Enabling a Unified View of Open Data Catalogs

Marcelo Iury S. Oliveira[1,2], Lairson Emanuel R. de Alencar Oliveira[2],
Glória de Fátima A. Barros Lima[2] and Bernadette Farias Lóscio[2]
[1]*Academic Unit of Serra Talhada, Federal Rural University of Pernambuco, Serra Talhada - PE, Brazil*
[2]*Center for Informatics, Federal University of Pernambuco, Recife – PE, Brazil*

Keywords: Open Data, Catalog, Metadata, DCAT, Management, Standardization, Design.

Abstract: In this article we present a solution, called DataCollector, which allows the cataloging and the discovery of data distributed in multiple open data portals. Our solution collects metadata about datasets available in multiple open data portals and it offers a uniform interface to access them. The proposed solution was evaluated by its viability in cataloging 14 Brazilian open data portals, covering a total of 29,540 datasets. The preliminary results indicate the DataCollector offers a robust solution for cataloging and access to distributed datasets in multiple platforms for open data publication.

1 INTRODUCTION

The amount of OGD available as well as their data consumers and producers are growing at a fast pace around the world. Several governments have launched initiatives to stimulate and promote open data usage and production (Chun et al., 2010). However, as open data is fairly recent movement, most of open data available are not properly known (Barbosa et al., 2014). Despite government domain is one of the most important sources for open data, there is a vast amount of open data being published in other areas, including: environment, cultural, science, and statics. In this context, where open data comes from many sources and are created in an autonomous way, it becomes necessary to have an easy way to find and to combine these distributed data. Providing tools to perform high-level analysis on top of these data is also important requirement. While developers are more interested on the raw data, policy and decision makers will profit from the analysis tools.

In this article we present a solution, called *DataCollector*, for cataloging and searching of data published in open data portals. Our solution collects metadata about datasets available in multiple open data portals and it offers a uniform interface to access them. The proposed solution also offers an easy way to query data based on both graphical interfaces and remote access mechanisms, such as Web Services, which enable automated processing and facilitate searching activities. In addition, our proposal enables users collaboration and provides features to collect and manage feedback about the open data. Finally, to promote interoperability, DataCollector uses the standardized vocabularies such as DCAT vocabulary (Data Catalog Vocabulary)(Maali et al., 2014) to describe its data catalog.

Besides facilitating dataset selection and access, our solution provides tools to perform datasets analysis. Using DataCollector, it is possible to visualize the domains of the datasets as well as the domains with higher number of datasets, for example. Other aspects like usage license and data formats can also be analysed. The DataCollector may be used to perform analysis about open data scenarios as well as to facilitate the automatic identification of relevant datasets for a given user or application. Considering the growing number of open data, these tasks become time consuming and therefore very expensive.

The remainder of this article is organized as follows. In Section 2 we discuss some related work. In Section 3 we present our Metadata Model. In Section 4 we present a general overview of the proposed solution, contextualizing the services and main components of the DataCollector. In Section 5 we present more details about the implementation of the DataCollector. In Section 6 we discuss the evaluation of our proposal and in Section 7 we present the final considerations and future work.

Oliveira, M., Oliveira, L., Lima, G. and Lóscio, B.
Enabling a Unified View of Open Data Catalogs.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 2*, pages 230-239
ISBN: 978-989-758-187-8

2 RELATED WORK

The use of open data has grown substantially in the last years. In general, open data is available in catalogs, which provide an interface between who makes data available (publisher) to those interested in using them (consumers). Architectures and platforms for data cataloging, as well as the harvesting of data portals, have been the subject of some works reported in the literature or solutions available on the Web (Ribeiro et al., 2015; Foundation, 2015a; Miranda, 2015; Foundation, 2015b; Project, 2015).

