Structural Properties as Proxy for Semantic Relevance in RDF Graph Sampling*

Laurens Rietveld[1], Rinke Hoekstra[1,2], Stefan Schlobach[1], and Christophe Guéret[3]

[1] Dept. of Computer Science, VU University Amsterdam, NL
{laurens.rietveld,k.s.schlobach,rinke.hoekstra}@vu.nl
[2] Leibniz Center for Law, University of Amsterdam, NL, hoekstra@uva.nl
[3] Data Archiving and Network Services (DANS), NL,
christophe.gueret@dans.knaw.nl

Abstract. The Linked Data cloud has grown to become the largest knowledge base ever constructed. Its size is now turning into a major bottleneck for many applications. In order to facilitate access to this structured information, this paper proposes an automatic sampling method targeted at maximizing answer coverage for applications using SPARQL querying. The approach presented in this paper is novel: no similar RDF sampling approach exist. Additionally, the concept of creating a sample aimed at maximizing SPARQL answer coverage, is unique. We empirically show that the relevance of triples for sampling (a semantic notion) is influenced by the topology of the graph (purely structural), and can be determined without prior knowledge of the queries. Experiments show a significantly higher recall of topology based sampling methods over random and naive baseline approaches (e.g. up to 90% for Open-BioMed at a sample size of 6%).

Keywords: subgraphs, sampling, graph analysis, ranking, Linked Data

1 Introduction

The Linked Data cloud grows every year [4,10] and has turned the Web of Data into a knowledge base of unprecedented size and complexity. This poses problems with respect to the scalability of our current infrastructure and tools. Datasets such as DBPedia (459M triples) and Linked Geo Data (289M triples) are central to many Linked Data applications. Local use of such large datasets requires investments in powerful hardware, and cloud-based hosting is not free either. These costs are avoidable if we know which part of the dataset is needed for our application, i.e. if only we could pick the data a priori that is actually being used, or required to solve a particular task. Experience in the OpenPHACTS and Data2Semantics projects[1] shows that for the purposes of prototyping, demoing

* This work was supported by the Dutch national program COMMIT, and carried out on the Dutch national e-infrastructure with the support of SURF Foundation.

[1] See http://openphacts.org and http://www.data2semantics.org, respectively.

or testing, developers and users are content with relevant subsets of the data. They accept the possibility of incomplete results that comes with it. A locally available subset is also useful when the connection to a cloud based server is inaccessible (something which happens frequently [10]). Since more users can host subsets of very large data locally, this will lift some of the burden for (often non-commercial) Linked Data providers, while links to the remaining parts on external servers remain in place.

Our analysis of five large datasets (>50M triples) shows that for a realistic set of queries, at most 2% of the dataset is actually used (see the 'coverage' column in Table 1): a clear opportunity for pruning RDF datasets to more manageable sizes.[2] Unfortunately, this set of queries is not always known: queries are not logged or logs are not available because of privacy or property rights issues. And even if a query set is available, it may not be representative or suitable, e.g. it contains queries that return the entire dataset.

We define *relevant sampling* as the task of finding those parts of an RDF graph that maximize a task-specific relevance function while minimizing size. For our use case, this relevance function relies on semantics: we try to find the smallest part of the data that entails as many of the original answers to typical SPARQL queries as possible. This paper investigates whether we can use structural properties of RDF graphs to predict the relevance of triples for typical queries.

To evaluate this approach, we represent "typical use" by means of a large number of SPARQL queries fired against datasets of various size and domain: *DBpedia 3.9* [3], *Linked Geo Data* [5], *MetaLex* [19], *Open-BioMed*[3], *Bio2RDF* [8] and *Semantic Web Dog Food* [24] (see Table 1). The queries were obtained from server logs of the triple stores hosting the datasets and range between 800 and 5000 queries for each dataset. Given these datasets and query logs, we then 1) rewrite RDF graphs into directed unlabeled graphs, 2) analyze the topology of these graphs using standard network analysis methods, 3) assign the derived weights to triples, and 4) generate samples for every percentile of the size of the original graph. These steps were implemented as a scalable sampling pipeline, called *SampLD*.

Our results show that the topology of the hypergraph alone helps to predict the relevance of triples for typical use in SPARQL queries. In other words, we show in this paper that without prior knowledge of the queries to be answered, we can determine to a surprisingly high degree which triples in the dataset can safely be ignored and which cannot. As a result, we are able to achieve a recall of up to .96 with a sample size as small as 6%, using *only* the structural properties of the graph. This means that we can use purely *structural* properties of a knowledge base as proxy for a *semantic* notion of relevance.

[2] The figure of 2% depends on the assumption that the part of the graph touched by queries is relatively stable over time. We intend to investigate this further in future work.

[3] See `http://www.open-biomed.org.uk/`

This paper is structured as follows. We first discuss related work, followed by the problem definition and a description of our approach. The fourth section discusses the experiment setup and evaluation, after which we present the results. Finally we discuss our conclusions.

