Dataset Retrieval

Sven R. Kunze
Department of Computer Science
Chemnitz University of Technology
Straße der Nationen 62, 09111 Chemnitz
Email: sven.kunze@informatik.tu-chemnitz.de

Sören Auer
AKSW, Institute of Computer Science
University of Leipzig
Augustusplatz 10, 04109 Leipzig
Email: auer@informatik.uni-leipzig.de

***Abstract*—Recently, a large number of dataset repositories, catalogs and portals are emerging in the science and government realms. Once a large number of datasets are published on such data portals, the question arises how to retrieve datasets satisfying a user's information need. In this article, we present an approach for retrieving datasets according to user queries. We define *dataset retrieval* as a specialization of information retrieval. Instead of retrieving documents that are relevant to a certain information need, dataset retrieval describes the process of returning relevant RDF datasets. As with information retrieval, the term *relevance* cannot be clearly defined when using traditional methods like stemming. The inherent usage of RDF in RDF datasets enables a better way of retrieving relevant ones. We therefore propose an additional retrieval mechanism, which is inspired by facet search: *dataset filtering*. When querying, the entire set of available datasets is processed by a set of *semantic filters* each of which can unambiguously decide whether or not a given dataset is relevant to the query. The resulting set is then given back to the requester. We implemented and evaluated our approach in CKAN, which fuels publicdata.eu and is the most popular data portal worldwide.**

I. Introduction

Integrating and analyzing large amounts of data plays an increasingly important role in todays society. Often, however, new discoveries and insights can only be attained by integrating information from dispersed sources. Despite recent advances in structured data publishing on the Web (such as RDFa and the schema.org initiative) the question arises how to publish and to describe larger datasets in order to make them easily discoverable and facilitate their integration as well as their analysis.

One approach for addressing this problem are data portals, which enable organizations to upload and describe datasets using comprehensive meta-data schemes. Similar to digital libraries, networks of such data catalogs can support the description, archiving and discovery of datasets on the Web. Recently, we have seen a rapid growth of data catalogs being made available on the Web. The data catalog registry datacatalogs.org, for example, lists already 285 data catalogs worldwide. Examples, for the increasing popularity of data catalogs are Open Government Data portals, data portals of international organizations and NGOs as well as scientific data portals. Governments and public administrations started to publish large amounts of structured data on the Web, mostly in the form of tabular data such as CSV files or Excel sheets. Examples are the US' data portal data.gov, the UK's data portal data.gov.uk, the European Commission's open-data.europa.eu portal as well as numerous other local, regional and national data portal initiatives. Also in the research domain, data portals can play an important role. Almost every researcher works with data. However, quite often only the results of analyzing the data are published and archived. The original data, that is ground truth, is often not publicly available thus hindering repeatability, reuse as well as repurposing and consequently preventing science to be as efficient, transparent and effective as it could be. An example of a popular scientific open data portal is the Global Biodiversity Information Facility data portal (gbif.org). Further, many international and non-governmental organizations operate data portals such as the World Bank Data portal or the data portal of the World Health Organization. Despite being a relatively new type of information system first commercial (e.g. Socrata) and open-source data portal implementations are already available.

In this article, we present an approach for retrieving datasets according to user queries. We define *dataset retrieval* as a specialization of information retrieval. Instead of retrieving documents that are relevant to a certain information need, dataset retrieval describes the process of returning relevant RDF datasets. As with information retrieval, the term *relevance* cannot be clearly defined when using traditional methods like stemming. The inherent usage of RDF in RDF datasets enables a better way of retrieving relevant ones. We therefore propose an additional retrieval mechanism, which is inspired by facet search [1]: *dataset filtering*. When querying, the entire set of available datasets is processed by a set of *semantic filters* each of which can unambiguously decide whether or not a given dataset is relevant to the query. The resulting set is then given back to the requester. We implemented and evaluated our approach in CKAN, which fuels publicdata.eu and is the most comprehensive data portal worldwide.

This article is structured as follows: We describe our concept of dataset retrieval in section II. We discuss a usage scenario in section III. Our extension of the VoID dataset metadata vocabulary to accommodate retrieval relevant information is presented in section IV. We describe our dataset similarity ranking in section V and their usage for recommendation in section VI. Our implementation of the approach for CKAN is outlined in section VII and evaluated in section VIII. We present related work in section IX and conclude with an outlook on future work in section X.

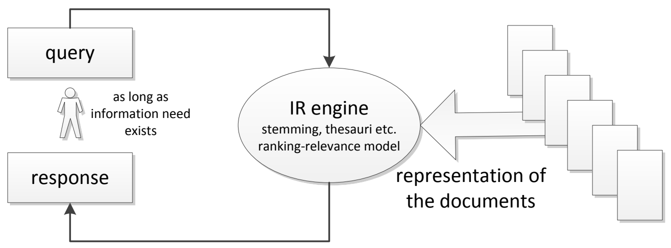


Fig. 1. The traditional information retrieval process.

II. Dataset Retrieval

Most today's human knowledge is stored in documents of many different formats, languages, depths etc. for later retrieval by humans. Information Retrieval is therefore the process of determining whether or not a document of a given corpus is relevant to current information need of a certain user [2], [3]. This process is depicted in Figure 1.