In this context, CKAN[1] and SOCRATA[2] are the main solutions currently adopted for open data cataloging. CKAN is an open source software, which provides support for the creation of data portals, as well as offers features for publishing, storage and management of datasets (van den Broek et al., 2012). CKAN provides an Application Programming Interface (API) for automated access and has functionalities for data previewing, creation of graphs and maps and the searching of geolocated datasets. The Socrata, instead, is a proprietary solution based on cloud computing and Software as a Service paradigms. One of the main differentials of Socrata is the possibility of building more complete data visualizations (conditional formatting, graphs and maps) (Miranda, 2015).

DataHub.io (Foundation, 2015a) is an open source platform for data management provided by the Open Knowledge Foundation and powered by CKAN. DataHub.io allows everybody to host their data and is also widely used to aggregate datasets published elsewhere in one place. Despite DataHub.io provides a single point to find datasets published in different sites, it strongly depends on the cooperation of users to publish data and it doesn't offer additional support for data analysis or datasets evaluation. PublicData.eu (Foundation, 2015b) and OpenDataMonitor.eu (Project, 2015) have been proposed to offer access to multiple open data portals at the same time. These data platforms provide access to datasets published by governmental agencies all over Europe. In particular, the OpenDataMonitor also provides automated evaluation of datasets.

Different from the solutions mentioned above, the DataCollector may be instantiated to collect data about different sets of data portals, *e.g.*, it can be used to collect datasets from brazilian open government data portals, as well as it could be used to collect datasets from biological data portals. Besides collecting datasets, the DataCollector provides free access to its metadata and also allows performing analysis about the collected datasets.

Finally, it is worth mentioning the UrbanProfiler (Ribeiro et al., 2015), which is a tool that automatically extracts detailed information about datasets to help datasets selection. UrbanProfiler aims to collect information to help data consumers to explore and understand the contents of a dataset. In a similar way, we also offer information about the datasets in order to help datasets identification and selection.

3 METADATA MODEL

As previously described, the Data Collector offers an unified view of open datasets available in several open data portals or catalogs. Considering the heterogeneous nature of the available data portals, in order to offer this uniform view it becomes necessary adopt of a common data model to describe the data catalogs and available datasets. In our work, we use the DCAT vocabulary (Maali et al., 2014) to overcome heterogeneity of the open data portals being integrated.

DCAT is a RDF vocabulary, proposed by the W3C, which seeks to facilitate interoperability between data catalogs. In general, vocabularies are used to classify terms of a particular application, characterize possible relationships and define restrictions on the use of these terms. Specifically, DCAT defines the concepts and relationships used in the field of data catalogs, as catalogs and datasets. DCAT also incorporates terms of other existing vocabularies, in particular, the *Dublin Core* [3] and *FOAF* [4]. Figure 1 presents the main classes of the DCAT.

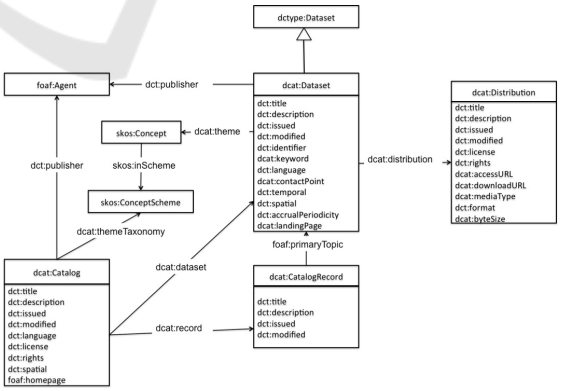

Figure 1: DCAT model.

The central entity of the DCAT vocabulary is the Dataset, which is a collection of data, available for access or download in one or more formats (Maali et al.,

[1] http://www.ckan.org
[2] http://www.socrata.com
[3] http://www.dublincore.org
[4] http://www.xmlns.com/foaf/spec

2014). Datasets are described by general metadata, *e.g.* publication date, keywords, language (Maali et al., 2014), and other information that makes it easier to discover and use dataset.