2 Related Work

Other than naive random sampling [30], extracting relevant parts of Linked Data graphs has not been done before. However, there are a number of related approaches that deserve mentioning: relevance ranking for Linked Data, generating SPARQL benchmark queries, graph rewriting techniques, and non-deterministic network sampling techniques.

Network Sampling [23] evaluates several non-deterministic methods for sampling networks: random node selection, random edge selection, and exploration techniques such as random walk. Quality of the samples is measured as the structural similarity of the sample with respect to the original network. This differs from our notion of quality, as we do not strive at creating a structurally representative sample, but rather optimize for the ability to answer the same queries. Nevertheless, the sampling methods discussed by [23] are interesting baselines for our approach; we use the random edge sampling method in our evaluation (See Section 5).

Relevance Ranking Existing work on Linked Data relevance ranking focuses on determining the relevance of individual triples for answering a single query [2,7,12,20,22]. Graph summaries, such as in [11], are collections of important RDF *resources* that may be presented to users to assist them in formulating SPARQL queries, e.g. by providing context-dependent auto completion services. However, summarization does not produce a list of triples ordered by relevance.

TRank [32] ranks RDF entity types by exploiting type hierarchies. However, this algorithm still ranks entities and not triples. In contrast, TripleRank [12] uses 3d tensor decomposition to model and rank triples in RDF graphs. It takes knowledge about different link types into account, and can be seen as a multi-model counterpart to web authority ranking with HITS. TripleRank uses the rankings to drive a faceted browser. Because of the expressiveness of a tensor decomposition, TripleRank does not scale very well, and [12] only evaluate small graphs of maximally 160K triples. Lastly, TripleRank prunes predicates that dominate the dataset, an understandable design decision when developing a user facing application, but it has an adverse effect on the quality of samples as prominent predicates in the data are likely to be used in queries as well.

ObjectRank [7] is an adaptation of PageRank that implements a form of link semantics where every type of edge is represented by a particular weight. This approach cannot be applied in cases where these weights are not known beforehand. SemRank [2] ranks relations and paths based on search results. This approach can filter results based on earlier results, but it is not applicable to *a*

priori estimation of the relevancy of the triples in a dataset. Finally, stream-based approaches such as [15] derive the schema. This approach is not suitable for retrieving the most relevant *factual* data either, regarding a set of queries.

Concluding, existing approaches on ranking RDF data either require prior knowledge such as query sets, a-priori assignments of weights, produce samples that may miss important triples, or focus on resources rather than triples.

Synthetic Queries SPLODGE [14] is a benchmark query generator for arbitrary, real-world RDF datasets. Queries are generated based on features of the RDF dataset. SPLODGE-based queries would allow us to run the sampling pipeline on many more datasets, because we would not be restricted by the requirement of having a dataset *plus* corresponding query logs. However, *benchmark* queries do not necessarily resemble *actual* queries, since they are meant to test the performance of Linked Data storage systems [1]. Furthermore, SPLODGE introduces a dependency between the dataset features and the queries it generates, that may not exist for user queries.[4]

RDF Graph Rewriting RDF graphs can be turned into networks that are built around a particular property, e.g. the social aspects of co-authorship, by extracting that information from the RDF data [33]. Edge labels can sometimes be ignored when they are not directly needed, e.g. to determine the context of a resource [20], or when searching for paths connecting two resources [17].

The networks generated by these rewriting approaches leave out *contextual* information that may be critical to assess the relevance of triples. The triples $\langle :bob, :hasAge, \text{“50”}\rangle$ and $\langle :anna, :hasWeight, \text{“50”}\rangle$ share the same literal ("50"), but it respectively denotes an *age* and a *weight*. Finally, predicates play an important role in our notion of *relevance*. They are crucial for answering SPARQL queries, which suggests that they should carry as much weight as subjects and objects in our selection methodology. Therefore, our approach uses different strategies to remove the edge labels, while still keeping the context of the triples.

In [18], RDF graphs are rewritten to a bipartite network consisting of subjects, objects and predicates, with a separate statement node connecting the three. This method preserves the role of predicates, but increases the number of edges and nodes up to a threefold, making it difficult to scale. Additionally, the resulting graph is no longer *directed*, disqualifying analysis techniques that take this into account.

3 Context

In the previous section, we presented related work on RDF ranking and network sampling, and showed that sampling for RDF graphs has not been done

[4] In an earlier stage of this work, we ran experiments against a synthetic data and query set generated by SP^{2}Bench [29]. The results were different from any of the datasets we review here, as the structural properties of the dataset were quite different, and the SPARQL queries (tailored to benchmarking triple-stores) are incomparable to regular queries as well.

before. This section introduces a very generic framework for such RDF sampling. We elaborate on different RDF sampling scenarios, and present the particular sampling scenario addressed by this paper.

3.1 Definitions

A '*sample*' is just an arbitrary subset of a graph, so we introduce a notion of *relevance* to determine whether a sample is a suitable replacement of the original graph. Relevance is determined by a relevance function that varies from application to application. The following definitions use the same SPARQL definitions as presented in [25].