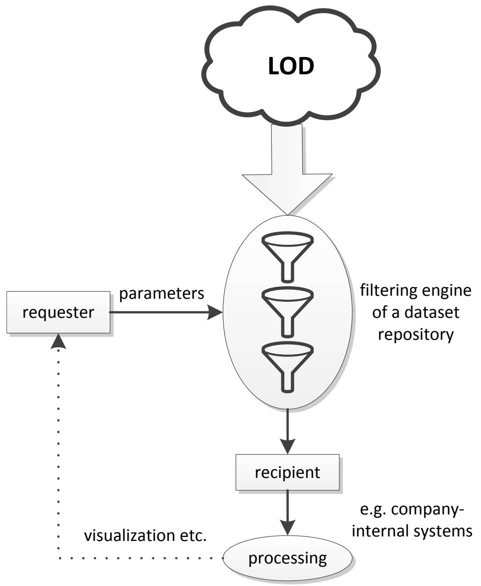
A particular way to store knowledge is an RDF dataset. An RDF dataset contains statements, where each of its distinct parts (subject, predicate, object) belongs to a certain vocabulary. A vocabulary is a set of related concepts like classes, properties, individual, datatypes etc. where each of them can be identified globally by a URI. In most cases each of these URIs have a common base URI, which is usually referred to as the URI of the vocabulary. The relationship (subclass, subproperty etc.) between the items of a vocabulary and therefore their semantics can be described by utilizing special vocabularies like the OWL or RDFS.

A strength of RDF data is the presence of several aspects that can be clearly extracted from a dataset. The most important ones are:

- *topic* – usage of vocabularies, classes, properties and individuals; even through inference,
- *location* – within a given geographical region; e.g. using Basic Geo Vocabulary,
- *time* – within a given temporal interval; using the XML Schema datatype `dateTime`,
- *language* – usage of language tags,
- *datatypes*
- *statistics* – amount of statements etc.

For the purpose of this paper, we define *dataset retrieval* as a specialization of information retrieval with the difference that instead of retrieving relevant documents the engine returns a list of relevant datasets. As with information retrieval, the term *relevance* cannot be clearly defined when using traditional methods like stemming. For RDF-based datasets, that situation is not satisfactory as the strength of RDF data is to have clear semantics in each aspect.

Another aspect one should consider is that datasets are usually made for machines for further processing like visualizations etc. whereas traditional documents are made for humans to feed a certain information need [4]. Especially the huge size of many datasets (> 100.000 statements) makes it practically impossible for humans to look through all that data efficiently by hand. We therefore propose a particular way of dataset retrieval: *dataset filtering*. It is inspired by facet search – a way to filter out the most relevant items by different aspects of the items in question. Facet search has very promising qualities in terms of flexibility and comprehensibility as Bartel et al. showed in [5]. The requester sets parameters for *semantic filters* each of which can unambiguously decide whether or not a dataset meets the given requirements. Hence, each filter reduces the amount of clearly relevant datasets. The resulting set is then passed to a recipient as can be seen in Figure 2. The recipient may be an entity that is capable of processing the filtered data. Recipient and requester may or may not be the same entity. Dataset filtering systems like dataset repositories should provide an API in order to automatically integrate filtered out datasets into the requester's systems and notify in case new datasets that match the filter parameters are published.



LOD
requester
parameters
filtering engine of a dataset repository
recipient
e.g. company-internal systems
visualization etc.
processing

Fig. 2. A special form of dataset retrieval – dataset filtering.

III. Usage Scenario

Imagine a construction company is constantly looking for new tenders. The projects shall take place in the city of its head office or within a distance of 100km from that location. The company is registered at a local dataset repository like data.gov or data.gov.uk. Let us further assume that datasets describing tenders always use the *Public Contract Ontology*[1] in order to do so. When an employee of that company is supposed to search for relevant tenders, the dataset repository provides means to query for URI of the vocabulary and the relevant region.

Since the scope of the company will not vary very rapidly, the employee shall be able to store his query for later re-use. Even an automatic notification service that makes use of the

[1]http://opendata.cz/public-contracts-ontology

the stored filter parameters can be implemented. From that perspective, we have two distinct types of filter parameters:

- *temporary* like ad-hoc searches (quick searches or testing before storing filter parameters),
- *persistent* like tong-term user interests (used for personalization).

A dataset repository should provide means to handle both types of filter parameters.

In case of datasets, traditional retrieval methods like stemming or using thesauri might not be as useful as filtering because datasets are usually not made for humans but for machines. Let A be an algorithm that can handle RDF data as input. We assume A only understands a special and therefore small set of all possible vocabularies V_A and hence only needs data that is described by using a subset of V_A. Filtering is thus sensible; ambiguous retrieval is not useful or can be harmful in case that too much useless data is given to the algorithm. In case of our construction company, there might be specialized systems for evaluating and visualizing tenders out of RDF datasets. Superfluous data only increases the processing time. Additionally, the increased response time might also have negative impact. We thus consider dataset filtering to be a pre-selection process in order to minimize negative impact on system's performance.

