Keeping Linked Open Data Caches Up-to-date by Predicting the Life-time of RDF Triples

Chifumi Nishioka*
Kyoto University Library
Kyoto, Japan
Germany
nishioka.chifumi.2c@kyoto-u.ac.jp

Ansgar Scherp
Kiel University and ZBW – Leibniz Information Centre for Economics
Kiel, Germany
asc@informatik.uni-kiel.de

ABSTRACT

Many Linked Open Data applications require fresh copies of RDF data at their local repositories. Since RDF documents constantly change and those changes are not automatically propagated to the LOD applications, it is important to regularly visit the RDF documents to refresh the local copies and keep them up-to-date. For this purpose, crawling strategies determine which RDF documents should be preferentially fetched. Traditional crawling strategies rely only on how an RDF document has been modified in the past. In contrast, we predict on the triple level whether a change will occur in the future. We use the weekly snapshots of the DyLDO dataset as well as the monthly snapshots of the Wikidata dataset. First, we conduct an in-depth analysis of the life span of triples in RDF documents. Through the analysis, we identify which triples are stable and which are ephemeral. We introduce different features based on the triples and apply a simple but effective linear regression model. Second, we propose a novel crawling strategy based on the linear regression model. We conduct two experimental setups where we vary the amount of available bandwidth as well as iteratively observe the quality of the local copies over time. The results demonstrate that the novel crawling strategy outperforms the state of the art in both setups.

CCS CONCEPTS

• **Information systems → Web crawling**; **Resource Description Framework (RDF)**; *Data management systems*; *Temporal data*;

ACM Reference format:
Chifumi Nishioka and Ansgar Scherp. 2017. Keeping Linked Open Data Caches Up-to-date by Predicting the Life-time of RDF Triples. In *Proceedings of WI '17, Leipzig, Germany, August 23-26, 2017,* 8 pages.
https://doi.org/10.1145/3106426.3106463

*This work was done when the author studied at Kiel University and ZBW – Leibniz Information Centre for Economics.

WI '17, August 23-26, 2017, Leipzig, Germany

ACM ISBN 978-1-4503-4951-2/17/08…$15.00
https://doi.org/10.1145/3106426.3106463

1 INTRODUCTION

Since its advent in 2007, the Linked Open Data (LOD) cloud has been continuously evolving [14]. The LOD cloud is composed of entities (e .g., persons, organizations) and relations between them. The relations are represented as RDF triples consisting of a subject, predicate, and object. RDF triples are defined in an RDF document. Thus, an RDF document can be seen as a set of RDF triples. The data on the LOD cloud is covering a wide range of domains and is consumed by various applications [1, 15]. Therefore, understanding the temporal dynamics of the LOD cloud is not trivial while at the same time it is highly relevant in applications such as data caching [18] and caching of SPARQL queries [10, 21]. So far, only a few works have investigated the temporal dynamics of the LOD cloud, particularly on the triple-level. Käfer et al. [8] quantified changes with respect to a set of triples, set of links, and schema signature. They identified that most dynamic predicates are about trivial time stamps. Dividino et al. [2] measured changes with respect to the aggregation of RDF documents stemming from a common pay-level domain (PLD). A PLD is the part of an URI that is typically registered and payed for by organizations or individual persons. The authors developed a metric to quantify temporal dynamics of LOD sources and applied the metric to efficiently crawl LOD sources. Furthermore, we determined temporal patterns in terms of the changes of the entities (i .e., resources, instances) on the LOD cloud [11]. In summary, to the best of our knowledge only few works have conducted an analysis of temporal dynamics focusing on individual triples such as Käfer et al. [8].

In this paper, we propose a novel crawling strategy for RDF documents based on the predicted life span of triples in the RDF documents. We assume that predicting the temporal dynamics on the level of the atomic units (i .e., triples) provides more fine-grained insights and enables to better predict the dynamics of, e. g., RDF documents. As datasets, we use the Dynamic Linked Data Observatory (DyLDO) [9] as well as Wikidata [6]. The DyLDO dataset is a collection of weekly snapshots of various LOD sources over three years. Wikidata is one of the largest cross-domain knowledge graphs. The dataset contains almost monthly snapshots of Wikidata RDF exports. In both datasets, we observe that triples can be divided into *ephemeral* and *stable*. Ephemeral triples are alive for a short period. On the contrary, stable triples live for a long period and appear in many or even all snapshots. We first conduct an analysis which attempts to understand which feature of triples has a large influence on the life span of triples. To this end, we introduce different features such as PLD of subject URI, predicate, as well as object form (i .e., URI or literal) and object PLD. Along with these features, we convert triples into feature vectors. Over the feature

vectors, we compute a linear regression model in order to identify features that have a large influence on the life span of the triples. We also experimented with other regression models such as logistic regression and the gradient boosted regression trees. Among them, the linear regression model has competitive predictive power and requires less computation costs.

