Measures for assessing the data freshness in Open Data portals

Sebastian Neumaier* and Jürgen Umbrich†
Vienna University of Economics and Business, Vienna, Austria
Email: *sebastian.neumaier@wu.ac.at, †juergen.umbrich@wu.ac.at

***Abstract*—Many applications and use cases which consume Open Data rely on up-to-date information. However, scarcely any major Open Data portal provides the users with tangible on how current the data sources are. In order to create a freshness metrics that would solve the mentioned issues, we need to i) learn the change history of a data source and ii) apply a heuristic to estimate how up-to-date a data source is. The following paper assesses which information about the change behaviour of a data source in an Open Data portal are available to derive such a change history. In addition, the paper examines and evaluates different freshness estimation heuristics and establishes a freshness metric for Open Data sources.**

I. Introduction

This work addresses the problem of estimating the up-to-dateness of data sources in Open Data portals. It is crucial for the ecosystem and the value chain of the Open Data movement that the advertised data is always reflecting the most up-to-date information, especially for applications and use cases which really on current information. As such, outdated data sources are losing their value, which can negatively impact the data producer, consumers but also the overall Open Data success. For instance, an application that provides a current overview of the environmental data, such as water level, etc., is valuable to the user only if the data is current.

Currently, Open Data portals do not often provide any information on how up-to-date the sources, the users and data providers are dealing with, really are. Some portals provide meta data fields to specify the update frequency of a data source. However, there is no guarantee that the change frequency information is correct and that the data source follows the specified change ratio.

As such, we argue that it is crucial to provide data users and producers with a metric, which indicates how current/up to date the data sources in an Open Data portal are. In order to create a freshness metrics that would solve the mentioned issues, we need to undertake a series of steps. First, one has to learn the change history of a data source which ideally contains the exact time of the content change. Then apply a heuristic to estimate the next likely change time in order to estimate how up-to-date a data source is.

There are two possibilities to collect the change time for a given resource: i) push-based approaches for which the data publishers provides change notifications and ii) pull-based approaches for which one periodically checks if a data source has changed. Only portal providers can have access to the former push-based information, either if the data providers upload a new version of a data source to the portal or if specific metadata fields, in the existing metadata, are edited. For all other use cases, one has to rely on the latter, pull-based approach, to collect change information. In fact, as we will show, many data sources are not directly hosted at the portal itself but by third parties and linked via the metadata. The pull-based monitoring approach has the obvious drawbacks of being more resource consuming than push based change information. Additionally, the monitoring interval influences how accurate one can sample the change times. For instance, the monitoring approach is only able to detect the latest change between two monitoring intervals, even if several changes occurred during two monitoring points.

Once the change history of a data source is collected, we can apply estimation methods to both predict the next change time and check if a data source is still current.

The goal of this work is to first investigate and empirically study the scenarios of push vs pull based change information harvesting in Section II. We do so by discussing three prominent Open data portal software frameworks, namely, CKAN, OpenDataSoft and Socrata and outline the availability of change time information in the metadata of the portals. In Section III, we review suitable and in the literature established change prediction heuristics for the various scenarios. In Section IV we define a resource freshness metric for Open Data portals, based on our heuristics. As a reality check, Section V contains an empirically study to which extent the various scenarios are existing in the Open Data environment by analysing the metadata of 261 Open Data portals with over 4m resources. Next, in Section VI, using a controlled experiment, we evaluate our heuristics, in different scenarios and with varying levels of resource dynamicities. In result, we present first freshness estimates for selected Open Data portals. Lastly, we discuss related work in Section VII and conclude with Section VIII with an outlook of future directions.

II. Resource change information in Open Data Portals

In this section we outline how a portal owner, or external agent, can collect the time of a resource content update to build a change history which is necessary to estimate the freshness of a resource at a given time. First, we will introduce the formal concept of an Open Data portal, discuss different portal frameworks and then highlight three strategies to collect the change history of a particular resource in Open Data portals.

A. Web Data Portal

We consider a Web portal $P = (h_P, D_P, \text{Serv})$ as being a digital catalog, which is accessible via a URL h_P. The portal itself holds a set of dataset URLs $D_P = \{d_1, \ldots, d_n\}$ and provides a set of services $\text{Serv} = \{\texttt{meta}, \texttt{resource}\}$. These services allow a human, or legal agent, to access the provided metadata of the datasets ($\texttt{meta}(d)$ service) and the URL of the actual resources ($\texttt{resource}(d)$ service). The metadata description of a dataset is typically returned as a structured document and holds important contextual information.

In the context of data portals, a *resource* is any target of an URL, which can be hosted internally (i.e., hosted on the same server as the portal) or externally (i.e., a reference to a distant web or file server). The resource URLs of a dataset can be accessed via the $\texttt{resource}(d)$ service. Usually these services are available as HTTP-based RESTful APIs and therefore are accessed via the portal URL h_P. We denote the set of all resource URLs, occurring on a data portal, as R and the set of all dataset URLs as D respectively.

Currently, there are three leading frameworks for Open Data portals, namely CKAN, OpenDataSoft and Socrata. Each of them having their specifics wrt. hosting of resources and availability of change time information in their metadata.

