SLI models. For the height parameterization k, 0, 1, and 2 are reasonable options [17]. We thus compare a total of eighteen distinct index models.

4.2. Information Sources

Two datasets that were crawled from the Web of Data are used. The TimBL-11M dataset is the first one. It starts with a single seed URI, Tim Berners-Lee's FOAF profile, and comprises roughly 11 million triple crawls in a breadth-first search [13]. With 673k data instances spread across 18k data sources, the TimBL-11M dataset is a graph. On average, each instance is described in three data sources (SD: 40), has three entering properties (SD: 848), three types (SD: 40), and thirteen outbound properties (SD: 275). The latter makes it particularly difficult to look for data because triples with a common subject URI can be found in three different data sources on overage. The TimBL-11M dataset contains 2, 738 distinct RDF types and 3, 919 unique attributes overall.

There are 127M triples in the second dataset, DyLDO-127M [19]. Snapshots of the Web of Data are regularly provided by the Dynamic Linked Data Observatory (DyLDO). Approximately 95k representative seed URIs of the Web of Data are crawled in a breadth-first search to create the DyLDO dataset, as opposed to TimBL-11M, which employs only one seed URI [19]. Two hops is the maximum crawling depth that these seed URIs can

reach. The initial snapshot of the DyLDO dataset, which has 127M triples the largest one, is what we use. There are 7M data instances spread among 154k data sources in the DyLDO-127M dataset. On average, each instance is defined in two data sources (SD: 17), contains seven incoming properties (SD: 635), 17 outbound properties (SD: 6503), and a single type (SD: 17). The DyLDO-127M dataset has 31k unique types and 15k unique characteristics overall.

With 11 million triples and 127 million triples, respectively, both datasets are sizable enough to compute a gold standard for the stream-based index computation. By loading the complete dataset into main memory and computing the indices without regard to window size, a gold standard is produced. With the exception of eliminating triples that did not adhere to W3C standards, no pre-processing was done on the datasets.

5. Compression and Summarization Ratio in Experiment 1

5.1. Overview

We assess the index size for the chosen indices across the two aforementioned datasets. When an index is kept as an RDF graph, its size is the number of triples. The number of triples in the dataset (compression ratio) and the number of triples in the index are compared. Additionally, we examine the summarization ratio—the number of schema items in the index to the number of data instances in the dataset. This ratio provides insight into how well the specified schema structure may condense Web of Data data instances. We use exact indices for the summarization and compression ratios. This indicates that before beginning the index computation procedure, we loaded the entire data graph into the main memory.

5.2. Findings

Table 2 records the outcomes of the tests according to the summarization ratio and compression ratio. As can be seen, the degree to which indices effectively condense and condense the data varies greatly. With k = 1, SemSets' compression ratio for the TimBL-11M dataset is almost ten times more than that of every other index, with the exception of Weak Property Cliques, which is only roughly five times larger. SemSets' compression ratio (with k = 1) is up to 75 times greater for the DyLDO-127M dataset. Furthermore, the index size increases more than 10 times from k = 0 to k = 1, but not from k = 1 to k = 2. The summary ratio shows a comparable rise. The only index that compares the object URIs of each (s, p, o) triple is SemSets, which uses neighbor information but not neighbor type sets. The other indices, on the other hand, either disregard objects or simply take into account their type sets. The summarization ratio of SemSets is between 20% and 25%, meaning that four to five data instances often have the same schema structure. The summarization ratio for Characteristic Sets, the smallest index, is 0.3%, meaning that roughly 330 data instances have the same schema structure. The Weak Property Clique is a noteworthy outlier, exhibiting the most condensed summarization (summarization ratio of less than 0.1%). However, a significantly high compression ratio results from the combination of either incoming or exiting related characteristics in Weak Property Cliques. Characteristic indices are more than twice as large as Weak Property Clique indices. creates indices.

The index size grows by roughly 3% more triples than SchemEX when the semantics of RDFS and owl:sameAs in SchemEX+U+I are taken into account. When the semantics of RDFS and owl:sameAs are included, fewer schema components are generated for k = 1, even though the index has more triples. In order to summarize the data instances, SchemEX+U+I needs more schema components than SchemEX for k = 0 and k = 2.

In conclusion, adding the RDF Schema and owl:sameAs semantics expands the index's size. It can, however, result in fewer schema elements. Additionally, employing weak equivalencies results in a small number of very big schema components that summarize every occurrence of the data.

6. Stream-based Index Computation in Experiment 2

6.1. Synopsis

The purpose of this experiment is to determine how effectively the indices support queries with different levels of complexity when the SLI is calculated across a stream of graph edges. The concept, which was inspired by stream-databases, is to view the triples in the datasets as a stream that is accessed through windows of sizes of 1,000, 100,000, and 200,000. In theory, this enables us to scale the calculation to input graphs of any size [13]. However, because only a portion of the data graph is simultaneously stored in main memory and the rest is unknown or unavailable, the method results in approximation mistakes. As a result, we might extract schema structures that are not full.