Definition 1. *An RDF graph $\mathcal{G}$ is a set of triples. A* sample $\mathcal{G}'$ *of $\mathcal{G}$ is a proper subset of $\mathcal{G}$. A sample is* relevant *w.r.t. a relevance function $F(\mathcal{G}', \mathcal{G})$ if it maximizes F while minimizing its size.*

Finding a relevant sample is a multi-objective optimization problem: selecting a sample small enough, while still achieving a recall which is high enough. Moreover, there is no sample that fits all tasks and problems, and for each application scenario a specific relevance function has to be defined. In this paper, relevance is defined in terms of the *coverage of answers* with respect to SPARQL queries.

Definition 2. *The* relevance function for SPARQL querying $F_s(\mathcal{G}', \mathcal{G})$ *is the probability that the solution μ to an arbitrary SPARQL query Q is also a solution to Q w.r.t. $\mathcal{G}'$.*

As usual, all elements $T \in \mathcal{G}$ are triples of the form $(s, p, o) \in I \times I \times (I \cup L)$, where s is called the subject, p the predicate, and o the object of T. I denotes all IRIs, where L denotes all literals. For this paper, we ignore blank nodes.

In other words, a *relevant sample* of a RDF graph is a smallest subset of the graph, on the basis of which the largest possible number of answers that can be found with respect to the original RDF graph. As common in multi-objective optimization problems, the solution cannot be expected to be a single *best* sample, but a set of samples of increasing size.

3.2 Problem Description

The next step is to determine the method by which a sample is made. As briefly discussed in the introduction, this method is restricted mostly by the pre-existence of a suitable set of queries. To what extent can these queries be used to *inform* the sampling procedure? As we have seen in the related work, sampling without prior knowledge – *uninformed* sampling – is currently an unsolved problem. And in fact, even if we *do* have prior knowledge, a method that does not rely on prior knowledge is still useful.

With *informed* sampling, there is a complete picture of what queries to expect: we know exactly which queries we want to have answered, and we consequently know which part of the dataset is required to answer these queries. Given the size of Linked Data sets, this part can still be too large to handle.

Dataset	#Tripl.	Avg. Deg.	Tripl. w/ literals	#Q	Coverage	Q w/ literals	#Triple patt. per query (avg / stdev)
DBpedia 3.9	459M	5.78	25.44%	1640	0.003%	61.6%	1.07 / 0.40
LGD	289M	4.06	46.35%	891	1.917%	70.7%	1.07 / 0.27
MetaLex	204M	4.24	12.40%	4933	0.016%	1.1%	2.02 / 0.21
Open-BioMed	79M	3.66	45.37%	931	0.011%	3.1%	1.44 / 3.72
Bio2RDF/KEGG	50M	6.20	35.06%	1297	2.013%	99.8%	1.00 / 0.00
SWDF	240K	5.19	34.87%	193	39.438%	62.4%	1.80 / 1.50

Table 1: Data and query set statistics

This indicates a need for heuristics or uninformed methods to reduce the size even more.

If we have an incomplete notion of what queries to expect, e.g. we only know part of the queries or only know their structure or features, we could still use this information to create a *semi-informed* selection of the data. This requires a deeper understanding of what features of queries determine relevance, how these relate to the dataset, and what sampling method is the best fit.

In this paper we focus on a comparison of methods for uninformed sampling, the results of which can be used to augment scenarios where more information is available. For instance, a comparison of query features as studied in [26,28], combined with performance of our uninformed sampling methods, could form the basis of a system for semi-informed sampling.

To reiterate, our hypothesis is that we can use standard network metrics on RDF graphs as useful proxies for the relevance function. To test this hypothesis, we implemented a scalable sampling pipeline, SampLD, that can run several network metrics to select the top ranked triples from RDF datasets and evaluate the quality of those samples by their capability to answer real SPARQL queries.

3.3 Datasets

We evaluate the quality of our sampling methods for the six datasets listed in the introduction: DBPedia, Linked Geo Data (LGD), MetaLex, Open-BioMed (OBM), Bio2RDF[5] and Semantic Web Dog Food (SWDF). These datasets were chosen based on the availablity of SPARQL queries. These datasets were the only ones with an available correponding large query set. The MetaLex query logs were made available by the maintainers of the dataset. In the other five cases, we used server query logs made available by the USEWOD workshop series [9].

Table 1 shows that the size of these dataset ranges from 240K triples to 459M triples. The number of available queries per dataset ranges between 193 and 4933. Interestingly, for all but one of our datasets, less than 2% of the triples is actually *needed* to answer the queries. Even though the majority of

[5] For Bio2RDF, we use the KEGG dataset [21], as this is the only Bio2RDF dataset for which USEWOD provides query logs. KEGG includes biological systems information, genomic information, and chemical information.