Finally, we introduce a novel crawling strategy to effectively update local copies of RDF documents. A crawling strategy assigns a preference score, which determines the order of RDF documents for crawling. Dividino et al. [2] conducted an extensive evaluation of different crawling strategies for LOD sources using features such as their age and size. While Dividino et al. [2] conducted an experiment aggregating the RDF documents from PLDs, we use single RDF documents as it is more common to update the local copies with respect to individual RDF documents. An RDF document comes from a URI, which corresponds to what is commonly also called a "context" (URI). Dividino et al. concluded that the strategy based on data dynamics is the best one. In contrast to the existing methods, our novel strategy gives a preference score based on the life span of individual triples included in an RDF document. Following Dividino et al. [2], we evaluate our novel strategy in two setups: In the first setup, we vary the percentage of available bandwidth and thus control the amount of triples that can be updated by a crawler. Second, we apply the crawling strategy from Dividino et al. [2] as baseline and compare it with our own approach in a setup where we iteratively update the local copies of RDF documents over 20 consecutive weeks. The results show that our novel strategy outperforms the state of the art of Dividino et al. [2] in terms of both precision and recall in both setups.

The remainder of the paper is organized as follows: The subsequent section reviews related work. We introduce formalizations and the datasets in Sections 3 and 4. Section 5 introduces our linear regression model for predicting the temporal dynamics (i .e., life span) of triples. Based on the regression model, we formulate a novel crawling strategy for RDF documents and empirically evaluate it in Section 6, before we conclude the paper.

2 RELATED WORK

We first introduce works investigating the temporal dynamics of RDF data on the LOD cloud. Subsequently, we review crawling strategies for web pages and RDF documents.

Dividino et al. [4] measured the degree of how often the `last-modified` field in HTTP header of RDF documents is available and how often it is correctly used. The analysis revealed that on average only 7% of the documents provide accurate information for the `last-modified` field. Therefore, the `last-modified` field cannot be used for the temporal analysis and crawling strategies. Furthermore, Dividino et al. [2] evaluated crawling strategies for LOD sources with limited bandwidth. Crawling strategies define which LOD source should be preferentially visited at a given point in time. In their previous work [3], they proposed a monotone and positive function to represent the dynamics as single numerical value and applied it to the crawling strategies. The experiment revealed that strategies based on dynamics of LOD sources [3] performed best, compared to strategies based on the data source's age, PageRank, or size. Käfer et al. [8] provided a comprehensive analysis of the temporal dynamics of the LOD cloud based on monitoring 86,696 RDF documents for 29 weeks. They found out that 5.0% of documents had gone offline and 62.2% of documents had no change. In addition, they conducted the analysis focusing on triples. They observed that object literals are the most dynamic element of triples. Predicates (i .e., properties) and RDF types defined by the predicate `rdf:type` are static. Furthermore, the authors identified that the most dynamic predicates were often about trivial time stamps.

Crawling strategies fetch and store documents as local copies. In the past, many strategies for web pages have been developed and investigated. The works aim at increasing the coverage of documents (i .e., finding new documents) and/or maintaining fresh copies of data. In this paper, we assume that we have a list of the documents that we want to crawl. Thus, we do not tackle the problem of finding new RDF documents. Edwards et al. [5] argued that the crawling frequency should be increased for frequently-modified web pages. Tan et al. [16] developed a clustering-based incremental crawling strategy. In contrast to the strategy by Edwards et al. [5], their strategy exploits the content of web pages. In addition, the strategy does not require gathering a long history of the web pages before it can start crawling the pages. They first cluster web pages based on features that correlate to their change frequencies. At each point in time, they crawl a few web pages in each cluster. Only if the crawled web pages of a cluster have many changes, they download and update all the web pages in the cluster. In terms of features employed for clustering, they use static features such as content features (e .g., words in texts, the number of images), URL features (e .g., name of the top level domain), and linkage features (e .g., the number of incoming links). In addition, they exploit dynamic features which calculate how much each content feature and linkage feature have changed in the latest two successive snapshots. Their experiment revealed that the combined features (i .e., content features + dynamic features) are best for the crawling strategies. Furthermore, Radinsky et al. [13] showed that content of web pages as well as their related web pages significantly improve prediction performance of web page modifications. In terms of the crawling strategies for RDF documents, the strategy based on the LOD source dynamics proposed by Dividino et al. [2] is the state of the art. Thus, we use this strategy as a baseline in the experiment shown in Section 6.

3 BASIC FORMALIZATION

We introduce definitions and notations. Tables 1 and 2 show symbol notations and a small example of LOD snapshots, respectively.

Data fetched from the LOD cloud is represented in the form of N-Triples[1]. A triple is represented as (s, p, o) where s, p, and o correspond to a subject, predicate, and object, respectively. Furthermore, we define the sets of all possible URIs U and literals L. The subject $s \in U$ is a URI, the predicate $p \in U$ a URI, and the object $o \in U \cup L$ a URI or a literal. Furthermore, we define X_t as a set of triples (i .e., snapshot) captured at a point in time t and $\mathbb{X} = \{X_{t_1}, X_{t_2}, \ldots, X_{t_n}\}$ as collection of the snapshots at the different points in time. Using the example in Table 2, X_{t_1} contains five triples and each of X_{t_2} and X_{t_3} has eight triples. In the experiment of crawling strategies shown in Section 6, we take into account contexts c, indicating from where a triple is fetched [7]. Contexts are also referred as RDF

[1] https://www.w3.org/TR/n-triples/, last accessed on 03/04/2017

Table 1: Symbol Notation

t	a point in time
$x = (s, p, o)$	triple
s, p, o	subject, predicate, object
c	context indicating from where a triple is fetched
X_t	snapshot i .e., set of triples captured at t
X'_t	set of triples in the local copy at t
$X_{t,c}$	set of triples captured at t from c
$\mathbb{X}$	set of all snapshots X_t

documents. Extending X_t, $X_{t,c}$ denotes a set of triples which are contained in an RDF document c at a point in time t. In addition, X'_t represents a set of triples in the local copy at a point in time t.