1) CKAN Portal software: This open-source project developed by the Open Knowledge Foundation, is the most prominent Open Data portal software framework. In CKAN the metadata description of a dataset is available as a JSON document and the fields are a mix of predefined fields with fields that are added by the portal provider. By default, CKAN offers the metadata field "`last_modified`" to specify the last update time of a resource. This metadata field relates updates to the actual resources and not modifications of the corresponding metadata.

Regarding the hosting of the resources, the CKAN software allows for the resources to be hosted on external servers and are only described in the portal. Due to this, a data provider, who does not upload his data to the portal, has the responsibility to edit the metadata whenever a resource is changing. In case the resource is directly hosted at the portal, the CKAN software should take care of updating the metadata field, as soon as a new version of the resource is uploaded. According to a CKAN changelog[1] the last-modified field gets updated if the resource is stored and updated locally at the CKAN portal. However, this feature is only available for CKAN versions newer than December 2015. In addition, it has to be taken into consideration that not all portals have been able to upgrade their software yet.

2) Socrata Portal software: The proprietary Socrata data portal product serves as a cloud software, i.e. it hosts the data on their own server, and provides access to datasets via an API. There is a number of states and cities using Socrata, e.g. Washington D.C. or the state of New York. In contrast to CKAN, Socrata data portals contain no references to external resources. A Socrata dataset consists of tabular data and corresponding metadata, which is stored in the system's internal database, and can be exported in several data formats. When it comes to updaing the data, we observe one relevant metadata field on Socrata portals: "`rowsUpdatedAt`".

[1]http://docs.ckan.org/en/latest/changelog.html#v2-5-1-2015-12-17, last accessed 2015-02-12

TABLE I
CHANGE TIME CHARACTERISTICS OF OPEN DATA PORTALS.

	CKAN	Socrata	OpenDataSoft
resources	local /external	local	local
metadata field	`last_modified`	`rowsUpdatedAt`	`modified`

3) OpenDataSoft Portal software: A French company that provides commercial data portal software, similar to the Socrata portal. Similarly to Socrata, the OpenDataSoft portals do not distinguish between a resource and a dataset. The portal's framework requires that all resources are directly uploaded and stored in the portal. Consequently, the metadata descriptions of OpenDataSoft datasets are partially aligned with the predicates in the DCAT schema [8]. On addition, the temporal update information of a dataset is stored in the metadata field "`modified`". Table I lists the key findings.

B. Strategies for collecting the change history

Freshness estimators generally use the change history of a resource to estimate how likely it is that the given resource is up-to-date. The change history of a resource contains the points in time at which changes to the content are known. There are three possible scenarios when collecting the the change history from the Open Data portals:

1) **Push-based change history:** The first scenario assumes that the data provider pushes change notification to the portal, by either uploading a new version to the portal or by editing specific metadata fields in the description of the dataset for a resource. In that scenario, the change history will contain the complete set of all change times.

PULL-BASED CHANGE HISTORY BASED ON SAMPLING:

The second and third scenario assumes that the push-based change history of a resource is not accessible. It is either because we are not the portal provider or because the data is stored externally or the data provider does not provide metadata updates. In such case, we need to actively monitor the metadata of a resource for change information, upon available, or monitor the resource for content changes directly. In [7], the authors identified two categories of information sampling:

2) **Age sampling:** The second scenario assumes that an agent has access to the latest change time of a resource, also referred to as the age of a resource. Such age information can be either collected from specific fields in the metadata or HTTP header. In terms of HTTP response information, we refer to the `last_modified` header field which should be returned upon a HTTP GET or HEAD request. Yet, not all portals or HTTP servers provide such last-modification date information.
3) **Comparison sampling:** The third scenario relies upon monitoring the actual content of a resource and detect changes by comparing two versions. In contrast to the age-sampling approach, the third scenario only enables

us to detect if there are any changes in comparison to the last sampling point.

III. Change estimation heuristics

As discussed in the previous section, there are three different approaches to collect change information: (i) comparison-based, (ii) age-based, and (iii) push-based. This section introduces existing estimation heuristics and discusses their applicability for the three identified sources of information.

A. Comparison sampling

We assume that the sampling interval S is constant with n sampling points s_i, $0 \leq i \leq n$. Then $T = nS = \frac{n}{f}$ is the total monitoring period and $f = 1/S$ the frequency at which the resource is accessed. Let X_i be a boolean variable which results in 1 if the resource changed between the sampling points s_i and s_{i-1} and 0 otherwise. Furthemore, let $X = \sum_{i=1}^{n} X_i$ be the total number of detected updates.

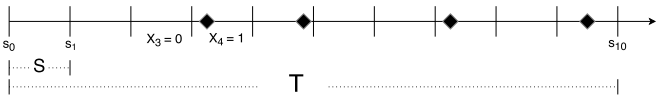

Fig. 1. Example change history of a resource.

1) Empirical distribution ($\mathrm{C}_{\mathrm{EmpDist}}$): As a first method, we build an empirical distribution function based on the intervals between observed changes.

Example III.1. For instance, based on the change history, Figure 1, we get the following binary information:

$$(0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1) \tag{1}$$

where 0 indicates no change, and 1 a change based on comparing the content at the current point to the content at the previous sampling point. This history results in the intervals $(4\ 1\ 3\ 2)$, which we use as input for an empirical distribution.