IV. Using Extended VoID Surrogates

In traditional information retrieval, a surrogate for each document is created and maintained. A surrogate contains vital data for identifying a document by means of keywords. When it comes to RDF datasets, each newly added dataset will also be analyzed and a concise representation of a *VoID*[2] graph – also called *VoID surrogates* – will be generated and stored in a Virtuoso RDF triplestore[3] instance for efficient querying. VoID is suitable for this task as it provides a standard way to express all vocabularies, classes and properties that appear in the dataset. We use *LODStats*, a statement-stream-based tool [6], for generating the appropriate VoID surrogates.

In order to provide the necessary data to query for a region or for a time interval, we extended VoID by adding properties for the minimum and maximum values of the literal range interval that is present a property sub-dataset:

- http://stats.lod2.eu/vocabulary/dataset#minValue
- http://stats.lod2.eu/vocabulary/dataset#maxValue

The domain of these properties are VoID datasets and the range are literal values. By this extension, the filtering engine is able to filter out not only by the presence of certain URIs but also by literal value ranges like geographical or temporal coverage. For geographical coverage, we used the *Basic Geo Vocabulary*, which is widely used for spatial tagging. For temporal coverage, we used the well-know XML Schema datatype `dateTime`. Other extensions of VoID are also conceivable like present languages or datatypes.

[2]Vocabulary of Interlinked Datasets – http://vocab.deri.ie/void

[3]OpenLink Software Virtuoso – http://virtuoso.openlinksw.com/

A VoID surrogates of an RDF dataset is the equivalent of keyword surrogates of documents in traditional information retrieval. They provide a concise yet sufficiently deep insight into the actual topic of the dataset. Furthermore, they can be generated automatically without any human interaction required and because of the clear semantics of RDF data they are at least as reliable as human-curated keyword surrogates.

V. Ranking by Dataset Similarities

In case of strict filtering, i.e. all given user requirements have to be fulfilled, ranking might not be as necessary as for the traditional information retrieval since all datasets have to be fully parsed and interpreted by domain-specific algorithms. However, ranking can also be useful when it comes to generating recommendations for datasets that do not fully adhere to the specified requirements. Since RDF data is based on the open world assumption, such datasets could contain partial useful data as well, although not all required vocabularies are used within.

Traditional retrieval techniques often determine the degree of relevance of a document by determining the degree of similarity between the query, which itself could be a document, and the document in question [7], [8]. Based on that observation, we propose an extension to that approach: a *dichotomous similarity relation*. We start by defining the similarity value $S^C(d_1, d_2) \in \mathbb{R}^+$ of two datasets d_1 and d_2. S^C describes to which degree both datasets are similar to each other or how related they are. C stands for one of the corresponding aspects (topic T, location L, time Z etc.) as different aspects cannot be intermingled[4]. Furthermore, we assume $S^C(d_1, d_2) = S^C(d_2, d_1)$, i.e. S^C is symmetric. In the sequel, we define S^T, S^L and S^Z.

A. Similarity Value of Topical Aspect

The topical aspect T of a dataset resembles the keyword-based perspective of traditional information retrieval. Two datasets are the more similar the more frequently they use concepts from the same vocabularies. The restriction to vocabularies (instead of concepts like classes, properties etc.) can improve calculation performance since most datasets use at least one concept of a vocabulary but never less. Otherwise, the vocabulary would not be present at all in that dataset. Moreover, vocabularies represent closely related concepts, which are likely to co-occur when describing a certain subject. We thus define S^C as:

$$S^C(d_1, d_2) = \sum_{v \in V} w(v) g(d_1, v) g(d_2, v) \tag{1}$$

where V is the set of all vocabularies used by both datasets. g is defined as follows:

$$g(d, v) = \begin{cases} 1, \text{if } d \text{ uses concepts of } v \\ 0, \text{otherwise} \end{cases} \tag{2}$$

[4]Which pair of datasets is more similar: two datasets describing entities in a distance of 15km or two with 15 shared vocabularies?

The weight function $w(v)$ represents the importance of the given vocabulary. One could set $w(v) = 1$ in order to weigh all vocabularies alike. However, as traditional information retrieval already has shown, not every keyword matters the same [9]. There are especially such keywords that provide no clue about the actual topic of the document such as stopwords like forms of 'to be', prepositions etc. whereas some of them do. One way to calculate the weight of a keyword is the inverse document frequency (IDF) [10]. We transfer this concept to vocabularies, which assume the same role as keywords do in traditional information retrieval. The general observation is that the more frequent a vocabulary is the less specific and the less significant it is. One example of a very non-specific vocabulary is the RDF vocabulary itself, Dublin Core Terms, RDFS or OWL only to name a few. Statistics from LODStats[5] back that observation. Given that, we can define $w(v)$ as follows:

$$w(v) = -\log q(v) \tag{3}$$
$$q : V \to [0, 1] \tag{4}$$
$$q(v) = \frac{|D_v|}{|D|} \tag{5}$$
$$D_v = \{d \in D : g(d, v) = 1\} \tag{6}$$

$q(v)$ is the relative frequency of a vocabulary v and D is the set of all datasets. In fact, only those vocabularies that matter for a particular dataset corpus will be considered in the calculation of $w(v)$. Since the overall usage of a vocabulary only changes very slowly, $w(v)$ can be pre-calculated once in a while in order to reduce computation time.