A single dataset may be available in different Distributions to meet interoperability and usage requirements of different users. These distributions might represent different formats of the dataset or different endpoints (Maali et al., 2014). Examples of distributions include a downloadable CSV file, an API or an RSS feed. Distributions are also described by metadata. The format property (*dct:format*), for example, specifies data format used in the distribution. In order to ensure interoperability, the format should be a standard MIME type, like text/csv or application/json. The accessURL (*dcat:accessURL*) property determines a resource that gives access to the distribution of dataset, like a landingpage or a SPARQL endpoint. The downloadURL property (*dcat:downloadURL*) represents a file that contains the distribution of the dataset in a given format. Other relevant metadata about a distribution include the size of a distribution in bytes (*dcat:byteSize*) and the license (*dct:license*) under which the distribution is made available.

Finally, a Catalog is defined as a curated collection of metadata about datasets. Catalogs are also described by several properties, including: date of publication, date of modification, language used in the textual metadata describe datasets and the entity responsible for online publication of the catalog (Maali et al., 2014). A record in a data catalog, describing a single dataset, is represented as an instance of the CatalogRecord class.

In our work, besides concepts offered by DCAT, we also use other vocabulary proposals to model feedback collected from data consumers about specific datasets. In general, data cataloguing solutions don't offer means to collect and to share feedback about datasets. However, gathering such information produces benefits for both producers and consumers of data, contributing to improve the quality of the published data, as well as to encourage the publication of new data.

The feedback data is modeled using two vocabularies have been developed by the W3C Working Group: Dataset Usage Vocabulary[5] and Web Annotation Data Model[6]. The former one aims to define a vocabulary that offers concepts to describe how datasets published on the Web have been used as well concepts to capture data consumer's feedback and dataset citations. The later describes a structured model and format to enable annotations to be shared and reused across different platforms.

Figure 2 presents the main classes used to model feedback. *duv:Usagefeedback* allows to capture consumer's feedback in the form of annotations. As described in the Web Annotation Data Model(Sanderson et al., 2015), an Annotation is a rooted, directed graph that represents a relationship between resources, whose primary types are Bodies and Targets. An Annotation *oa:Annotation* has a single Body, which is a comment or other descriptive resource, and a single Target refers to the resource being annotated. In our context, the target resource is a dataset or a distribution. The body may be a general comment about the dataset, a suggestion to correct or to update the dataset, a dataset quality evaluation or a dataset rating. The property *oa:motivation* may be used to explicitly capture the motivation for a given feedback. For example, when the consumer suggests a dataset update then the value of oa:motivation will be "reviewing".

duv:RatingFeedback and *dqv:UserQuality Feedback* are subclasses of *duv:UsageFeedback*. The first one allows to capture feedback about the quality of the dataset, like availability, consistency and freshness[7]. The second one allows consumers to evaluate a dataset based on a grade or a star schema, for example.

Feedback could be captured from data consumers by rating questionnaires, where users are asked to provide a value point on a fixed scale (Amatriain et al., 2009). This interaction model could be viewed as a form of Web collaboration in such way that datasets evaluation is accomplished by engaging data consumers as "processors" to annotate datasets according taxonomies/folksonomies subjects, for example. Moreover, a data consumer can also provide feedback annotations to fulfill missing metadata about datasets. Finally, a user may submit a report based on the problems they have witnessed when consuming the data. In general, the annotation model provides the user with a added value for little effort, because it facilitates finding a desirable dataset, as well as, becomes easier to find similar new resources that could be of interest (Hotho et al., 2006).

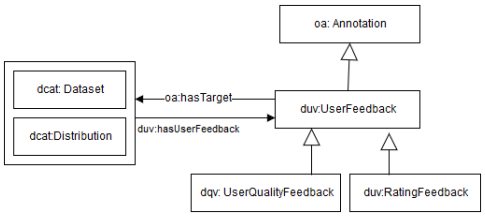

Figure 2: Feedback model.

[5]http://www.w3.org/TR/vocab-duv/

[6]http://www.w3.org/TR/annotation-model/

[7]http://www.w3.org/TR/vocab-dqv/

4 DATACOLLECTOR ARCHITECTURAL OVERVIEW

The DataCollector is a Web platform that aims to provide cataloging and searching of open data stored in multiple distributed open data catalogs or portals. In order to achieve its goals, the DataCollector defines several loosely coupled modules that compose its logical architecture, as seen in Figure 3 and described in the following.