2) Poisson process ($\mathrm{C}_{\mathrm{ChoNaive}}$ & $\mathrm{C}_{\mathrm{ChoImpr}}$): There are several scholarly approaches in estimating the update frequency of data, in the context of web sites and web crawling. In [5] and [9] the authors assume that a data item changes by a Poisson process since "a Poisson process is often used to model a sequence of random events that happen independently with a fixed rate over time". In [5] Cho and Garcia-Molina suggest X/T as an *intuitive* estimation of how often a resource has changed in a given time period, when only binary information between accesses (i.e., change or no change) is available. The authors assume that the updates of online data sources follow a Poisson process with rate λ. They define the ratio of the chance frequency to the access frequency $r = \lambda/f$ and derive the following estimated frequency ratio:

$$\hat{r} = \frac{\hat{\lambda}}{f} = \frac{1}{f}\left(\frac{X}{T}\right) = \frac{X}{n} \tag{2}$$

Then $\hat{r}$ is used to estimate the Poisson parameter $\hat{\lambda}(= rf)$.

As reported in [5], this estimation is highly biased if the update history is incomplete, i.e. if there are changes to the resource, which are not detected by the sampling. Therefore, the authors provide an improved estimator ($\mathrm{C}_{\mathrm{ChoImpr}}$) with the frequency ratio

$$\hat{r} = \log\left(\frac{n - X + 0.5}{n + 0.5}\right) \tag{3}$$

3) Markov chain approach ($\mathrm{C}_{\mathrm{UmbMarkov}}$): A Markov chain is a probabilistic process where the probability distribution of the next state depends on the current state only. This is stated in the so-called Markov property: given a present state, the future and past states are independent. In [13] the authors use Markov chains to schedule the next crawl time for URLs based on previously observed changes to the content (i.e., *comparison sampling*).

Considering the history of detected changes $X_1, \ldots, X_n$ for sampling points s_1 to s_n, we construct the probabilities for a Markov chain based on the binary state transitions in the series.

Example III.2. Using the change history of example III.1, we can build a state-change matrix by counting the state transitions, cf. Figure 2(a).

i \i+1	1	0	TOTAL
0	3	3	6
1	1	2	3

(a) Simple state-change matrix.

i \i+1	1	0	TOTAL
00	2	1	3
01	1	1	2
10	1	1	2
11	0	1	1

(b) History of depth 2.

Fig. 2. Transition-counts for history k=1 (a) and k=2 (b).

The estimated probabilities for a change in the next state, under the assumption that there was no previous change (i.e. current state 0), is computed based on the counts: $P(1|0) = 3/6$. This approach can be further generalized by considering the last k states for computing the probabilities. For example, $k = 2$ results in the transition matrix in Figure 2(b). Here, the probability of a next state depends on the last two observed sampling points, e.g. $P(1|00) = 2/3$.

However, this generalization approach leads to exponential (2^k) increase of possible state changes and therefore is only practical for small k.

B. Age sampling

1) Empirical distribution ($\mathrm{A}_{\mathrm{EmpDist}}$): In line with the empirical distribution of comparison-based samples (cf. Section III-A1), we build an empirical distribution function by using the intervals between the observed last-modified times.

2) Poisson process ($\mathrm{A}_{\mathrm{ChoNaive}}$ & $\mathrm{A}_{\mathrm{ChoImpr}}$): Regarding age sampling (i.e. availability of last-modified time) the authors use in [5] X/T ($\mathrm{A}_{\mathrm{ChoNaive}}$) as an intuitive estimator to predict the change frequency λ of a Poisson process. In the age-sampling case, X denotes the number of detected changes, while T is the sum of the timespan between change and access time (and not the total monitoring period).

Additionally, in order to improve the bias for small N, the authors propose the λ-estimation X'/T ($\mathrm{A}_{\mathrm{ChoImpr}}$):

$$X' = (X - 1) - X/(N * log(1 - X/N)) \tag{4}$$

For an in depth explanation of these estimators and their implementation we refer the reader to [5].

C. Push-based information

In case the complete set of all change times is available, we propose two change estimation heuristics: the use of an empirical distribution ($\mathrm{P}_{\mathrm{EmpDist}}$) or a Poisson-based distribution ($\mathrm{P}_{\mathrm{ChoNaive}}$) with an intuitive estimator $\lambda = X/T$.

We build the empirical distribution function for push-based information by considering the intervals between all update times. Likewise, we use all update times to compute an intuitive estimator for a Poisson process model, i.e., X is the number of all push-updates and T the total time elapsed.

D. Implementation details

We implement the introduced methods for two real-world case studies of Open Data providers:

(i) **Estimate change time for a given confidence**: A portal provider wants to know if the resource is still up-to-date with a given confidence value. This would allow for triggering notifications or warnings for likely outdated resources. To do so, the estimator should return the predicted time interval for the next change (given a confidence value).

(ii) **Estimate probability for a given time**: A portal provider wants to know the probability that the resource is up-to-date for a given timespan, after the last known update. This can be considered as the inverse problem of the previous use case: such an estimator should return the probability that a change occurs within a given timespan. This estimation serves as our freshness metric for a portal.

We discuss the implementation and calculation of the mentioned case studies, using our heuristics in the following.