6.2. SchemEX+U+I: Pre-processing versus On-the-Fly Processing

In this experiment, we assess two variations of the SchemEX+U+I index model: The so-called schema graph is an extra data structure needed for the RDFS inferencing computation [20]. The triples are built into this schema graph using RDFS range, domain, subClassOf, or subPropertyOf. We deduce additional types and attributes for the remaining data instances using the domain, range, and hierarchical types/properties information. One variant, SchemEX+U+oI, extracts and infers the RDFS information in real time. In this case, we build the schema graph concurrently with the index calculation. The benefit is that the dataset only needs to be run once. However, information for the inferencing may be absent because the schema graph is constructed when the index is being calculated. To create the schema graph in the alternative version, SchemEX+U+pI, we first extract all RDFS data as a pre-processing step. The benefit is that only the entire schema graph is used for triple inferencing. The disadvantage is that the dataset must be run twice.

6.3. Inquiries

Selecting which queries to run across the indices is a major challenge for this experiment. Here, we build on the work of Konrath et al. [13], who evaluated approximation graph indexes using a data-driven query generation approach. This indicates that the actual data instances in the datasets—that is, their combination of types and properties—are the source of the queries. Simple queries (SQ) and complex queries (CQ) are the two categories of inquiries that we distinguish. Data instances with a common type set (or, in the case of SemSets, a common set of objects) are sought for by simple searches. The majority of SPARQL requests in search systems are simple questions, according to analyses of existing query logs [21]. Complex searches, on the other hand, look for data instances that match the entire schema structure specified by the particular index model; for example, characteristic sets with k = 2 may have property routes spanning two hops.

6.4. Quantifications

We run both basic and sophisticated queries on the gold standard SLI and the SLI calculated with a fixed window size. The two sets Dgold and Dwindow, which contain the corresponding data source URIs, are the queries' results for our data search task. In accordance with Konrath et al. [13], the F1-score is used to compare Dgold and Dwindow in order to determine the approximation quality.

6.5. Findings

The approximation quality for the chosen index models is displayed in Figure 4 as an F1-score. The results are displayed as a table in Table 3. The simple and sophisticated queries are similar for indices with a height parameter k = 0. Furthermore, type information (or object information) is not used by Weak Property Cliques or Characteristic Sets. For these index structures, straightforward queries are therefore unavailable.

Our experiment's findings allow us to conclude that simple searches routinely exhibit better F1-scores than complex queries. The only indices with a better F1-score on the DyLDO-127M dataset than on the TimBl-11M dataset are TermPicker and Weak Property Cliques. The only index that does not use the path information feature is TermPicker, as explained in Section 2. The only distinction between the schema structure and SchemEX is this restriction. On the TimBL-11M dataset, however, TermPicker's F1-score is 50% worse than SchemEX's.

The tests with k = 1 and 2 have the highest F1-scores for Weak Property Cliques in relation to the complex questions. Weak Property Cliques have F1-scores that are 0.12 to.54 greater than the other indices for k = 2. The F1-score of SemSets only slightly decreases from k = 1 to k = 2. The F1-score of SchemEX+U+oI is consistently lower than that of SchemEX. In a pre-processing step, we extracted the RDF Schema information for SchemEX+U+pI. SchemEX+U+pI has consistently higher F1-scores than SchemEX+U+oI. Additionally, SchemEX+U+pI outperforms SchemEX in terms of F1-scores for window sizes of 100k and 200k (with the exception of k = 2 for TimBL-11M).

Additionally, we see that the crawled dataset's properties have an impact on the quality of the approximation. When comparing the DyLDO-127M dataset to the TimBL-11M dataset, the average F1-score for all indices is.15 lower. Specifically, simple inquiries have significantly lower F1-scores. When comparing the DyLDO-127M dataset to the TimBL-11M dataset, basic searches typically have.25 lower F1-scores and complex queries have.04 lower F1-scores. Furthermore, F1-scores are regularly improved by increasing window sizes. In contrast, our experiment's F1-scores were lower with on-the-fly inferencing than without it.