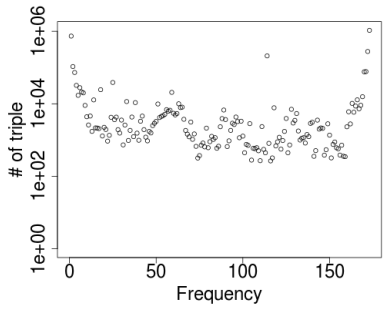
Table 2: An example of three snapshots of the LOD cloud.

X_{t_1}: a snapshot at time t_1		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
db:Green_Village	db:population	224123
X_{t_2}: a snapshot at time t_2		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
uni:John_Brown	db:location	db:Green_Village
uni:John_Brown	db:works	db:Green_Institute
db:Green_Village	db:population	223768
X_{t_3}: a snapshot at time t_3		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
uni:John_Brown	db:location	db:Green_Village
uni:John_Brown	db:works	db:Green_University
db:Green_Village	db:population	223540

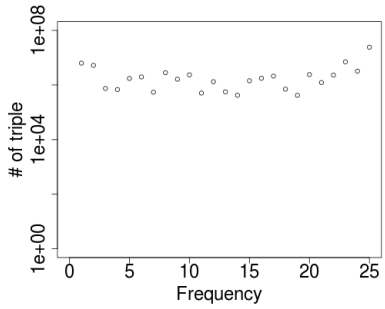
4 DATASETS

DyLDO. As first dataset, we use the Dynamic Linked Data Observatory (DyLDO) dataset[2] [8, 9]. It has been created to monitor a fixed set of RDF documents on a weekly basis. The dataset is composed of 173 weekly snapshots from 11/27/2012 to 03/27/2016 and covers various well known LOD sources as well as less commonly known ones [9]. From the original dataset, we first identify RDF documents that were crawled at every snapshot by analyzing the access logs. We extract all unique triples that are contained at one of the identified RDF documents. Then, we filter out triples that contain blank nodes, since blank nodes may have different identifiers in different snapshots. In order to conduct an analysis on the level of triples, we extract all unique triples from all the snapshots $\mathbb{X}$. In total, all the snapshots contain 3, 271, 944 unique triples. For each unique triple, we count its frequency, i .e., in how many snapshots the triple appears. We interpret the frequency of triples as their availability. The maximum and minimum frequency are 173 and 1, respectively. Figure 1(a) shows the distribution of triple frequencies. We can see that there are triples along the full range from very ephemeral to very stable. On average, each triple is alive in 99.29 snapshots (SD: 77.44) over the entire period from 2012 to 2016.

Wikidata. Second, we use Wikidata [20] as one of the largest cross-domain knowledge graphs. We obtain the snapshots from the Wikidata RDF exports[3], where the data are converted into N-triples [6]. In the experiment of the crawling strategies, we consider triples that share a common subject URI as one RDF document. We use 25 snapshots of Wikidata from 04/20/2014 to 08/01/2016, which are captured almost monthly. In total, all the snapshots contain 73, 583, 940 unique triples. For each unique triple, we count its frequency, i .e., in how many snapshots the triple appears. The maximum and minimum frequency are 25 and 1, respectively. Figure 1(b) shows the distribution of triple frequencies. On average, each triple is alive in 16.51 snapshots (SD: 9.14).



(a) DyLDO dataset



(b) Wikidata dataset

Figure 1: Distribution of frequencies regarding the availability of unique triples in the two datasets.

5 PREDICTING THE LIFE SPAN OF TRIPLES BY LINEAR REGRESSION

We investigate which features on the triples determine their life span. Specifically, we train a linear regression model to provide insights on which features have a large influence on the life span on certain triples.

[2] http://swse.deri.org/dyldo/, last accessed on 03/04/2017

[3] `wikidata-simple-statements.nt.gz` from each directory on https://tools.wmflabs.org/wikidata-exports/rdf/exports/

Table 3: Descriptive statistics of the datasets. The table provides the number of snapshots, the number of RDF documents, the number of unique triples in the entire dataset, the average frequency of triples, and the average number of triples per snapshot. The standard deviation is given in parentheses.