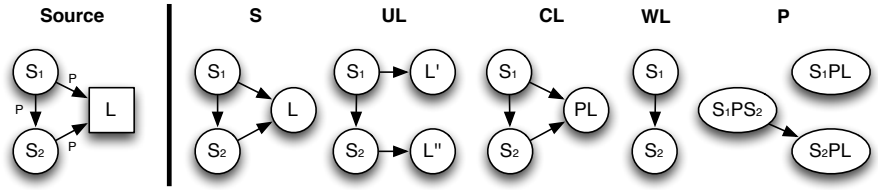


Fig. 1: Rewrite methods

these queries are machine generated (see Section 7), this indicates that only a very small portion of the datasets is relevant, which corroborates our intuition that the costs for using Linked Data sets can be significantly reduced by selecting only the relevant part of a dataset. However, these low numbers make finding this small set of relevant triples more difficult as well. Other relevant dataset properties shown in this table are the average degree of the subjects and objects, the percentage of triples where the object is a literal, the percentage of queries of which at least one binding uses a literal, and the average and standard deviation of the number of triple patterns per query.

4 Sampling Pipeline

The SampLD pipeline calculates and evaluates the quality of samples across different uninformed sampling methods for multiple large datasets.[6] The procedure consists of the following four phases:

1. *rewrite* an RDF graph to a directed unlabeled graph,
2. *analyze* the rewritten graph using standard network analysis algorithms,
3. *assign* the node weights to triples, creating a ranked list of triples,
4. *generate* samples from the ranked list of triples.

We briefly discuss each of the four phases here.

Step 1: Graph Rewriting Standard network analysis methods, are not readily suited for *labeled* graphs, nor do they take into account that a data model may be reflected in the verbatim RDF graph structure in many ways [16]. Since the edge labels (predicates) play an important role in RDF, simply ignoring them may negatively impact the quality of samples (see the related work). For this reason, the SampLD pipeline can evaluate pairwise combinations of network analysis techniques and alternative representations of the RDF graph, and compare their performance across sample sizes.

SampLD implements five rewriting methods: a *simple* (S), *unique literals* (UL), *without literals* (WL), *context literals* (CL), and *path* (P). As can be seen

[6] The SampLD pipeline and evaluation procedure are available online at `https://github.com/Data2Semantics/GraphSampling/`

in Figure 1, the first four methods convert every triple (s, p, o) to a directed edge $s \rightarrow o$. If several triples result in the same edge (e.g. when only the predicate differs), we do not assert that edge more than once. These first four methods differ primarily with respect to their treatment of literal values (i.e. non-IRI nodes) in the graph. It is important to note that for all approaches, any removed literals are re-added to the RDF graph during the round-trip phase detailed below. Since literals do not have outgoing edges they have a different effect on network metrics than IRIs, e.g. by acting as a 'sink' for PageRank.

The Simple (S) method retains the exact structure of the original RDF graph. It treats every *syntactically* unique literal as a single node, taking the data type and language tag of the literal into account. Two occurrences of the literal "50"^^xsd:Integer result in a single node. The Unique literals (UL) method converts every occurrence of a literal as a separate node. The average degree drops, and the graph becomes larger but will be less connected. The Context literals (CL) approach preserves the context of literals, it groups literals that share the same predicate together in a single node. This allows us to make sure that the network metrics distinguish e.g. between integers that express weight, and those that express age. This also results in fewer connections and a lower average degree since predicate-literal pairs will be less frequent than literals. The Without literals (WL) method simply ignores all occurrences of literals. As a result the graph becomes smaller.

The fifth method, Path (P), is triple, rather than resource oriented. It represents every triple (s, p, o) as a single node, and two nodes are connected when they form a path of length 2, i.e. their subject and object must overlap. Asserting edges between triples that share *any* resource discards the direction of the triple, and produces a highly verbose graph, cf. [18], as a resource shared between n triples would generate $\frac{n(n-1)}{2}$ edges. Also, occurrences of triples with `rdf:type` predicates would result in an extremely large number of connections. The path method has the advantage that it results in a smaller graph with low connectedness, and that maintains directedness, where we can assign weights directly to *triples* rather than resources (as with the other methods).

Step 2: Network Analysis In the second step, SampLD applies three common network analysis metrics to the rewritten RDF graph: *PageRank*, *in degree* and *out degree*. These are applied "as is" on the rewritten graphs.

Step 3: Assign Triple Weights Once we have obtained the network analysis metrics for all nodes in the rewritten graph, the (aggregated) values are assigned as *weights* on the triples in the original graph. For method P, we simply assign the value of the node that corresponds to the triple. For the S, UL, WL and CL methods, we retrieve the values for the subject and object of each triple, and assign whichever is *highest* as weight to that triple[7]. When the object of a

[7] One can also use the minimum or average node weight. We found that the maximum value performs better

triple is a literal, it has no corresponding node in the graph produced through the WL method: in such cases, the value of the *subject* will function as weight for the triple as a whole. The result of this assignment phase is a ranked list of triples, ordered by weight in descending order. The distribution of triple weights typically follows a 'long tail' distribution, where large numbers of triples may share the same weight. To prevent potential bias when determining a sample, these triples with equal weights are added to the ranked list in random order.

Step 4: Generating Samples Given the ranked list of triples, generating the sample is a matter of selecting the desired top-k percent of the triples, and removing the weights. The 'best' k value can differ per use-case, and depends on both the minimum required quality of the sample, and the maximum desired sample size. For our current purposes SampLD produces samples for each accumulative percentile of the total number of triples, resulting in 100 samples *each* for every combination of dataset, rewrite method and analysis algorithm.