7. Conversation

Important conclusions drawn from our experiments are: (1) The performance of SLI models varies greatly based

on the queries and dataset characteristics in terms of the summarization ratio, compression ratio, and approximation quality. (2) Three criteria determine the approximation quality of an index combined in a stream-based approach: First, we see that the crawled dataset's properties have an impact. Second, straightforward queries always perform better than complicated ones. Third, the quality usually only slightly improves with a greater window size. In order to comprehend the relationship between the summarization ratio and the compression ratio, we carried out a thorough investigation with reference to the first insight. For the n = 32 SLI shown in Table 2, we calculated the Pearson and Spearman correlation coefficients. According to the results of the Pearson correlation, the compression ratio and summarization ratio were significantly correlated (r(30) =.84, p <.0001), and the Spearman was significantly correlated (rs(30) =.64, p <.0001). Additionally, the summarization ratio and the approximation quality of a stream-based processing approach have a considerable negative association. We calculated the Spearman and Pearson correlation coefficients for the 1k, 100k, and 200k cache sizes. We contrasted the summarization ratio of the associated gold standard index (Table 2) with the reported F1-scores for the complicated queries (for k = 0, we utilized the simple queries) for each cache size (Figure 4). We discovered a negative correlation coefficient with p <.05 for each of the three cache sizes (see Table 4).

A lower summarization ratio results in a higher F1-score, according to the statistical analysis. This implies that a stream-based method can generate index structures with high accuracy that summarize well, that is, summarize multiple data instances to the same schema element. We allocate more data instances to the correct schema element for index models with a low summary ratio than for index models with a high summarization ratio when we compute correct schema elements using the stream-based technique.

Keep in mind that the highest F1-scores are also obtained from Weak Property Cliques with the excessive summarization ratio. This may suggest a reason for the correlation that has been noticed. A data instance is more likely to be summarized by the appropriate schema element when there are only a few schema items in the index (see Table 2). As a result, the F1-score is raised overall.

We also see that the features of the data crawling process have an impact. First, when comparing the DyLDO-127M dataset to the TimBL-11M dataset, the average F1-score for all indices is.15 lower. We use the various crawling strategies to explain this observation. A single seed URI served as the starting point for the crawl of the TimBL-11M dataset. On the other hand, almost 95,000 seed URIs from 652 distinct pay-level domains were used to crawl the DyLDO-127M dataset [19]. Additionally, there is a two-hop limit on the crawling depth. Data instances in the two datasets exhibit distinct features as a result of this variation in the crawling approach. First, compared to the TimBL-11M dataset, the DyLDO-127M dataset has approximately four times as many unique attributes and eleven times as many distinct types (see Section). Furthermore, compared to the DyLDO-127M dataset, the TimBL-11M has less data sources and data instances described in fewer data sources. The overall better performance on the TimBL-11M dataset may be explained in part by this. The magnitude of the index is also influenced by the dataset feature. For the TimBL-11M dataset, the average compression ratio of indices is 14.9%, whereas for the DyLDO-127M dataset, it is 10.5%. Furthermore, there is less variation in the number of types but more variation in the number of outgoing attributes among data instances in the DyLDO-127M dataset. On the TimBL-11M dataset, however, the indices that use types (SchemEX, TermPicker, and SchemEX+U+I) consistently obtain superior summarization and compression ratios. On the DyLDO-127M dataset, the evaluated indices that do not use types (Characteristic Sets, W-Property Cliques, and SemSemts) achieve greater summarization and compression ratios. Therefore, it appears that the number of properties, rather than the number of types, has a greater influence on the complexity of the combination of type sets and properties.

Finally, we find that inferencing RDF Schema information in a pre-processing stage (SchemEX+U+pI) results in higher F1-scores than inferencing on-the-fly (SchemEX+U+oI). SchemEX+U+oI requires processing of the final triple using an RDFS property before the schema graph information used for inferencing is complete. Inferencing is hence another source of approximation mistakes for SchemEX+U+oI. Even though adding the owl:sameAs and RDFS semantics makes the index larger, in several experiments it decreased the number of schema items, improving the summarization ratio.

8. Conclusion

For various index structures, queries, and datasets, our empirical evaluations show significant differences in the compression ratio, summarization ratio, and approximation quality. This supports the idea that there isn't a unique schema-level index model that works well for every task and that the SLI model's efficiency varies depending on the particular query types and dataset properties. Nonetheless, we found significant connections in the outcomes that aid in selecting the appropriate index model for a certain task, query type, and dataset.

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, Mar´ıa Poveda-Villal´on, Raúl Garc´ıa-Castro, and Asuncio´n Go´mez-P´erez. 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, Franc¸ois Goasdou´e, 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 Hluchy´. 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. Goasdou´e , H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika. Summarizing semantic graphs: a survey. *VLDB J.*, 28(3):295–327, 2019.
15. Franc¸ois Goasdou´e , Pawel Guzewicz, and Ioana Manolescu. Incremental structural summariza- tion 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 Ka¨fer, 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. Ferna´ndez, Miguel A. Mart´ınez-Prieto , and Pablo de la Fuente. An empirical study of real-world SPARQL queries. *CoRR*, abs/1103.5043, 2011.