1) Probability distribution approach: The quantile function (or inverse cumulative distribution function) of a probability distribution returns the value at which a given probability of a random variable is less than or equal to this value. In relation to case study (i), we calculate the estimated time interval for the next change of a resource by considering the corresponding p-quantile (e.g., 80%-quantile) of the distribution of the observed change-times.

Computing the probability that a change occurs within a fixed, given, timespan (case study (ii)) corresponds to the cumulative distribution function: the probability that a random variable is less or equal than a given value (i.e., the probability that the next change time is less or equal to the given time).

2) Markov chain approach: In order to compare the introduced Markov estimation approach to the other approaches in a meaningful way, we have to consider the minimum number of estimated future intervals, such that the probability of observing an update to the resource is higher than p.

This number of sampling points can be easily computed by considering the complementary problem: The minimum number of intervals such that the probability of no updates is smaller than $q(= 1 - p)$. The complementary problem, i.e., the probability of a number of recurrent 0s, is calculated by multiplying the probabilities based on a given history.

Example III.3. Considering example III.1: The probability for three consecutive intervals without a change (i.e., three 0s), on the condition that we previously observed two states 01, is:

$$P(0|01) \cdot P(0|10) \cdot P(0|00) = 1/2 \cdot 1/2 \cdot 1/3 \qquad (5)$$

We can compute, the number of intervals based on a given probability p (cf. case study (i)) and a history H of fixed length k, in the following way:

```
Estimate(p, H):
    q = 1 - p
    I = 0   // estimated number of intervals
    Pz = 1  // probability for recurrent zeros
    While (Pz > q) do
        I = I + 1
        Pz = Pz * P(0|H);
        H = H[1:] + "0";//shift left and add 0
    return I
```

We shift the last history H to the left and add a new 0 in every iteration. Note, in the special case when H contains only 0 (meaning the resource is static) the algorithm would not terminate ($P(0|H) = 1$ since we have only one state).

Analogously to the above pseudo code, we can compute the probability that a change occurs in a given number of intervals I (cf. use case (ii)) with the following function:

```
Estimate(I):
    i = 0
    Pz = 1  // probability for recurrent zeros
    While (i < I) do
        i = i + 1
        Pz = Pz * P(0|H);
        H = H[1:] + "0";
    return 1 - Pz
```

IV. Freshness Metric

We propose a measure for the freshness of resources on Open Data portals, under the assumption that the update frequency information of a resource is not available. Therefore, the proposed metric relies on previously observed or monitored change history and utilizes an estimation algorithm for calculating a freshness value.

Table II lists the introduced estimation methods for push-, comparison-, and age-based update frequency estimation. The methods' applicability depends on the available history source, i.e. push- or pull-based.

TABLE II
Introduced estimation approaches.

Acronym	Sampling type	Authors/*Comment*
$\mathrm{C}_{\mathrm{EmpDist}}$	Comparison	*Empirical Distribution*
$\mathrm{C}_{\mathrm{ChoNaive}}$	Comparison	Cho & Garcia-Molina
$\mathrm{C}_{\mathrm{ChoImpr}}$	Comparison	Cho & Garcia-Molina
$\mathrm{C}_{\mathrm{UmbMarkov}}$	Comparison	Umbrich et al.
$\mathrm{A}_{\mathrm{EmpDist}}$	Age	*Empirical Distribution*
$\mathrm{A}_{\mathrm{ChoNaive}}$	Age	Cho & Garcia-Molina
$\mathrm{A}_{\mathrm{ChoImpr}}$	Age	Cho & Garcia-Molina
$\mathrm{P}_{\mathrm{EmpDist}}$	Push-based	*Empirical Distribution*
$\mathrm{P}_{\mathrm{ChoNaive}}$	Push-based	Cho & Garcia-Molina

Based on our initial formalization of data portals (cf. Section II-A) we denote the set of all resources of a portal P as $\mathtt{R}(P) = \bigcup_{d \in D_P} \mathtt{resource}(d)$. Note, that we union all resources of the datasets of a portal ($\bigcup_{d \in D_P}$) since a dataset

can potentially hold multiple resources and a resource can appear in multiple dataset.

We then define the freshness of a portal P at time t as the average of the probabilities that the resource is up-to-date (denoted by $1 - \texttt{Estimate}(r,t)$):

$$\texttt{Fresh}(P,t) = \frac{1}{|\,\texttt{R}(P)|} \sum_{r \in \texttt{R}(P)} (1 - \texttt{Estimate}(r,t)) \quad (6)$$

For the calculation of these probabilities for the different approaches see the discussion of case study (ii) in Section III-D.

V. Empirical study of 261 Open Data Portals

We empirically study the metadata of 4.4m resources in 261 Open Data portals and investigate which of the three scenarios can be applied to estimate the freshness for a given resource. Our empirical evaluation is based on the data gathered by the Open Data Portal Watch (ODPW) monitoring framework [14].[2] At the time of writing, the framework monitors 261 Open Data portals (148 CKAN, 102 Socrata, and 11 OpenDataSoft portals) with 1.1M datasets describing 4m resources, of which are 2.2m distinct ones. Additionally, we study how many resources are available for push-based and pull-based (age vs comparison sampling) freshness estimation for each portal framework separately.