One could argue that not only the IDF but also the *term frequency* (TF) should be included into the calculation as well. The first occurrence of a relevant URI (keyword) is the strongest indicator whether a dataset is relevant to a given URI query [10]. Additional occurrences only lead to small increases in many relevance models. When considering the actual structure of most datasets, we assume that TF would usually not lead to enhancing retrieval performance. RDF datasets typically have the following structure:

- lists of similar objects (same class, same set of properties etc.)
- and additional metadata

Given that, most relevant URIs should be present more than once. Furthermore, using another TF function than g would break the symmetry property.

B. Similarity Value of Geographical Aspect

When it comes to the geographical aspect of a dataset, we need to consider two different situations: first, two datasets share a common geographical region; second, they do not. The reason for that distinction is the different qualities of the relationship between dataset pair P_W (they share a common area) and dataset pair P_D (they do not share a common area but have a certain distance in between). Since the principle of locality holds for most macrophysical objects like earth, we assume a greater similarity for P_W than for P_D as P_W describe partially the same physical area.

It is sensible to distinguish between local datasets (relatively small area) and global datasets (relatively large area). Determining the similarity values of global datasets might make actually no sense as they cover vast area of the globe with the result that a similarity value has no real meaning. Furthermore, we assume a bounding circle for each dataset that represents the actual geographical region of that dataset. In case of global datasets, the necessary parameters for the bounding circle model and the resulting bounding circle itself become very inaccurate and questionable as well.

Informally, we can define S^{LW} and S^{LD} as:

$$S^{LW} := \text{area of the shared geographical region} \tag{7}$$
$$S^{LD} := \text{smallest distance between the datasets} \tag{8}$$

Although using a simple model for the geographical region of a dataset, the required computations can be very costly. We therefore use the diameter instead of the real area, which is a monotonically increasing function of the diameter. We therefore approximate S^{LW} and S^{LD} as:

$$S^{LW}(d_1, d_2) \approx \frac{\min(2r_{min}, -\delta(d_1, d_2))}{2r_{max}} \tag{9}$$
$$S^{LD}(d_1, d_2) = \delta(d_1, d_2) \tag{10}$$
$$\delta(d_1, d_2) = \Delta(d_1, d_2) - (r_{d_1} + r_{d_2}) \tag{11}$$
$$\Delta(d_1, d_2) - \text{distance of the centers} \tag{12}$$
$$r_{min} = \min(r_{d_1}, r_{d_2}) \tag{13}$$
$$r_{max} = \max(r_{d_1}, r_{d_2}) \tag{14}$$
$$r_d - \text{radius of the bounding circle of } d \tag{15}$$

We furthermore require only one of these values to be valid:

$$S^{LW}(d_1, d_2) > 0 \tag{16}$$
$$S^{LD}(d_1, d_2) \geq 0 \tag{17}$$

C. Similarity Value of Temporal Aspect

Two datasets are the more similar the smaller the temporal distance is or the greater the shared temporal region is. For the very same reason as with geographical similarity, we need to distinguish between a dataset pair P_W with shared time interval(s) and a dataset pair P_D without one. Thus, we can define S^{ZW} and S^{ZD}:

$$S^{ZW} := \text{amount of shared time} \tag{18}$$
$$S^{ZD} := \text{smallest time span between the datasets} \tag{19}$$

In our implementation for CKAN, we used the time interval model for a dataset. Each dataset's temporal coverage will be represented by one time interval consisting of the minimum and maximum point in time that is described by the dataset.

[5]LODStats vocabulary statistics – http://stats.lod2.eu/vocabularies?sort=datasets

Precisely, our computation looks like this:

$$S^{ZW}(d_1, d_2) = \frac{\underline{t} - \overline{t}}{\max(\Delta^1, \Delta^2)} \tag{20}$$

$$S^{ZD}(d_1, d_2) = \overline{t} - \underline{t} \tag{21}$$

$$\underline{t} = \min(t^1_{max}, t^2_{max}) \tag{22}$$

$$\overline{t} = \max(t^1_{min}, t^2_{min}) \tag{23}$$

$$\Delta^1 = t^1_{max} - t^1_{min} \tag{24}$$

$$\Delta^2 = t^2_{max} - t^2_{min} \tag{25}$$

Additionally, we require:

$$S^{ZW}(d_1, d_2) > 0 \tag{26}$$

$$S^{ZD}(d_1, d_2) \geq 0 \tag{27}$$

D. Dichotomous Similarity Relation

S^T, S^{LW} and S^{ZW} are called weight-based similarity values. They represent the relationship of shared elements among the datasets (same vocabularies, same regions, same time). A weight-based order relation $R^W_d \subseteq D \times D$ can be constructed as follows:

$$(d_1, d_2) \in R^W_d \Leftrightarrow S^C(d, d_1) \leq S^C(d, d_2) \tag{28}$$

S^{LD} and S^{ZD} each define a distance-based order relation $R^D_d \subseteq D \times D$ for each dataset d by the following construction:

$$(d_1, d_2) \in R^D_d \Leftrightarrow S^C(d, d_1) \geq S^C(d, d_2) \tag{29}$$

Given R^W_d and R^D_d, we can construct the further order relation $R^C_d \subseteq D \times D$:

$$(d_1, d_2) \in R^C_d \Leftrightarrow \big((d_1, d_2) \in R^W_d\big) \vee \big((d_1, d_2) \in R^D_d\big) \vee \big(\forall e : (e, d_1) \in R^D_d\big) \wedge \big(\forall f : (d_2, f) \in R^W_d\big) \tag{30}$$