Implementation Because the datasets we use are quite large, ranging up to 459 Million triples for DBPedia, each of these steps was implemented using libraries for scalable distributed computing (Pig [13] and Giraph [6]). Scale also means that we are restricted in the types of network metrics we could evaluate. For instance, Betweenness Centrality is difficult to paralellize because of the need for shared memory [31]. However many of the tasks *are* parallelizable, e.g. we use Pig to fetch the weights of all triples.

Given the large number of samples we evaluate in this paper (over 15.000, considering all sample sizes, datasets and sampling methods), SampLD uses a novel scalable evaluation method that avoids the expensive procedure (in terms of hardware and time) of loading each sample in triple-stores to calculate the recall.

5 Experiment Setup and Evaluation

The quality of a sample is measured by its ability to return answers on a set of queries: we are interested in the *average recall* taken over all queries. Typically, these queries are taken from publicly available server query logs, discarding those that are designed to return all triples in a dataset, and focusing on SELECT queries, as these are the most dominant. A naive approach would be to execute each query on both the original dataset and the sample, and compare the results. This is virtually impossible, given the large number of samples we are dealing with: 6 datasets means 15.600 samples (one, for every combination of dataset (6), sampling method (15) and baseline (1+10), and percentile(100)), or $1.4 \cdot 10^{12}$ triples in total.

Instead, SampLD (a.) executes the queries once on the original dataset and analyzes which triples are used to answer the query, (b.) uses a cluster to check which weight these triples have. It then (c.) checks whether these triples *would have been* included in a sample, and calculates recall. This avoids the need to load and query each sample. Below, we give a detailed description of this procedure.

Terminology For each graph G we have a set of SELECT queries $\mathcal{Q}$, acting as our relevance measure. Each $Q \in \mathcal{Q}$ contains a set of variables $\mathcal{V}$, of which some may be projection variables $\mathcal{V}_p \subseteq \mathcal{V}$ (i.e. variables for which bindings are returned). Executing a query Q on G returns a result set R_q^g, containing a set of query solutions $\mathcal{S}$. Each query solution $S \in \mathcal{S}$ contains a set of bindings $\mathcal{B}$. Each binding $B \in \mathcal{B}$ is a mapping between a projection variable $V_p \in \mathcal{V}_p$ and a value from our our graph: $\mu : \mathcal{V}_p \rightarrow (I \cup L)$

Required Triples Rewriting a SELECT query into a CONSTRUCT query returns a bag of *all* triples needed to answer the SELECT query. However, there is no way to determine what role individual triples play in answering the query: some triples may be essential in answering all query solutions, others just circumstantial. Therefore, SampLD extracts triples from the query on a *query solution* level. It instantiates each triple pattern, by replacing each variable used in the query triple patterns with the corresponding value from the query solution. As a result, the query contains triple patterns without variables, and only IRIs and literals. These instantiated triple patterns ('query triples') show us which triples are *required* to produce this specific query solution. This procedure is not trivial.

First, because not all variables used in a query are also projection variables, and blank nodes are inaccessible as well, we rewrite every SELECT query to the '`SELECT DISTINCT *`' form and replace blank nodes with unique variable names. This ensures that all nodes and edges in the matching part of the original graph G are available to us for identifying the query triples. However, queries that already expected DISTINCT results need to be treated with a bit more care. Suppose we have the following query and dataset:

```
SELECT DISTINCT ?city  WHERE {
  ?university :inCity ?city ;
              :rating :high .
}
```

```
<university1> :inCity <London> .
<university1> :rating <high> .
<university2> :inCity <London> .
<university2> :rating <high> .
```

Rewriting the query to '`SELECT DISTINCT *`' results in two query solutions, using all four dataset triples. However, we only either need at least the first two triples, or the last two, but not all four. SampLD therefore tracks each distinct combination of bindings for the projection variables $\mathcal{V}_p$.

Secondly, when the clauses of a UNION contain the same variable, but only one clause matches with the original graph G, the other clause should not be instantiated. We instantiate each clause following the normal procedure, but use an ASK query to check wether the instantiated clause exists in G. If it does not exist, we discard the query triples belonging to that clause.

Thirdly, we ignore the GROUP and ORDER solution modifiers, because they do not tell us anything about the actual triples required to answer a query. The LIMIT modifier is a bit different as it indicates that the user requests a specific number of results, but not exactly *which* results. The limit is used in

recall calculation as a cap on the maximum number of results to be expected for a query. In other words, for these queries we don't check whether *every* query solution for G is present for the sample but only look at the proportion of query solutions.

Finally, we currently ignore negations in SPARQL queries since they are very scarce in our query sets. Negations may *increase* the result set for smaller sample sizes, giving us a means to measure precision, but the effect would be negligible.