	# snapshots	# RDF documents	# unique triples	average frequency of triples	average # triples per snapshot
DyLDO	173	11,917	3,271,944	99.29 (77.44)	1,877,875.82 (76,203.44)
Wikidata	25	9,753,532	73,583,940	16.51 (9.14)	48,609,241.20 (7,500,579.23)

5.1 Features Computed on the Triples

We examine the following three features which are the subject PLD, predicate, and object form and PLD of the triples. The features are motivated by the works [13, 16] that predict future changes of web pages by looking into their content. Therefore, we assume that it is possible to predict changes (i .e., life span) of triples by analyzing their subjects, predicates, and objects.

Subject PLD Subjects are defined by a URI. From this subject URI, we use the pay level domain (PLD) as feature. For instance, if a subject URI is `http://dbpedia.org/resource/Facebook`, the subject PLD is `http://dbpedia.org`. We motivate this feature from Umbrich et al. [19] which showed that entities coming from the same PLD show similar temporal dynamics. The PLD of a URI is extracted using Guava[4]. If Guava identifies no PLD, the placeholder "other subject PLDs" is assigned.

Predicate Triples that have a common predicate may have similar temporal behavior. For instance, a triple whose predicate is `dbo:areaLand` is assumed static, because an area of places such as countries do not change frequently. In contrast, as the statistics of the population is updated, a triple whose predicate is `dbo:populationTotal` disappears and a new triple whose predicate is `dbo:populationTotal` appears. Thus, a triple with the predicate `dbo:populationTotal` can be assumed to be more ephemeral.

Object form and PLD Objects are either a literal or a URI. If an object is a literal, the feature takes the value "literal". Otherwise, it is assigned the PLD of the object URI as we do for the subject PLD.

Please note that we have also investigated topological features such as degrees of subject URI and object URI, which were used to predict ontology changes [12]. However, the prediction performance using these features is similar to the baseline. Therefore, we omit them.

In terms of the DyLDO dataset, we extract 1, 706 subject PLDs and 3, 295 predicates from all unique triples. In 1, 573, 797 (48.10%) triples, the object is defined by a literal. There are 3, 059 object PLDs in triples whose object is defined by a URI. Regarding the Wikidata dataset, there is only one unique subject PLD. Thus, the feature of the subject PLD is ignored in the Wikidata dataset. We find 2, 204 predicates. In terms of the object, 19, 291, 060 (26.22%) triples are defined by a literal. There are 239, 405 object PLDs in triples whose object is defined by a URI.

[4]https://github.com/google/guava/wiki/Release19, last accessed on 03/31/2017

5.2 Computing the Linear Regression Model

We train a linear regression model to predict the frequencies (i .e., life span) of triples from the triple features. The linear regression model is defined as below:

$$\hat{y} = w_0 a_0 + w_1 a_1 + w_2 a_2 + \cdots + w_d a_d \quad (1)$$

In Equation 1, w_i denotes a weight of a feature. w_0 equals to the intercept of the model. a is a feature value and d stands for the number of features in the model. $\hat{y}$ is a predicted value. Please note, we distinguish an actual value and a predicted value using y and $\hat{y}$, respectively. Since we aim to predict the frequencies of triples, y is a value between 1 and 173 for DyLDO and 1 to 25 for Wikidata, respectively. Although a_0 is originally not contained in the equation, we introduce it as a constant $a_0 = 1$ to ease the notation. In short, Equation 1 is represented as $\hat{y} = \boldsymbol{w}^T \boldsymbol{a}$. $\boldsymbol{w}$ and $\boldsymbol{a}$ denote weights of features and feature values, which are $d + 1$ dimensional vectors.

Subsequently, we introduce how to construct feature vectors $\boldsymbol{a}$. Since all the triple features are nominal data, we convert triples into feature vectors by one-hot encoding. Using an example in Table 2, there are two unique subject PLDs, three unique predicates, and two object forms (i .e., one is literal and the other is a PLD "db:"). It results in a seven dimensional vector, where the first and second element show "db:" and "uni:" (subject PLDs) followed by "db:location", "db:works", "db:population" (predicates), followed by "literal" and "db:" (object form and PLDs). A triple ⟨db:Anne_Smith, db:location, db:Green Village⟩ is converted in (1, 0, 1, 0, 0, 0, 1). Since there are many subject PLDs, predicates, and object form and PLDs that are used only by a few triples, we integrate them into one feature, namely "other subject PLDs", "other predicates", and "other object PLDs" to reduce the dimension of feature vectors. Specifically, we merge subject PLDs, predicates, and object PLDs which are used by 10 or less triples into "other subject PLDs", "other predicates", and "other object PLDs", respectively. The triple features used by more than 10 triples cover over 99% of unique triples, because the frequencies of subject PLDs, predicates, and object form and PLDs follow the power-law distribution. This power-law distribution is also shown by Tummarello et al. [17].

5.3 Training the Linear Regression Model

First, we describe the training of the linear regression model. Subsequently, we evaluate how the resulted model can predict the frequencies of triples.

5.3.1 Training Model. We train the linear regression model using the features introduced in Section 5.1. The number of dimensions is $d = 2,613$ (i .e., the joint of 705 subject PLDs, 1, 335 predicates, and 573 object form and PLDs) for the DyLDO dataset and $d = 2,720$ (i .e., the joint of 1 subject PLDs, 1, 739 predicates, and

Table 4: Performance of the prediction of triple frequencies.