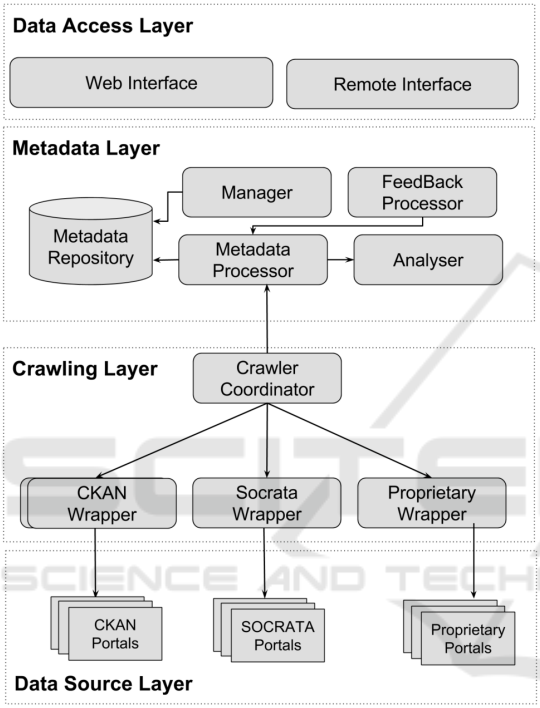

Figure 3: Portal Architecture.

4.1 Data Source and Crawling Layers

The *Data Source Layer* is composed by the data portals registered on the DataCollector. Each data portal is considered as a data source that provides one or more datasets. As mentioned before, data portals are usually implemented based on platforms such as CKAN and Socrata. However, we also consider data portals that use other technologies, including content management systems like Drupal[8] and Wordpress[9].

The *Crawling Layer* is responsible for collecting metadata from data sources and representing them according to the DCAT data model. In general, metadata extraction is an automatic task. However, in cases such as lack of API access or lack of structured metadata, manual creation of metadata may be required. In these cases, the metadata creation is performed in a standardized manner in order to ensure the conformity and the metadata quality.

The *Crawler Coordinator* uses standardized *Wrapper* components to extract metadata from a given data source. Given the heterogeneity of the data portals, it becomes necessary to have a specific wrapper for each portal. Therefore, subscribed sources need to provide a wrapper that implements communication and transformation functions from the original model to the DataCollector supported model. As there are prominent platforms for publishing open data (*e.g.* CKAN), a custom wrapper implementation may be reused among different data sources based on the same platform.

In order to maintain the freshness of the DataCollector catalog, the registered data sources should be continually crawled to check for updates, including: inclusion of new datasets, removal of existing datasets and metadata updates. In this light, the *Crawler Coordinator* is the module responsible for determining the schedule in which data sources should be crawled to get fresh metadata. Furthermore, to deal with the dynamic behaviour of open data portals, the Crawler Coordinator also supports monitoring mechanisms to detect the unavailability of data sources. The Crawler Coordinator may suspend and resume specific crawling tasks when necessary.

4.2 Metadata Layer

The *Metadata Layer* is composed by four major modules: *Manager*, *Metadata Processor*, *Feedback Processor* and *Analyser*.

The *Manager* module is responsible for the registration of data sources managed by the system. During the registration process, the Manager captures basic information from data sources, such as their Web address. The registration process still involves the specification of which type of wrapper should be used for the metadata extraction. In its turns, the *Feedback Processor* module is responsible for processing data consumers feedback annotations. It provides tools to process the annotations and to store them in the *MetadataRepository*. The Feedback Processor also provides features to storage feedback data and other information inferred from the feedback.