Calculate Recall For all query triples discovered in the previous step, and for each combination of rewrite method and network analysis algorithm, we find the weight of this triple in the ranked list of triples. The result provides us with information on the required triples of our query solutions, and their weights given by our sampling methods. SampLD can now determine whether a query solution would be returned for any given sample, given the weight of its triples, and given a k cutoff percentage. If a triple from a query solution is not included in the sample, we mark that solution as *unanswered*, otherwise it is *answered*.

Recall is then determined as follows. Remember that query solutions are grouped under distinct combinations of projection variables. For every such combination, we check each query solution, and whenever one of the grouped query solutions is marked as *answered*, the combination is marked answered as well. For queries with OPTIONAL clauses, we do not penalize valid query solutions that do not have a binding for the optional variable, even though the binding may be present for the original dataset.[8] If present, the value for the LIMIT modifier is used to cap the maximum number of answered query solutions.

The recall for the *query* is the number of answered projection variable combinations, divided by the total number of distinct projection variable combinations. For each *sample*, SampLD uses the average recall, or arithmetic mean over the query recall scores as our measure of relevance.

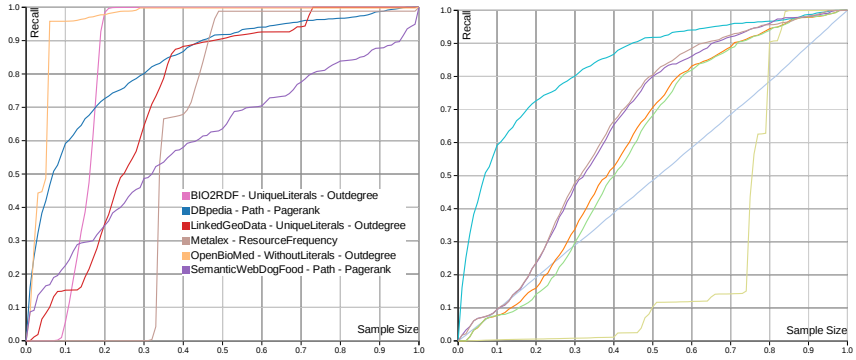
Baselines In our evaluation we use two baselines: a *random selection* (rand) and using *resource frequency* (freq). Random selection is based on 10 random samples for each of our datasets. Then, for each corresponding query we calculate recall using the 10 sampled graphs, and average the recall over each query. The resource frequency baseline counts the number of occurrences of every subject, predicate and object present in a dataset. Each triple is then assigned a weight equal to the sum of the frequencies of its subject, predicate and object.

6 Results

This section discusses the results we obtained for each of the datasets. An interactive overview of these results, including recall plots, significance tests, and degree distributions for *every* dataset, is available online.[9] Figure 2a shows the best performing sample method for each of our datasets. An ideal situation would

[8] Queries with only OPTIONAL clauses are ignored in our evaluation

[9] See `http://data2semantics.github.io/GraphSampling/`



(a) Best performing sample methods per dataset

(b) PageRank (DBPedia)

(c) In degree (DBPedia)

(d) Out degree (DBPedia)

Fig. 2: Best samping methods per dataset (a) and comparison of methods for DBPedia (b,c,d)[12]

show a maximum recall for a low sample size.[10] For *all* the datasets, these best performing sampling methods outperform the random sample, with often a large difference in recall between both. The P rewrite method (see Figure 1) combined with a PageRank analysis performs best for Semantic Web Dog Food and DBpedia (see also Figure 2). The UL method combined with an out degree analysis performs best for Bio2RDF and Linked Geo Data. For Open-BioMed the WL and out degree performs best, where the naive resource frequency method performs best for MetaLex. For each dataset, the random baseline follows an (almost) linear line from recall 0 to 1; a stark difference with the sampling methods.

Figure 2a also shows that both the sample *quality* and *method* differs between datasets. Zooming in on sample sizes 10%, 25% and 50%, the majority of the best

[10] Note that all plots presented in this paper are clickeable, and point to the online interactive version

performing sampling methods have significantly better average recall ($\alpha = 0.05$) than the random sample. Exceptions are LGD and Bio2RDF for sample size 10%, and MetaLex for sample size 10% and 25%.

The dataset properties listed in Table 1 help explain results for some sampling methods. The resource frequency baseline performs extremely bad for OBM[9]: for all possible sample sizes, the recall is almost zero. Of all objects in OpenBioMed triples, 45.37% are literals. In combination with 32% duplicate literals, this results in high rankings for triples that contain literals for this particular baseline. However, *all* of the queries use at least one triple pattern consisting *only* of IRIs. As most dataset triples contain a literal, and as these triples are ranked high, the performance of this specific baseline is extremely bad.

Another observation is the presences of 'plateaus' in Figure 2a, and for some sample sizes a steep increase in recall. This is because some triples are required for answering a large number of queries. Only once that triple is included in a sample, the recall can suddenly rise to a much higher level. For the Bio2RDF sample created using PageRank and the path rewrite method (viewable online), the difference in recall between a sample size of 1% and 40% is extremely small. In other words, choosing a sample size of 1% will result in more or less the same sample quality as a sample size of 40%.