Prediction model	DyLDO		Wikidata	
	RMSE	MAE	RMSE	MAE
Mean average (baseline)	77.36	73.54	8.95	7.77
Linear regression model: subject PLD	44.87	27.55	NA	NA
Linear regression model: predicate	42.72	26.39	5.72	3.82
Linear regression model: object form and PLD	65.22	53.63	7.98	6.72
Linear regression model: all triple features	30.77	15.47	5.16	3.24

980 object form and PLDs) for the Wikidata dataset. Before running the linear regression, we split the data into training data and test data. We randomly pick 90% of triples as training data and the rest of the data is used for test in each dataset.

To avoid overfitting, we use l2 regularization that penalizes models with extreme parameter values. Thus, the function being optimized is:

$$\min_{w} \sum_{i=0}^{n} (\boldsymbol{w}^T \boldsymbol{a}_i - y_i)^2 + \lambda \cdot ||\boldsymbol{w}||_2^2 \quad (2)$$

The first term shows the residual squared sum (RSS) which is employed as a loss function. The second term is the regularization term which mitigates overfitting. We optimize the parameter $\lambda = 316.23$ by 10-fold cross-validation on the training data. We use the implementation of the linear regression provided by GraphLab Create[5].

5.3.2 Resulted Model. We start from the resulted model of the DyLDO dataset. In terms of subject PLD features, `ranselrazer.nl`, `fotolog.net`, and `blip.fm` have the largest weights. `today.com` and `nbcnews.com`, which provides news information, have the smallest weights. Regarding the predicate features, `http://edgarwrap.ontologycentral.com/vocab/edgar\#issued` has the largest weight. In contrast, the predicate `http://www.w3.org/ns/auth/rsa\#public\_exponent` has the smallest weight. It is used to note an exponent to encrypt a message. Since such exponents are frequently updated, triples with this predicate are alive only for a short period.

5.3.3 Prediction Power. Using the resulted linear regression model, we predict the frequencies of triples with the test data. As evaluation measures, we employ rooted mean squared error $RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2}$ and mean absolute error $MAE = \frac{1}{M}\sum_{i=1}^{M}|y_i - \hat{y}_i|$, where M denotes the number of data points in the test data. In both measures, lower values indicate better performance. RMSE indicates how well the predicted values fit to the linear regression model. MAE shows the prediction power, i .e., how close the predicted values are to the resulted values. In order to demonstrate the effectiveness of the linear regression model, we compare it with a baseline providing the mean average of frequencies in the training data to all triples in the test data as a prediction.

In addition, we also train the linear regression model using only subject PLD, predicates, and object form and PLD, respectively. Thus, we show which triple feature is most powerful as well as how well the linear regression model works when using all the three triple features. As shown in Table 4, the linear regression model significantly outperforms the baseline. MAE of the linear regression model with all triple features is 15.47 in the DyLDO dataset and 3.24 in the Wikidata dataset. It indicates that the model predicts the life span with about 5% error rate. Therefore, it is possible to predict the life span of the triples by looking at the features of the triples. In addition, the linear regression model with all features outperforms the one that are solely computed on the features of subject PLD, predicate, and object form and PLD. Thus, all triple features have a positive influence on predicting the life span.

[5]https://turi.com/products/create/docs/generated/graphlab.linear_regression.LinearRegression.html, last accessed on 03/31/2017

6 CRAWLING STRATEGIES FOR RDF DOCUMENTS BASED ON PREDICTED LIFE SPAN OF TRIPLES

This section shows how the prediction of the life span of triples provides a better crawling strategy for RDF documents compared to the state of the art. In many cases, LOD applications pre-fetch data from the LOD cloud and store them as local copies, in order to speed up access to the data. As the LOD cloud is continuously changing, local copies are quickly outdated and no longer reflect the current state. Thus, it is important for the LOD applications to update the local copies. In an ideal setting, the local copies are always up-to-date by continuously visiting all the RDF documents. But, due to limited network bandwidth or frequent changes on the RDF documents, this is difficult to be achieved. Therefore, it is indispensable to choose an efficient crawling strategy to update the local copies. Crawling strategies assign a preference score to each RDF document and determine which one should be preferentially updated. Dividino et al. [2] conducted an extensive evaluation to find out the most effective crawling strategy. They found out that the strategy based on the dynamics of the LOD sources outperforms other strategies. Motivated by the analysis shown in Section 5, we propose a crawling strategy which gives preference scores to RDF documents as calculated by the linear regression model.

Below, we first describe the crawling strategies including the state of the art as well as the proposed novel strategy based on the trained linear regression model. Subsequently, we explain the details of the experiment and report the results.

6.1 Crawling Strategies

We first formalize the problem of the crawling strategies, following Dividino et al. [2]. A crawling strategy determines which RDF documents $c \in C$ should be visited in the time slice between t_i and t_{i+1}. Specifically, it assigns a preference score $f(c, t_i)$ to each RDF document c at a point in time t_i. RDF documents that have

larger preference scores are first visited. In addition, we define the limitation of the bandwidth as a restriction to download at most up to K triples at each point in time. Thus, if K triples are fetched from RDF documents into the local copies, the updating process is stopped until the subsequent point in time.