A. Resource hosting: ratio of local vs external sources

The first study is to compute the ratio of local vs external sources to estimate to how many portals we could apply only the push-based freshness estimators. Since Socrata and OpenDataSoft portals host all of their resources locally (cf. Section II-A), in this evaluation, we will focus only on the data of the 130 CKAN portals. To compute if a resource is local or not, we i) match the domain name of the resource URL to the domain name of portal URL and ii) inspect if the metadata key-value pair 'url_type': '$upload$' is present in the metadata of the resource, indicating that the resource was uploaded to the portal.

The results of analysing 130 CKAN portals, describing 3.8m resources (with 2m distinct resource identifiers), are listed in Table III; portals are grouped by their local vs external ratio. Only a small amount of 9 CKAN portals host all of the described resources locally, while the majority of the resources belong to 54 portals with a local to external ratio between 0 and 25%. Another 40 portals have a local to external resource ratio between 25% and 99.99%. This indicates, that push-based freshness estimators are not sufficient for CKAN portals and pull-based approaches need to be applied in addition to cover all resources.

TABLE III
Ratio of local vs. external resources on CKAN.

ratio	0	< 0.25	< 0.5	< 0.75	< 1	1
$\|p\|$	27	54	9	7	27	9
% of $\|r\|$	5.76%	88.48%	0.38%	0.05%	1.12%	4.21%

[2] http://data.wu.ac.at/portalwatch/

TABLE IV
Change information in metadata and resources.

	CKAN	**Socrata**	**OpenDataSoft**
resources	4049851	181548	8757
distinct	2116940	165966	8757
Push-based approach			
local	227204	181548	8757
Age-based sampling			
metadata			
exists	3884657	175332	8742
with value	146230	175332	8742
external with value	130587	0	0
HTTP			
URLs	4049851	181548	8757
HTTP lookups	3097665	67198	7958
with value	1936612	67198	0
no age information	122612	5331	15

B. Availability of change time information

Next, we inspect the metadata of the datasets and the resource HTTP header responses for change time information about the resources. In the CKAN portals, we scan the metadata for the "`last_modified`" field, in Socrata portals for the "`rowsUpdatedAt`" field and in OpenDataSoft metadata for the "`modified`" field. We performed a lookup on valid URLs to collect their HTTP response headers to find out if the resource was available (response code between 200 and 400). Please note, that we perform HTTP HEAD lookups, if applicable, to retrieve the necessary information. Since Socrata portals do not implement the HTTP HEAD protocol, we performed HTTP GET operations and cancel the connection after receiving the first content bytes.

The results of this study are summarized in Table IV. An interesting observation is the ratio of total vs distinct resource identifiers; we see that the resources in CKAN portals occur in more than one portal. We also see the very low number of local resources in CKAN portals (cf. Table III).

Regarding the change time information in the metadata, we observe that nearly all datasets in Socrata and OpenDataSoft portals provide these information with values. This stands in contrast to observations about CKAN portals. Here a very small percentage of datasets provide actual time information about changes in the metadata, of which most datasets describe external resources. This results indicate that the majority of the CKAN portal data publishers do not yet update the metadata information of their datasets if the resource content changed.

Inspecting the HTTP Headers of a total of 3.1m resources reveals that 66% of the CKAN, 100% of the Socrata and 0% of the OpenDataSoft resources have the `last_modified` header field.

C. Discussion of results

Overall, our empirical evaluation of the 261 portals reveals that applying only push-based freshness estimators is suitable for Socrata and OpenDataSoft portals; all their resources are

locally stored. For CKAN portals, if full resource coverage is desirable, there is a need for applying additional pull-based estimators. In regards to the pull-based estimators, we suggest direct monitoring of the metadata of Socrata and OpenDataSoft portals, for which the majority of the metadata contains change time information. Similarly, CKAN portals require direct inspection of the HTTP header information and downloading the content of the files, since not all metadata provide change time information.

VI. Evaluation of heuristics

In this section we evaluate our implemented heuristics, with various parameters in a controlled environment, to cater for the different scenarios. In addition, we will apply our estimators on changing resources from 77 Socrata portals to estimate their resource freshness.

A. Controlled environment evaluation

In order to test and evaluate the above introduced estimation approaches in different scenarios and with varying levels of document dynamicities we performed a controlled experiment: we simulate the changes of resources on Open Data portals by using the real-world revision histories of Wikipedia articles. We gathered a collection of data by randomly selecting 1562 articles from the Wikipedia API.[3] Among the 1562 articles, we filtered out articles with a revision history shorter than 3 years and less than 30 revisions in total. Eventually, we grouped the articles based on their average change frequency (λ) and their range of maximum and minimum update intervals (δ), cf. Table V (numbers given in days). Further, we divided the documents into two subsets, the first one containing 70 documents, showing a more regular change behavior with change intervals closer to their average change intervals, and the second set covering 341 documents that show more irregular change updates.

TABLE V
Groups of Wikipedia articles for simulation.