Figure 2 (b,c,d) shows the performance of the sampling methods for DBpedia. The P method combined with either PageRank or in degree performs best on DBpedia, where both baselines are amongst the worst performing sampling methods. A sample size of 7% based on the P and PageRank sampling method already results in an average recall of 0.5. Notably, this same rewrite method (P) performs worst on DBpedia when applied with out degree. This difference is caused by triples with literals acting as sink for the path rewrite method: because a literal can never occur in a subject position, that triple can never link to any other triple. This causes triples with literals to *always* receive an out degree of zero for the P rewrite method. Because 2/3 of the DBpedia queries require at least one literal, the average recall is extremely low. This 'sink' effect of P is stronger compared to other rewrite methods: the triple weight of these other methods is based on the weight of the subject and object (see section 4). For triples containing literals, the object will have an out degree of zero. However, the subject may have a larger out degree. As the subject and object weights are aggregated, these triples will often receive a non-zero triple weight, contrary to the P rewrite method.

Although our plots show striking differences between the datasets, there are similarities as well. First, the out degree combined with the UL, CL and S methods performs very similar across all datasets and sample sizes (See our online results[9]). The reason for this similarity is that these rewrite methods *only* differ in how they treat literals: as-is, unique, or concatenated with the predicate. These are exactly those nodes which *always* have an out degree of zero, as literals only have incoming edges. Therefore, this combination of rewrite methods and network analysis algorithms performs consistently the same. Second, the in degree of the S and CL rewrite methods are similar as well for all datasets with

only a slight deviation in information loss for DBpedia. The main idea behind the CL is appending the predicate to the literal to provide context. The similarity for the in degree of both rewrite methods might indicate only a small difference between the literals in both rewrite methods regarding incoming links: adding context to these literals has no added value. DBpedia is the only dataset with a difference between both rewrite methods. This makes sense, as this dataset has many distinct predicates (53.000), which increases the chances of a single literal being used in multiple contexts, something the CL rewrite method is designed for.

What do these similarities buy us? They provide rules of thumb for applying SampLD to new datasets, as it shows which combinations of rewrite methods and network analysis algorithms you can safely ignore, restricting the number of samples to create and analyze for each dataset.

7 Conclusion

This paper tests the hypothesis as to whether we can use uninformed, network topology based methods to estimate semantic relevance of triples in an RDF graph. We introduced a pipeline that uses network analysis techniques for scalable calculation and selection of ranked triples, *SampLD*. It can use five ways to rewrite labeled, directed graphs (RDF) to unlabeled directed graphs, and runs a parallelized network analysis (indegree, outdegree and PageRank). We furthermore implemented a method for determining the recall of queries against our samples that does not require us to load every sample in a triple store (a major bottleneck). As a result, SampLD allows us to evaluate 15.600 different combinations of datasets, rewritten graphs, network analysis and sample sizes.

RDF graph topology, query type and structure, sample size; each of these can influence the quality of samples produced by a combination of graph rewriting and network analysis. This paper does not offer a definitive answer as to which combination is the best fit: we cannot yet predict the best performing sampling method given a data and query set. To make this possible, we plan to use Machine Learning on the results for several more independent data and query sets. Although SampLD provides the technical means, the number of publicly available query sets is currently too limited to learn significant correlations (6 query sets in total for USEWOD 2013, only 3 in 2014)[11].

In other work [27,28] we propose a SPARQL client (YASGUI), of which its logs allows us to gather and analyze queries for a wider range of datasets. [28] shows that these queries are quite different from the ones we can find in the publicly available server logs that we had access to for the purposes of this paper. In future work, we will use these query features in abstract, intermediate representations of queries and query sets. This allows us to better understand the effect of query types on our sampling methods, and forms the basis for sample

[11] See `http://data.semanticweb.org/usewod/2013/challenge.html` and `http://usewod.org/reader/release-of-the-2014-usewod-log-data-set.html`, respectively

quality prediction: a prerequisite for more informed sampling methods. Finally, a deeper analysis of the queries will also reveal information about the dynamics of Linked Data usage: does the part of the data touched by queries remain stable over time, or is it very volatile? In other words, is the 2% coverage from Table 1 predictive for future situations?

Our results, all of which are available online[12], indicate that the topology of RDF graphs *can* be used to determine good samples that, in many cases, significantly outperform our baselines. Indeed, this shows that we can mimic *semantic* relevance through *structural* properties of RDF graphs, without an a-priori notion of relevance.