Below, we introduce two different crawling strategies which output preference scores, namely our triple linear regression model from Section 5 and the baseline LOD source dynamics [2].

Triple Linear Regression. Our novel strategy provides preference scores to each source, taking into account triple features contained in the RDF document. The preference score is computed as below:

$$f_{\text{triple}}(c,t) = \left(\frac{1}{|X_{c,t}|}\sum_{x \in X_{c,t}} LR(x)\right)^{-1} \tag{3}$$

The function $LR(x)$ returns a triple's life span based on the linear regression model as shown in Section 5. We compute the mean average life span by averaging over $LR(x)$ for all triples x in a data source c. Finally, a preference score is defined by the reciprocal of the mean average. We take the reciprocal because of the following reason: According to Dividino et al. [2], crawling strategies should visit RDF documents starting from those with larger scores. However, the RDF documents with triples whose life span is short should be preferentially visited, since they contain more dynamic triples. Thus, we take the reciprocal as output, in order to reverse the order of the sources.

LOD Source Dynamics. Dividino et al. [2] reported that the LOD source dynamics strategy performed best. Thus, we employ it as baseline. The strategy assigns preference scores considering history about how many triples in a source have been updated in the past. The preference score is computed as below:

$$f_{\text{LODsrcdy}}(c,t_i) = \sum_{i=0} \frac{\delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}})}{t_{lu(c,i)} - t_{lu(c,lu(c,i)-1)}} \tag{4}$$

$lu(c,i)$ is a function that returns a point in time at which the given RDF document c was last updated at point in time i. Thus, $lu(c,i) \leq i$. This function can be used recursively. For example, the update prior to the last update is represented as $t_{lu(c,lu(c,i)-1)}$. δ is a function which returns the amount of changes between two snapshots. Although Dividino et al. [2] compared two variants of δ based on Jaccard distance and Dice coefficient, we only apply Jaccard distance as shown in Equation 5. We choose the Jaccard distance, because the results of the two variants are very similar and the variant based on Jaccard distance slightly outperforms the other.

$$\delta(X_{c,t_1}, X_{c,t_2}) = 1 - \frac{|X_{c,t_1} \cap X_{c,t_2}|}{|X_{c,t_1} \cup X_{c,t_2}|} \tag{5}$$

Please note that we refer to c as URI of an RDF document, while Dividino et al. [2] integrated the RDF documents with a common PLD into one document. We use single RDF documents as sources, because it is more common to update the local copies with respect to individual RDF documents rather than entire PLDs.

6.2 Experiment

This section describes our experimental procedure, datasets, as well as applied evaluation metrics.

6.2.1 Experimental Procedure. In the experiment, we compare the two crawling strategies in two setups, namely single-step and iterative progression, following Dividino et al. [2].

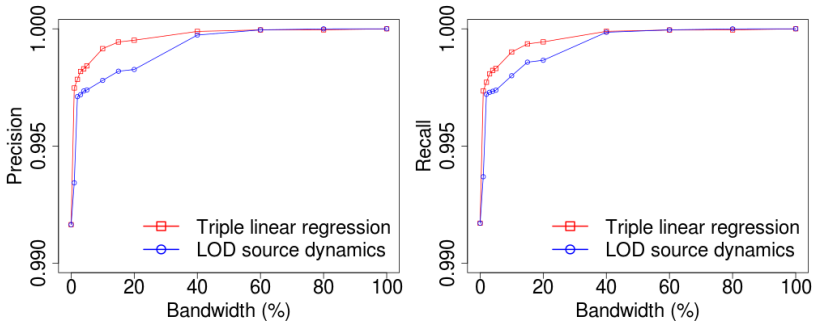
Single-Step We evaluate the performance of the crawling strategies for a single update of a local copy. We start from a perfect copy at a point in time t_i and compare the quality of the local copy at a point in time t_{i+1} achieved by the different crawling strategies.

Iterative Progression We evaluate the evolution of the quality of the local copies when considering iterative updates over a longer period of time. Starting from a perfect copy at a point in time t_i, we aim to measure how well different crawling strategies perform in terms of maintaining an accurate local copy at subsequent points in time $t_{i+1}, t_{i+2} \cdots t_{i+n}$. We evaluate local copies obtained at subsequent 20 points in time (approximately five months) for the DyLDO dataset and 4 points in time for the Wikidata dataset. Thus, $n = 20$ for the DyLDO dataset and $n = 4$ for the Wikidata dataset.

Furthermore, we simulate network constraints by limiting the relative bandwidth κ in the two setups. Specifically, we stepwise increase the bandwidth κ from 0% to 5% in steps of 1%, from 5% to 20% in steps of 5%, and from 20% to 100% in steps of 20%. Therefore, the number of triples that can be fetched at a point in time t is calculated as $K = \kappa \cdot |X_t|$.