One can interpret R^C_d as follows: d_2 is more similar to d than d_1, if and only if d_2 is distance-based or weight-based more similar to d than d_1. The lower part of the construction ensures transitioning from distance-based to weight-based similarity. Datasets with a weight-based similarity relation to d are always more similar to d than those with a distance-based one, i.e. *part-of-other* relationships will be valued higher than *being-a-neighbor* relationships. We call R^C_d a dichotomous similarity relation as it consists of two disjoint sub-relations.

VI. Using Similarities for Recommendations

The purpose of the dichotomous similarity relation of the last section is to efficiently create a relevance ranking used when recommending datasets to the user; automatic dataset retrieval. In order to avoid overspecialization[6], we used two different methods in our CKAN implementation for creating such recommendations:

- dataset-based recommendations
- personalized recommendations

First, dataset-based recommendations are presented on each dataset page. For each dataset aspect, the user is given a list of the 5 most similar datasets to the currently viewed dataset. Second, personalized recommendations can be generated on basis of the pre-calculated similarity relation, too. It is moreover important to present the *reasons* why a certain dataset has been recommended. That is necessary in order to have the user in control and make it possible for him to decide whether or not he should modify his preferences. We propose the following algorithm for generating personalized recommendations and their reasons:

We model the user interests as the set I of explicitly relevant datasets (the user showed explicit interest in them), explicitly relevant users and explicit relevant properties of datasets (relevant URIs, geographical regions and time intervals).

$$I = I_D \cup I_U \cup I_S \tag{31}$$

$$I_D - \text{relevant datasets} \tag{32}$$

$$I_U - \text{relevant users} \tag{33}$$

$$I_S - \text{relevant property sets} \tag{34}$$

We start with the set D of all datasets and reduce the number of items to be recommended step by step. Since the user already familiarized himself with the datasets of I_D and the datasets that have the explicitly stated relevant properties, we can remove them:

$$D^{(1)} := (D \setminus I_D) \setminus \bigcup_{s \in I_S} d(s) \tag{35}$$

$$d(s) - \text{datasets that have properties s} \tag{36}$$

$D^{(1)}$ is the set of datasets that is likely to be unknown to the user. We now have to find an order of $D^{(1)}$ from which we can pick the k most suited and present them to the user. Most suited are those where the user can comprehend the recommendation. A recommendation is the more comprehensible the more *selective user interests* it has. A user interest is selective for a dataset d when it has a high similarity value to d:

$$v(i) - \text{datasets for which i is selective} \tag{37}$$

$D^{(2)}$ is the set of to be recommended datasets and can be constructed as follows:

$$D^{(2)} := \bigcup_{i \in I_D \cup I_S} v(i) \tag{38}$$

The set of reasons $b(d)$ for each recommended dataset d follows easily by:

$$b : D^{(2)} \rightarrow I \tag{39}$$

$$b(d) = \{i \in I : d \in v(i)\} \tag{40}$$

We can now define an order $R_V \subseteq D^{(2)} \times D^{(2)}$ on $D^{(2)}$:

$$(d_1, d_2) \in R_V \Leftrightarrow |b(d_1)| \geq |b(d_2)| \tag{41}$$

R_V orders $D^{(2)}$ by the number of reasons $|b(d)|$ for each recommended dataset d.

[6]Users only see datasets of their explicit interests.

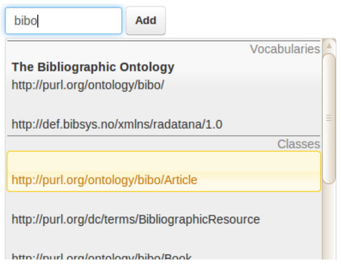

bibo
Add
Vocabularies
The Bibliographic Ontology
http://purl.org/ontology/bibo/
http://def.bibsys.no/xmlns/radatana/1.0
Classes
http://purl.org/ontology/bibo/Article
http://purl.org/dc/terms/BibliographicResource

Fig. 3. List of URI suggestions appears while typing in the URI filter text field. LOV provided a label for the vocabulary.

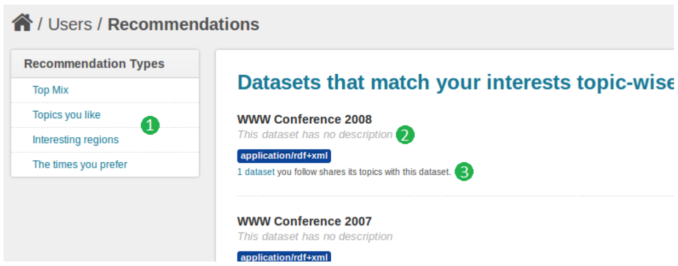

/ Users / Recommendations
Recommendation Types
Top Mix
Topics you like
Interesting regions
The times you prefer
1
Datasets that match your interests topic-wise
WWW Conference 2008
This dataset has no description
2
application/rdf+xml
1 dataset you follow shares its topics with this dataset.
3
WWW Conference 2007
This dataset has no description
application/rdf+xml

Fig. 4. Fragment of the recommendations page. (1) list of different dataset aspects; (2) recommended dataset; (3) reason for recommending that dataset.