	Articles	Filter	Average λ
All	1562	–	30d
Regular	70	$20d < \lambda < 60d \wedge \delta < 250d$	28d
Irregular	341	$\delta > 500d$	62d

We evaluate the comparison, age, and push-based estimation methods for the above case study (i): *given a confidence value p, we report the ratio of successfully predicted updates to Wiki articles.* For the push-based evaluation we build the change history based on all change points. For the pull-based estimator, we simulate the sampling process by setting a fixed interval of S days and used the first 10 updates to build the change history. This means also, that we only consider the last update in a sample interval, in contrast to the push-based, for which considers all updates. Afterwards, we compute the probability p that the next real-world update interval is part of the update interval. If the p value is higher than a given threshold, we consider that the estimator correctly predicted the document change. We repeat the same procedure for the next 20 updates for each document, and compute the average of all documents and update points. Given the space limitations of the paper, we report the results for $p = 0.7$ and $S = 10, 50$ and for $p = 0.9$ and $S = 10$, which we manually selected as being representative across our results.

1) Comparison sampling: Table VI reports the value of the average correct change prediction after 20 updates. Additionally, in order to get a better understanding by how close the estimators predict the actual change time, we compute the average distance of the estimation change time to the real update time. For instance, based on the $\mathrm{C}_{\mathrm{UmbMarkov}}$ approach, for a given p of 0.7, and a sampling rate S of 10 days, the average correctly estimated outdatedness is 73% for the *Irregular* articles. However, the average distance to the real update times is 96 days, meaning that on average the estimator predicts a change interval, which is 96 days longer than the actual change interval. We can see that the $\mathrm{C}_{\mathrm{UmbMarkov}}$ performs best in most of the test settings, followed by using the empirical distribution. However, the average distance to the actual update points is higher for these two estimation methods. It is because the influence of outliers on these methods is stronger than on the Poisson-based estimations. It is the result of the fact that the variance of a Poisson distribution is equal to the parameter λ. Overall, we see that the $\mathrm{C}_{\mathrm{UmbMarkov}}$ estimator has a prediction ratio close to the specified p value, while the other estimators show a slightly lower prediction ratio. This indicates, that the Markov-based approach provides a good and accurate estimation, which can be find an effective application in a real-world scenario.

TABLE VI
Comparison sampling results.

Estimator	*All*		*Regular*		*Irregular*	
`p = 0.7`	`S = 10d`					
$\mathrm{C}_{\mathrm{EmpDist}}$	0.59	40d	0.66	40d	0.60	90d
$\mathrm{C}_{\mathrm{ChoNaive}}$	**0.67**	36d	0.67	35d	0.63	83d
$\mathrm{C}_{\mathrm{ChoImpr}}$	0.66	**35d**	0.62	**34d**	0.61	**82d**
$\mathrm{C}_{\mathrm{UmbMarkov}}$	0.51	42d	**0.76**	41d	**0.73**	96d
`p = 0.7`	`S = 50d`					
$\mathrm{C}_{\mathrm{EmpDist}}$	0.54	40d	0.57	40d	0.57	84d
$\mathrm{C}_{\mathrm{ChoNaive}}$	**0.65**	**37d**	0.36	40d	0.63	78d
$\mathrm{C}_{\mathrm{ChoImpr}}$	0.27	43d	0.31	**36d**	0.47	**76d**
$\mathrm{C}_{\mathrm{UmbMarkov}}$	0.58	39d	**0.59**	40d	**0.68**	82d
`p = 0.9`	`S = 10d`					
$\mathrm{C}_{\mathrm{EmpDist}}$	0.81	66d	0.87	70d	0.80	145d
$\mathrm{C}_{\mathrm{ChoNaive}}$	0.71	38d	0.70	37d	0.67	85d
$\mathrm{C}_{\mathrm{ChoImpr}}$	0.57	**36d**	0.66	**35d**	0.60	**83d**
$\mathrm{C}_{\mathrm{UmbMarkov}}$	**0.88**	84d	**0.94**	85d	**0.90**	184d

Looking more closely into the $\mathrm{C}_{\mathrm{UmbMarkov}}$ approach, Table VII illustrates the average estimation rate for varying state change history. Based on the table, in general, we observe better results for a state history of 1 for the $\mathrm{C}_{\mathrm{UmbMarkov}}$ algorithm (i.e., the probabilities for the next state are only based on one previous state) than for considering two states. Each additional

[3] http://en.wikipedia.org/w/api.php

state requires exponentially more observed changes to compute the probability for all possible state change combinations. Unfortunately, there are not enough data points available in the evaluation data to collect sufficient information about all state change combinations. The missing information directly reflects the predication accuracy as observed in our evaluation with a history of 2 states.

TABLE VII
$C_{UmbMarkov}$ RESULTS, SAMPLING INTERVAL: 20D.

p	State history	*All*	*Regular*	*Irregular*
0.7	1	0.61	0.74	0.70
	2	0.60	0.73	0.68
0.9	1	0.81	0.94	0.89
	2	0.78	0.93	0.89

2) Age sampling: Similar to Table VI, Table VIII lists the average success rate of predicting updates to articles for the age sampling approaches. Please note, that only the distribution-based estimators are evaluated since they consider change intervals, while the Markov model is just using change states. Overall, the results show that the empirical estimator performs best for different p values and the three document sets. Interestingly, the $A_{EmpDist}$ estimators performs better in the age-based sampling than in the comparison sampling for increasing S, while also showing a higher average distance to the real interval (cf. $S = 10$ vs $S = 50$ for $p = 0.7$ in Table VIII and Table VI).