6.2.2 Datasets. We use the DyLDO dataset as well as the Wikidata dataset as described in Section 4. We compute the LOD source dynamics and the linear regression model based on available history information. In both setups, the history is composed of the last 50 snapshots for the DyLDO dataset and 8 snapshots for the Wikidata dataset, respectively. Thus, the length of the history is almost one year in both datasets. To this end, we experiment starting from t = `2013-11-24` for the DyLDO dataset and t = `2015-05-11` for the Wikidata dataset. In the single-step setup, we slide the starting point t_i by one point in time. For the iterative progression setup, the starting point t_i is every tenth snapshot for the DyLDO dataset and every second snapshot for the Wikidata dataset.

Referring to the LOD source dynamics, we compute the preference scores at point in time t_i, looking into the last 50 snapshots for DyLDO and 8 for Wikidata. Again, the initial history information is the same in both setups. The preference scores of the RDF documents are continuously updated in the iterative progression setup, as the RDF document is fetched from the original RDF documents as shown in Equation 4. Thus, the size of the history information increases along with the iterations in the iterative progression setup. In terms of the linear regression model, we train the model over the first 50 of the 173 snapshots for DyLDO and 8 snapshots of the 25 snapshots for Wikidata. We use the same linear regression models to all points in time and do not update the linear regression model, in order to demonstrate its generalizability over time. To construct the linear regression model, we first extract all unique triples in the first 50 snapshots and 8 snapshots of the respective datasets. Then, we count the frequency of subject PLDs and predicates, and



(a) Precision of DyLDO dataset

(b) Recall of DyLDO dataset

(c) Precision of Wikidata dataset

(d) Recall of Wikidata dataset

Figure 2: Single-step setup: Precision (left) and recall (right) of the local copies.

object PLDs as we did in Section 4. Thereafter, we only take the features into account which are used by 10 or more unique triples and integrate all the others as described in Section 5.

6.2.3 Evaluation Metrics. We evaluate the local copies resulting from applying the crawling strategies by precision and recall, which are defined as: $precision(X'_t, X_t) = \frac{|X'_t \cap X_t|}{|X'_t|}$ and $recall(X'_t, X_t) = \frac{|X'_t \cap X_t|}{|X_t|}$. Again, X'_t denotes the resulted local copy of the RDF documents at a point in time t. X_t is the snapshot, i.e., a perfect up-to-date local copy, at a point in time t, which is considered as ground truth.

6.3 Results

We report the results of the experiment with respect to the two setups.

6.3.1 Single-Step. Figure 2 shows the precision and recall of the resulted local copies produced by the single-step setup when varying the relative bandwidth. The novel triple linear regression strategy overall outperforms the state of the art LOD source dynamics strategy in terms of both precision and recall. When the relative bandwidth is small ($\kappa < 5\%$), the difference between the two strategies is low. However the difference gets larger as the relative bandwidth increases. Finally, the difference between the strategies disappears with 100% bandwidth and both strategies achieve a precision of 1.00.

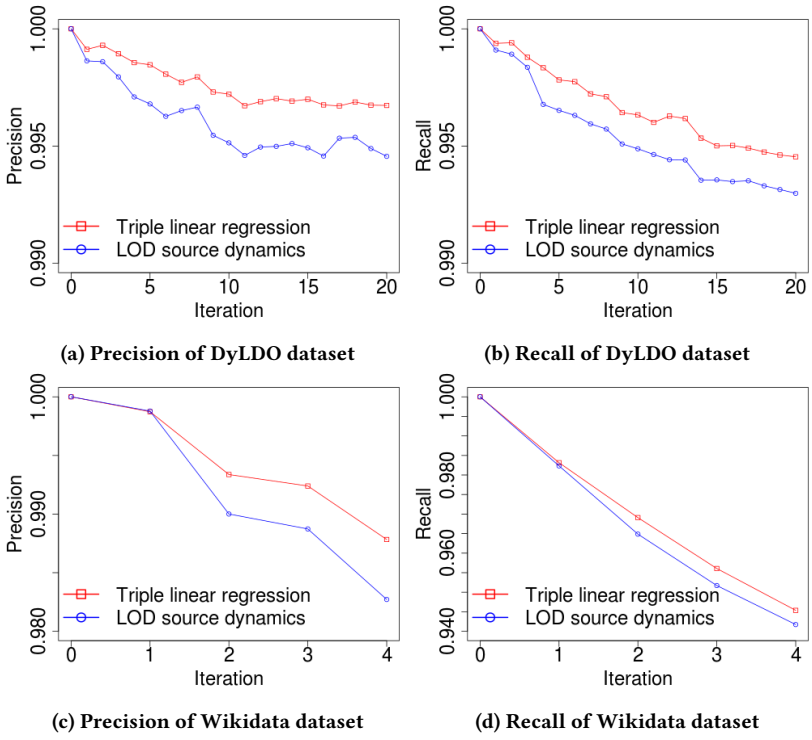
6.3.2 Iterative Progression. Figure 3 shows the result for the iterative progression setup when the relative bandwidth is $\kappa = 20\%$. The novel strategy always outperforms the other in terms of both precision and recall. Especially, the novel strategy is much better in terms of precision.