VII. Technology and Implementation Details

This section elucidates details of the implementation extending the functionality of the well-known dataset repository software CKAN[7]. The implementation is split into two different parts: CKAN core and a CKAN extension. The CKAN core implementation enables CKAN to store search queries (subscription API) and further personalization features. The point in integrating subscriptions into the search API is the principle of trial and error. A user can test out several search queries *before* he has to store the search query and thus receive notifications and recommendations based on that decision. The CKAN extension *ckanext-semantic*[8] extends searching by adding filters for relevant vocabularies, classes or properties, geographical regions and time spans. In order to assist the user, we provide URI suggestions based on what the user types in the URI filter text field. These suggestions are generated dynamically from all used vocabularies, classes, properties and the vocabularies provided by *Linked Open Vocabularies* (LOV)[9]. LOV furthermore provides labels for their vocabularies which we additionally utilize for querying URIs and displaying them (cf. Figure 3). Since VoID is an RDF vocabulary itself, the VoID surrogates are RDF data themselves. Thus, we can query that data not only by predefined SPARQL queries but also by a custom SPARQL query. Such queries can be stored via the subscription API, too.

We followed a *two-phased approach* in order to reduce response times. In the offline phase, periodical LODStats jobs generate a VoID surrogate for each new dataset and for each dataset whose VoID surrogate is too old. A suitable RDF representation of a dataset will be retrieved and analyzed by LODStats. The VoID surrogates are then stored in a Virtuoso RDF triplestore instance for efficient querying afterwards in the online phase. Recommendation lists and the supporting reasons generated from user interests by the procedure described in the previous section can be seen in Figure 4. The necessary similarity values are calculated lazily by the formulas presented in section VI and stored in the triplestore as well. We used the *Similarity Ontology*[10], which is eminently suitable for storing a dichotomous similarity relation.

[7]http://ckan.org/
[8]Source Code – https://github.com/srkunze/ckanext-semantic/
[9]LOV – http://lov.okfn.org/dataset/lov/
[10]http://kakapo.dcs.qmul.ac.uk/ontology/musim/0.2/musim.html

A notification service has also been implemented and it distinguishes between online notifications and offline notifications. Online or onsite notifications are represented by highlighted numbers. These numbers indicate the number of unseen relevant activities—activities that appertain to relevant datasets and users of the CKAN instance—and try to attract the user's attention. Additionally, the offline notification service regularly inspects the activity database for unseen relevant activities and notifies the user via emails provided that the user wished to receive them.

VIII. Discussion and Evaluation

In traditional information retrieval, we use metrics like precision and recall in order to determine the retrieval system's performance. *Precision* describes the ratio of relevant found documents to all found documents whereas *recall* is the ratio of relevant found documents to all relevant documents. There are numerous other metrics to measure the performance of a retrieval system. Most systems usually achieve a decreasing precision with an increasing recall and vice versa. Those metrics were introduced because of the uncertainties natural language usually comes with. They result from the ambiguity of natural-language words and imprecision and semantics changing of natural-language constructions. It is thus quite difficult or even impossible to determine whether a given natural-language statement describes the requested concepts [3]. Since most documents utilize at least one natural language to express the knowledge within, retrieval systems inherit these uncertainties and have to deal with them as well.

Metrics like precision and recall might not be as sensible for dataset filtering as they are for traditional retrieval systems because they do not serve their intended purpose. The key aspect here is the term *relevance*. We can clearly detect whether or not a given relevant URI is present within RDF data, in particular when a triple contains such URL either as subject, as predicate or as object. In that case, the entire dataset is relevant to the user as this very triple might contain information vital to the requester. We can even determine whether or not particular relevant triple patterns are within a dataset. Thus, when using such system as proposed in this paper, the term *relevance* is clearly defined and has no scope of interpretation. We therefore would automatically attain a precision and recall of 1.0 which is indeed not satisfying for evaluating the performance of a system.

We therefore need different metrics in order to determine the performance of a dataset filtering system. One way to start analyzing such systems is classifying them by certain capabilities. Our classification scheme distinguishes several *query types* – semantic properties by which a user can filter the dataset corpus:

- URIs
- URI patterns (unknown parts of URIs)
- literals
- literal patterns (unknown parts of literals)
- triple patterns (unknown parts of triples)
- graph patterns (contains graphs of given structure)
- SPARQL-like (complex graph patterns, filters, aggregations etc.)
- domain-specific (like geo, time, languages etc.)