TABLE VIII
AGE SAMPLING RESULTS.

Estimator	*All*		*Regular*		*Irregular*	
p = 0.7	S = 10d					
$A_{EmpDist}$	**0.66**	43d	**0.69**	42d	**0.64**	101d
$A_{ChoNaive}$	0.46	**36d**	0.59	**33d**	0.61	**84d**
$A_{ChoImpr}$	0.46	36d	0.59	33d	0.61	84d
p = 0.7	S = 50d					
$A_{EmpDist}$	**0.69**	60d	**0.71**	44d	**0.66**	105d
$A_{ChoNaive}$	0.33	**51d**	0.26	38d	0.48	**84d**
$A_{ChoImpr}$	0.35	51d	0.28	**37d**	0.49	84d
p = 0.9	S = 10d					
$A_{EmpDist}$	**0.84**	73d	**0.87**	72d	**0.82**	170d
$A_{ChoNaive}$	0.46	**36d**	0.59	**33d**	0.61	**84d**
$A_{ChoImpr}$	0.46	36d	0.59	33d	0.61	84d

3) Push-based information: Eventually, in Table IX we report our results for the introduced push-based methods. Thanks to a more accurate change history, the results show that the $P_{ChoNaive}$ estimators provide much better prediction results than for age- and comparison-based sampling. Interestingly, the $P_{EmpDist}$ estimator performs better than $P_{ChoNaive}$ with an increasing p.

4) Conclusion: As a conclusion of these experiments, we suggest to use the $C_{UmbMarkov}$ estimator for comparison-based sampling and high p values. For age sampling and push-based scenarios, the best/most effective estimator is the $A_{EmpDist}$ and $P_{EmpDist}$ respectively. However, more experiments and studies are required to fully understand and evaluate the strength and weaknesses of each estimator, in various scenarios.

TABLE IX
PUSH-BASED INFORMATION RESULTS.

p	Estimator	*All*		*Regular*		*Irregular*	
0.7	$P_{EmpDist}$	0.67	27d	0.69	29d	0.62	56d
	$P_{ChoNaive}$	0.71	28d	0.69	28d	0.70	60d
0.9	$P_{EmpDist}$	0.85	53d	0.89	62d	0.82	106d
	$P_{ChoNaive}$	0.71	28d	0.69	28d	0.70	60d

B. Freshness estimation of Socrata Open Data portals

Next, we inspect the average resource freshness of Open Data portals by computing the introduced metric (cf. Section IV). To do so, we use weekly harvested metadata available from the Open Data Portal Watch framework [14]. Since the framework does not provide weekly header information, we rely on the change time information in the metadata and apply age-based sampling. Based on our findings about the absence and reliability of change information in CKAN (cf. Table IX), we only consider 100 Socrata portals for this study.

We observed that 77 portals contain at least one resource which changed (meaning, we observe at least two different change times). Overall, among the 77 portals, only 30% of the resources changed. Next, we estimated the freshness for the dynamic resources of the 77 portals. As an input for the heuristics we use all available weekly snapshot between week 25 in 2015 and week 13 of 2016 and estimate the freshness for week 14, i.e., 7 days after the last snapshot.

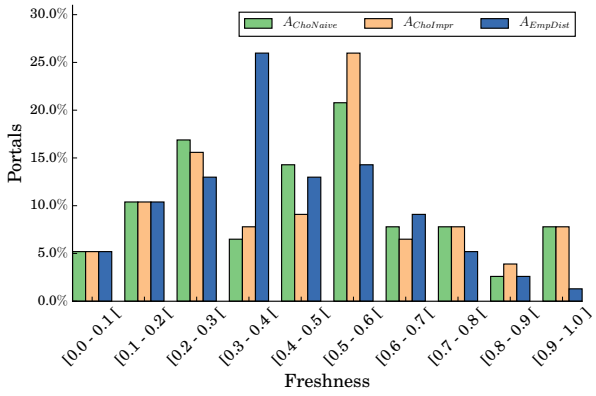

Fig. 3. Freshness values of Socrata portals.

The histogram in Figure 3 provides the distribution of the average freshness values for the dynamic resources of the 77 portals, based on our age sampling heuristics. According to our observations, the freshness ranges between 0.3 and 0.6 among the estimators, with $A_{EmpDist}$ estimating a higher ratio of the portals to be outdated than $A_{ChoNaive}$ and $A_{ChoImpr}$. Based on the previous results, we can expect that the estimation of $A_{EmpDist}$ are more accurate than the others. As such, we can

conclude that the majority of the 77 Socrata Open Data portals have an average resource freshness value of less than 0.5, meaning that the likelihood that a resource did not change is less than 50%.

VII. Related Work

The following section lists related approaches for measuring data freshness and estimating update frequencies.

a) Freshness based on known update frequency: Peralta [10] provides a taxonomy of existing works, dealing with data freshness and data accuracy in the context of Data Integration Systems. It discusses different freshness definitions and measurements as well as and identifies the need for "metadata about users' expectations and source properties" [10], in a freshness evaluation process.