6.4 Discussion

In both setups, we observe that the novel strategy outperforms the state of the art. Especially, it is noteworthy that in the iterative progression setup the novel triple linear regression strategy performs better. While the LOD source dynamics updates the preference scores at every iteration, we do not update the regression model after each iteration. In summary, our novel strategy has the advantage that after the linear regression model is trained, it does not require additional past snapshots to compute preference scores. In contrast, the LOD source dynamics by Dividino et al. [2] requires to update the model after each iteration.

Furthermore, since the novel strategy looks into which triples are actually included in RDF documents, it captures the dynamics of the RDF documents in a more fine-grained way compared to the PLDs used by Dividino et al. [2]. In terms of the linear regression model, we conclude that the model is generalized, because the performance of the triple linear regression does not get worse while sliding over further points in time.

Please note, we use a linear regression model due to the simplicity of the model. We also tried logistic regression and others. But they lead to almost the same results.



(a) Precision of DyLDO dataset **(b) Recall of DyLDO dataset**

(c) Precision of Wikidata dataset **(d) Recall of Wikidata dataset**

Figure 3: Precision (left) and recall (right) of the resulted local copies in the iterative progression setup with a middle bandwidth ($\kappa = 20\%$).

7 CONCLUSION

We have presented a novel crawling strategy for RDF documents based on a linear regression model that exploits features of triples instead of entire RDF documents. We have introduced different features of triples, namely subject PLD, predicate, and object form and object PLD. The linear regression model revealed which features of triples have a large influence on the life span of triples. To this end, we proposed a novel scheduling strategy based on the predicted life span of triples. The experiment demonstrated the superiority of the novel strategy over the state of the art.

ACKNOWLEDGMENT

This work was supported by the EU's Horizon 2020 programme under grant agreement H2020-693092 MOVING.

REFERENCES

[1] C. Becker and C. Bizer. 2008. DBpedia Mobile: A Location-Enabled Linked Data Browser. In *LDOW*.
[2] R. Dividino, T. Gottron, and A. Scherp. 2015. Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date. In *ISWC*. Springer, 356–373.
[3] R. Dividino, T. Gottron, A. Scherp, and G. Gröner. 2014. From changes to dynamics: dynamics analysis of linked open data sources. In *PROFILES*.
[4] R. Dividino, A. Kramer, and T. Gottron. 2014. An Investigation of HTTP Header Information for Detecting Changes of Linked Open Data Sources. In *ESWC 2014 Satellite Events*. Springer, 199–203.
[5] J. Edwards, K. McCurley, and J. Tomlin. 2001. An adaptive model for optimizing performance of an incremental web crawler. In *WWW*. ACM, 106–113.
[6] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić. 2014. Introducing Wikidata to the Linked Data web. In *ISWC*. Springer, 50–65.
[7] A. Harth and S. Decker. 2004. *Yet another RDF store: Perfect index structures for storing semantic web data with contexts.* Technical Report. DERI Technical Report.
[8] T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan. 2013. Observing Linked Data dynamics. In *ESWC*. Springer, 213–227.
[9] T. Käfer, J. Umbrich, A. Hogan, and A. Polleres. 2012. Towards a dynamic linked data observatory. In *LDOW*. CEUR.
[10] M. Martin, J. Unbehauen, and S. Auer. 2010. Improving the performance of semantic web applications with SPARQL query caching. In *ESWC*. Springer.
[11] C. Nishioka and A. Scherp. 2015. Temporal Patterns and Periodicity of Entity Dynamics in the Linked Open Data Cloud. In *K-CAP*. ACM, 22:1–22:4.
[12] C. Pesquita and F. M. Couto. 2012. Predicting the extension of biomedical ontologies. *PLoS Comput Biol* 8, 9 (2012), e1002630.
[13] K. Radinsky and P. N. Bennett. 2013. Predicting content change on the web. In *WSDM*. ACM, 415–424.
[14] M. Schmachtenberg, C. Bizer, and H. Paulheim. 2014. Adoption of the Linked Data best practices in different topical domains. In *ISWC*. Springer, 245–260.
[15] M. Schuhmacher and S. P. Ponzetto. 2014. Knowledge-based graph document modeling. In *WSDM*. ACM, 543–552.
[16] Q. Tan and P. Mitra. 2010. Clustering-based incremental web crawling. *TOIS* 28, 4 (2010), 17.
[17] G. Tummarello, R. Delbru, and E. Oren. 2007. Sindice.com: Weaving the Open Linked Data. In *ISWC/ASWC*. Springer, 552–565.
[18] J. Umbrich, M. Karnstedt, A. Hogan, and J. X. Parreira. 2012. Hybrid SPARQL queries: fresh vs. fast results. In *ISWC*. Springer, 608–624.
[19] J. Umbrich, M. Karnstedt, and S. Land. 2010. Towards understanding the changing web: Mining the dynamics of linked-data sources and entities. In *KDML*.
[20] D. Vrandečić and M. Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *CACM* 57, 10 (2014), 78–85.
[21] W. E. Zhang, Q. Z. Sheng, K. Taylor, and Y. Qin. 2015. Identifying and Caching Hot Triples for Efficient RDF Query Processing. In *DASFAA*. Springer, 259–274.