Since there is no scope of interpretation, we assume 100% recall and precision in traditional sense for each method when a system supports one of those methods. The way the user interacts with the system and is assisted when entering URIs, literals, patterns etc. is more important. Especially, retrieving potentially relevant URIs, literals, triple patterns etc. from existing datasets and user queries is challenging. We thus propose measuring the quality of those suggestions rather than evaluating the filtering process. Our approach supports URI suggestions generated from the VoID surrogates and LOV. Triple and graph patterns are possible via the SPARQL query interface, which might impose a steep learning curve for most users, though. Since geographical and temporal filters include domain-specific elements like maps, the majority of users can easily use them.

Besides the quality aspect, the runtime performance is also a key factor influencing the users retrieval experience. We performed our tests on a virtual machine (VirtualBox 4.2.12) running Ubuntu 10.04 LTS with 1,5GB memory and one core with 2.8GHz. Figure 5 depicts the resulting runtimes of several queries with 100, 400, 700 and 1000 datasets published in the test CKAN instance. Each runtime increases linearly with the number of datasets. Querying datasets by a URI (A) takes longer than querying datasets within bounding circle (B and C). It seems that the more complex SPARQL query that is necessary for (A) causes that behavior. Interestingly, when we increase the bounding radius from 50km to 500km response times also increase. Because the only difference is the number of found datasets, we suspect the post-processing of the CKAN core to be the reason for that increase. While the testing setup might not reflect a real production environment, we can achieve reasonable response times with the number of RDF datasets available in present dataset repositories. In the future, it will become necessary to investigate optimizations in order to increase the number of datasets without sacrificing response times and therefore jeopardizing the user experience. Our design decision of running (E) in the offline phase is supported by the extremely fast runtime of the current implementation of that operation.

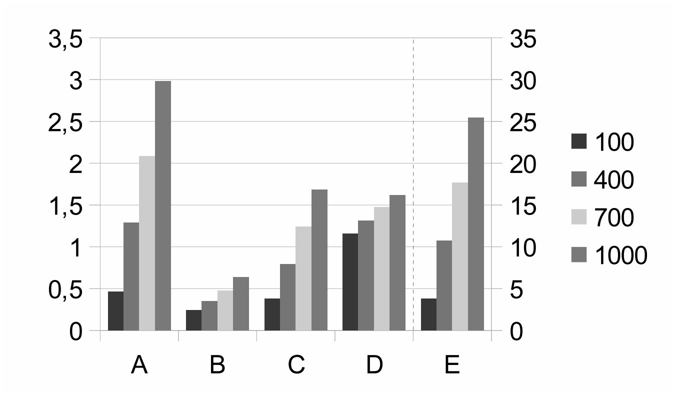


Fig. 5. Performance measurements; scale in seconds; (A) search for 1 URI; (B) search for random location within a distance of 50km; (C) search for random location within a distance of 500km; (D) querying similar datasets; (E) updating similarity values.

IX. Related Work

Handling large amount of RDF data becomes more and more important for users of the Semantic Web. One way of dealing with the problems of very large datasets is to let compute queries results in parallel like Farhan Husain et al. proposed in [11]. They used Hadoop and MapReduce in order to distribute the computation effort of evaluating SPARQL queries on very large RDF datasets. From the retrieval perspective, however, that solution is not satisfactory because of the computational effort required to evaluate big corpora of large datasets only to determine whether or not some of them are relevant. Another way of dealing with those issues has been investigated by Fernandez et al. [12]. They found that most of current RDF datasets lack meta information like provenance data for efficient retrieval. They proposed a special storage scheme (*HDT*[11] – Header, Dictionary, Triples) for enhancing storing and processing RDF data [13]. Retrieval of datasets can be accelerated because the necessary meta data is stored at the beginning of the dataset. We, however, think that data providers are not solely responsible for providing additional metadata like statistical or provenance data as their usefulness depends heavily on the use-cases that the data has been designed for. From the perspective of a dataset repository, data users and the dataset repository itself are responsible for evaluating and processing the acquired data accordingly to their needs. Especially when considering the huge datasets, providing different representations and access points will become quite infeasible for data providers.

Sindice[12] is a Semantic Web search engine. As of writing, it has indexed over 700,000 Web documents containing RDF data by collecting resource URIs, inverse-functional properties and keywords [14]. The user can enter keywords and URIs via text input field and is presented a list of all Web resources whose RDF data contains the entered keywords partially or completely. We find, albeit that approach follows the afore-

[11]HDT – http://www.w3.org/Submission/2011/SUBM-HDT-20110330/
[12]http://sindice.com/

mentioned philosophy of separating data creation/provision and data usage, not to rely solely on URIs or other semantically clear search arguments do not live to the expectations of a Semantic Web search. Tran et al. describe an approach of creating structured (exact) queries (SQL, SPARQL etc.) of unstructured (inexact) user queries from which the user shall choose one to be executed [15]. User queries indeed reflect a structured user information need. Nevertheless, many users may not be capable of deciding whether a structured query represents their very need. In [16] Elbassouni et al. propose a method of generating keywords of RDF data by splitting the subject, the predicate and the object of each statement into human-readable keywords. The core assumption here is that URIs usually contain human-readable strings. Users can then query an RDF graph by such keywords. As this method offers a simple way for most users to query a RDF dataset, it cannot provide semantically unambiguous queries.