Ballou et al. [2] propose a general formula to measure *timeliness* in information manufacturing systems.The paper defines *currency* as the age of the primitive data units and *volatility* as the shelf life of the item (i.e., how long the item remains valid). Based on these variables, the authors define *Timeliness* of primitive data units as: $Timeliness = \{\max[1 - currency/volatility), 0]\}^s$; s allows to control the "sensitivity of timeliness", i.e., it controls how the currency-volatility ratio affects the timeliness value. The timeliness definitions proposed in [2] have been used in various publications, dealing with time-related quality assessment [6], [11], [12].

In [3] the authors elaborate on the concept of *data freshness* with the intuitive meaning "Is it fresh enough with respect to the user expectations". They identify a "currency factor" as the gap between data extraction and data delivery and a "timeliness factor" as the creation frequency or update frequency of data. Based on these factors, the paper defines a currency, obsolescence, freshness-rate and timeliness metric.

A very basic definition of *freshness* and *age* in the context of database updates can be found in [4]. The authors define two freshness metrics and change models of the underlying data using a Poisson process model (further developed in [5]).

In 2014, Atz [1] proposed τ, a metric to assess *timeliness* of datasets in a data catalogue. To this end, the metric fully relies on the availability of two metadata properties in a dataset: *update frequency* and *last substantial update*. The timeliness of a catalogue is then calculated by averaging over the datasets. However, the paper identifies the substantial problem of non-standardized and missing update-frequency values in the inspected data portals which are required to assess this metric.

b) Estimation of Update Frequency: In [5] and [9] (introduced in Section III) the authors assume a Poisson process for changes of data on the Web. In [7] the authors highlight issues of the Poisson-based approaches and propose improved algorithms to overcome these biases.

VIII. Conclusion

In this work we have addressed the problem of establishing a freshness metric for Open Data portals. We have discussed the specifics of three prominent portal frameworks and three approaches (1 push-based and 2 pull-based) to build the change history for a given resource. Further, we have introduced four freshness estimators and a new freshness metric.

We have empirically studied the metadata of 4.4m resources in 261 portals and conclude that for Socrata and OpenDataSoft portals, one can directly apply push-based or pull-based approaches. For CKAN portals, one needs to use all three approaches to compute the freshness for all resources in a portal. Our evaluation of the four freshness estimators, based on the revision history of Wikipedia articles, has revealed that estimators based on markov-chains show the best "precision" in estimating the up-to-dateness of a resource.

Our future research will focus on improving and investigating additional freshness estimators. A promising approach is to study Markov chains, in greater detail. Additionally, we will study the change behaviour of Open Data resources to gain a better insight about change distributions (e.g. Poisson process). Eventually, we plan to periodically compute the freshness metric for all resources in the Open Data Portal Watch project.

Acknowledgement

This work has been supported by the Austrian Research Promotion Agency (FFG) under the project ADEQUATe (grant no. 849982).

References

[1] Ulrich Atz. The Tau of Data: A New Metric to Assess the Timeliness of Data in Catalogues. In *International Conference for E-Democracy and Open Government (CeDEM2014)*, 2014.

[2] Donald Ballou, Richard Wang, Harold Pazer, and Giri Kumar Tayi. Modeling information manufacturing systems to determine information product quality. *Management Science*, 44(4):462–484, April 1998.

[3] Mokrane Bouzeghoub. A framework for analysis of data freshness. *Proceedings of the 2004 international workshop on Information quality in informational systems - IQIS '04*, page 59, 2004.

[4] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. *ACM SIGMOD Record*, 29(2):117–128, 2000.

[5] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3):256–290, 2003.

[6] Olaf Hartig and Jun Zhao. Using Web Data Provenance for Quality Assessment. *Proceedings of the 1st Int Workshop on the Role of Semantic Web in Provenance Management SWPM at ISWC*, 526:6, 2009.

[7] Xiaoyong Li, Daren B H Cline, and Dmitri Loguinov. Temporal Update Dynamics under Blind Sampling. *Ieee Infocom 2015*, pages 1634–1642, 2015.

[8] Fadi Maali and John Erickson. Data Catalog Vocabulary (DCAT). http://www.w3.org/TR/vocab-dcat/, January 2014.

[9] Norman Matloff. Estimation of internet file-access/modification rates from indirect data. *ACM Transactions on Modeling and Computer Simulation*, 15(3):233–253, 2005.

[10] Verónika del Carmen Peralta Costabel. Data freshness and data accuracy: a state of the art. *Reportes Técnicos 06-13*, 2006.

[11] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211, 2002.

[12] Sandra de F Mendes Sampaio, Chao Dong, and Pedro R Falcone Sampaio. Incorporating the Timeliness Quality Dimension in Internet Query Systems. *Web Information Systems Engineering – WISE 2005 Workshops, New York, New York*, 3807:53–62, 2005.

[13] Jürgen Umbrich, Nina Mrzelj, and Axel Polleres. Towards capturing and preserving changes on the Web of Data. In *First DIACHRON Workshop on Managing the Evolution and Preservation of the Data Web co-located with 12th European Semantic Web Conference*, pages 50–65, 2015.

[14] Jürgen Umbrich, Sebastian Neumaier, and Axel Polleres. Quality assessment & evolution of open data portals. In *The International Conference on Open and Big Data*, pages 404–411, Rome, Italy, August 2015. IEEE.