References

1. Angles Rojas, R., Minh Duc, P., Boncz, P.A.: Benchmarking Linked Open Data Management Systems. ERCIM News 96, 24 – 25 (January 2014)
2. Anyanwu, K., Maduko, A., Sheth, A.: SemRank: ranking complex relationship search results on the semantic web. In: Proceedings of the 14th international conference on WWW. pp. 117–127. ACM (2005)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735 (2007)
4. Auer, S., Demter, J., Martin, M., Lehmann, J.: Lodstats–an extensible framework for high-performance dataset analytics. In: Knowledge Engineering and Knowledge Management, pp. 353–362. Springer (2012)
5. Auer, S., Lehmann, J., Hellmann, S.: Linkedgeodata: Adding a spatial dimension to the web of data. In: The Semantic Web-ISWC 2009, pp. 731–746. Springer (2009)
6. Avery, C.: Giraph: Large-scale graph processing infrastructure on hadoop. Proceedings of the Hadoop Summit. Santa Clara (2011)
7. Balmin, A., Hristidis, V., Papakonstantinou, Y.: Objectrank: Authority-based keyword search in databases. In: VLDB. pp. 564–575 (2004)
8. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2rdf: towards a mashup to build bioinformatics knowledge systems. Journal of biomedical informatics 41(5), 706–716 (2008)
9. Berendt, B., Hollink, L., Luczak-Rösch, M., Möller, K., Vallet, D.: Usewod2013 3rd international workshop on usage analysis and the web of data. In: 10th ESWC - Semantics and Big Data, Montpellier, France (2013)
10. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: Sparql web-querying infrastructure: Ready for action? In: The Semantic Web–ISWC 2013, pp. 277–293. Springer (2013)
11. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello, G.: Introducing RDF Graph Summary with application to Assisted SPARQL Formulation. In: 23rd International Workshop on Database and Expert Systems Applications (2012)
12. Franz, T., Schultz, A., Sizov, S., Staab, S.: Triplerank: Ranking semantic web data by tensor decomposition. The Semantic Web-ISWC 2009 pp. 213–228 (2009)
13. Gates, A.F., et al.: Building a high-level dataflow system on top of map-reduce: the pig experience. Proceedings of the VLDB Endowment 2(2), 1414–1425 (2009)

[12] Interactive plots of our results are available at `http://data2semantics.github.io/GraphSampling/`.

14. Görlitz, O., Thimm, M., Staab, S.: Splodge: systematic generation of sparql benchmark queries for linked open data. In: The Semantic Web–ISWC 2012, pp. 116–132. Springer (2012)
15. Gottron, T., Pickhardt, R.: A detailed analysis of the quality of stream-based schema construction on linked open data. In: Semantic Web and Web Science, pp. 89–102. Springer (2013)
16. Guéret, C., Wang, S., Groth, P., Schlobach, S.: Multi-scale analysis of the web of data: A challenge to the complex system's community. Advances in Complex Systems 14(04), 587 (2011)
17. Halaschek, C., Aleman-meza, B., Arpinar, I.B., Sheth, A.P.: Discovering and ranking semantic associations over a large rdf metabase. In: VLDB (2004)
18. Hayes, J., Gutierrez, C.: Bipartite Graphs as Intermediate Model for RDF. In: International Semantic Web Conference. pp. 47–61 (2004)
19. Hoekstra, R.: The MetaLex Document Server - Legal Documents as Versioned Linked Data. In: Alani, H., Tailor, J. (eds.) Proceedings of the 10th International Semantic Web Conference (ISWC 2011). p. 16. Springer (2011)
20. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: 2nd Workshop on Scalable Semantic Web Knowledge Base Systems (2006)
21. Kanehisa, M., et al.: From genomics to chemical genomics: new developments in kegg. Nucleic acids research 34(suppl 1), D354–D357 (2006)
22. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46(5), 604–632 (1999)
23. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: The 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 631–636 (2006)
24. Möller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for semantic web dog food. the eswc and iswc metadata projects. In: The Semantic Web, pp. 802–815. Springer (2007)
25. Pérez, J., Arenas, M., Gutierrez, C.: Semantics of SPARQL. In: Technical Report TR/DCC-2006-17, Department of Computer Science, Universidad de Chile (2006)
26. Picalausa, F., Vansummeren, S.: What are real sparql queries like? In: International Workshop on Semantic Web Information Management. p. 7. ACM (2011)
27. Rietveld, L., Hoekstra, R.: YASGUI : Not Just Another SPARQL Client. In: Proceedings of the ESWC2013, SALAD 2013 (2013)
28. Rietveld, L., Hoekstra, R.: Man vs. Machine: Differences in SPARQL Queries. In: ESWC, 4th USEWOD Workshop on Usage Analysis and the Web of Data (2014)
29. Schmidt, M., Hornung, T., Meier, M., Pinkel, C., Lausen, G.: Sp2bench: A sparql performance benchmark. In: Semantic Web Information Management, pp. 371–393. Springer (2010)
30. Sundara, S., et al.: Visualizing large-scale rdf data using subsets, summaries, and sampling in oracle. In: Data Engineering (ICDE), 2010 IEEE 26th International Conference on. pp. 1048–1059. IEEE (2010)
31. Tan, G., Tu, D., Sun, N.: A parallel algorithm for computing betweenness centrality. In: Proc. of ICPP. pp. 340–347 (2009)
32. Tonon, A., Catasta, M.: TRank: Ranking Entity Types Using the Web of Data. In: The Semantic Web - ISWC. pp. 640 – 656 (2013)
33. Wang, S., Groth, P.: Measuring the dynamic bi-directional influence between content and social networks. In: The 9th International Semantic Web Conference ISWC 2010. pp. 814–829. Springer (2010)