X. Conclusion

The goal of this work was to define the terms dataset retrieval and dataset filtering as subconcepts of the traditional information retrieval. Based upon that, we presented a usage scenario for dataset filtering and the associated added value for the user. Unlike traditional information retrieval systems, a dataset filtering system can rely on the semantic integrity of the RDF data to be retrieved and therefore have better ways in handling user requests without having to cope with the ambiguity of natural language. Because of huge sizes of today's datasets, datasets cannot be handled manually and dataset filtering is a necessary pre-selection step for the further automated processing of RDF data. We proposed VoID surrogates and an extension to the VoID vocabulary for efficient dataset filtering by URIs, SPARQL queries and domain-specific aspects of RDF datasets. Beyond that, we investigated how to implement automated dataset filtering using a dichotomous similarity relation consisting of a weight-based and a distance-based part. For each part, we presented formulas for the corresponding similarity values of topical, geographical and temporal aspects of RDF datasets. We implemented these concepts in form of an extension for the popular dataset repository software CKAN.

Future research includes analysis of the actual content and structure of real-world RDF datasets for a better filtering process. Even splitting larger datasets into smaller ones in order to accelerate after-retrieval processing is a sensible option which requires a more in-depth investigation. Moreover, metrics for measuring the quality of both URI retrieval and pattern generation needs to be investigated along with more expressive and user-friendly query types other than URL (patterns), triple (patterns), graph patterns, and SPARQL-like queries. That becomes necessary when involving user groups other than those experienced in writing RDF queries efficiently.

Acknowledgment

We would like to thank Sean Hammond and John Martin for assisting us during implementing the proposed concept.

References

[1] E. Sormunen, J. Kekäläinen, J. Koivisto, and K. Järvelin, "Document text characteristics affect the ranking of the most relevant documents by expanded structured queries," *Journal of Documentation*, vol. 5, no. 3, pp. 358–376, 2001.

[2] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1999.

[4] T. Berners-Lee. (2009, 3) Tim berners-lee on the next web. [Online]. Available: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

[5] T. Bartel, G. Quint, K. Reinhard, and S. Weichert, "Faceted Search: Die neue Suche aus Usability-Sicht," in *Usability Professionals 2009: Berichtband des siebten Workshops des German Chapters der Usability Professionals Association e.V.* Stuttgart, NRW, Germany: Fraunhofer IRB Verlag, 2009, pp. 40–45. [Online]. Available: http://www.usability.de/publikationen/studien/faceted-search.html

[6] J. Demter, S. Auer, M. Martin, and J. Lehmann, "LODStats – An Extensible Framework for High-performance Dataset Analytics," in *Proceedings of the EKAW 2012*, ser. Lecture Notes in Computer Science (LNCS) 7603. Springer, 2012.

[7] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th international conference on Machine Learning*, vol. 1. San Francisco, 1998, pp. 296–304.

[8] G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. Petrakis, and E. E. Milios, "Semantic similarity methods in wordNet and their application to information retrieval on the web," in *Proceedings of the 7th annual ACM international workshop on Web information and data management*. ACM, 2005, pp. 10–16.

[9] M. J. Nelson, "Correlation of term usage and term indexing frequencies," *Information Processing & Management*, vol. 24, no. 5, pp. 541–547, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0306457388900234

[10] S. E. Robertson, "Understanding Inverse Document Frequency: on theoretical arguments for IDF," *Journal of Documentation*, vol. 60, pp. 503–520, 2004.

[11] M. Farhan Husain, P. Doshi, L. Khan, and B. Thuraisingham, "Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce," in *Cloud Computing*, ser. Lecture Notes in Computer Science, M. Jaatun, G. Zhao, and C. Rong, Eds. Springer Berlin Heidelberg, 2009, vol. 5931, pp. 680–686. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10665-1_72

[12] J. D. Fernández, M. A. Martínez-Prieto, and C. Gutierrez, "Compact Representation of Large RDF Data Sets for Publishing and Exchange," in *The Semantic Web ISWC 2010*, ser. Lecture Notes in Computer Science, P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, Eds. Springer Berlin Heidelberg, 2010, vol. 6496, pp. 193–208. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17746-0_13

[13] J. D. Fernández, M. A. Martínez-Prieto, M. Arias, C. Gutierrez, S. Álvarez García, and N. R. Brisaboa, "Lightweighting the web of data through compact RDF/HDT," in *Proceedings of the 14th international conference on Advances in artificial intelligence: spanish association for artificial intelligence*, ser. CAEPIA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 483–493. [Online]. Available: http://dl.acm.org/citation.cfm?id=2075561.2075617

[14] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: a document-oriented lookup index for open linked data," *Int. J. Metadata Semant. Ontologies*, vol. 3, no. 1, pp. 37–52, 11 2008. [Online]. Available: http://dx.doi.org/10.1504/IJMSO.2008.021204

[15] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (rdf) Data," in *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, 4 2009, pp. 405–416.

[16] S. Elbassuoni and R. Blanco, "Keyword search over RDF graphs," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, ser. CIKM '11. New York, NY, USA: ACM, 2011, pp. 237–242. [Online]. Available: http://doi.acm.org/10.1145/2063576.2063615