The *Metadata Processor* module is responsible for processing the metadata extracted from the data sources. The processing tasks include metadata cleaning, analysing and loading. The concept of data cleaning was broadly used in data warehouse and data pro-

[8]http://www.drupal.org
[9]http://www.wordpress.org

cessing systems, which aims to clean up incomplete, erroneous and duplicated data (Simon, 1997). In our context, the cleaning phase is responsible for fixing incorrect or missing metadata about datasets. For instance, the most frequent cleaning tasks are related to correct information about data format and the byte size of the dataset. Very often the extracted values of these metadata don't correspond to the expected value and, therefore, they need to be corrected. In this case, cleaning tasks are performed to ensure metadata values are correct with respect to the actual descriptions of the datasets, therefore promoting more accuracy.

The analysis phase is responsible for inferring from extracted metadata if a new dataset is available or if an existing dataset was updated. In both cases, requests to insert or update the dataset in the *Metadata Repository* are created. In order to identify new versions of a given dataset, a timestamp is given to the dataset record. Finally, in the loading phase, all produced registry request are processed, storing the up-to-date processed metadata in the *MetadataRepository*.

The *Analyser* module evaluates the registered data sources and datasets according different criteria. The criteria are evaluated using metrics that can be objective (*i.e.* can be calculated automatically) and subjective (*i.e.* when human intervention is required to evaluate the criteria). The evaluation criteria are performed at dataset instance level and includes the following criteria:

- Dataset size: refers to the total amount of data stored (*e.g.* 10MB, 1GB);
- Data format: concerns the data formats used to publish the data (*e.g.* JSON, XML);
- Data domain: describes the subject and features of interests (*e.g.* Education, Health, Finance, Mobility);
- Creation and update history: concerns the history of dataset changes over time;
- License information: refers to the type of data license adopted (*e.g.* Creative Commons, Open Data Commons Open Database License);
- Contactability information: concerns the information available to a data consumer as the data publisher's contact point (*e.g.* an email address or HTTP URL);

The aforementioned evaluation may be performed once, if the datasets are static, or repeated over time, if the datasets are dynamic and their content changes.

4.3 Data Access Layer

Open data portals should provide easy access to open datasets for both humans and machines to take advantage of the benefits of sharing data using the Web infrastructure (Data, 2010). Moreover, the existence and location of datasets should be publicly known and discoverable. In order to support the data access requirements, the DataCollector provides a Web Interface and Remote Interfaces, which allows to users the access of its main functionalities. Both interfaces allow to perform datasets searching and datasets analysis.

The Web Interface provides visual abstractions over the DataCollector through Web dynamic pages that enables users to browse and discovery datasets. Users can use different dataset search mechanisms, such as keyword search mechanism as well as browsing by tags or related datasets. Moreover, It is also possible to select and visualize datasets with specific data formats or license, for example. The Web Interface aims to allow data visualization by different audiences with different needs and expectations. The Web Interface makes use of graphics components like maps and charts to enhance the experience for users.

The *Remote Interface* provides an HTTP based API to enable metadata retrieval as well as to perform more advanced operations such as searching and filtering. Almost all DataCollector features are accessible via this API, which provides alternative access to a variety of clients, including other Java clients, Web crawlers and Web services. The default data exchange format is JSON. In its current version, lacks an authentication and access control system. Finally, the Remote Interface provides different remote front ends using several Web technologies such as Rest, XML-RPC and Web Services.

5 DATACOLLECTOR IMPLEMENTATION

Figure 4 illustrates the main technologies used for implementing the modules that compose the logical architecture of the DataCollector. The platform was implemented using Java[10] and PHP programming language and can be deployed on several application servers to allow an easy management of distributed components.

In its turns, the *Metadata Repository* was implemented using MySQL[11], which is an open source re-

[10] http://www.java.com
[11] http://www.mysql.com

lational database management system. The data access to the *Metadata Repository* is indirectly done through the Data Access Object (DAO) design pattern. DAOs are design patterns that provide access to data that is usually read or written from one or more databases(Johnson, 2004). The goal of this design pattern is to enhance software maintainability, providing an interface independent of the underlying database technology. Therefore, a DAO pattern allows to replace the *Metadata Repository* solution without to recode all the *Metadata Layer*.

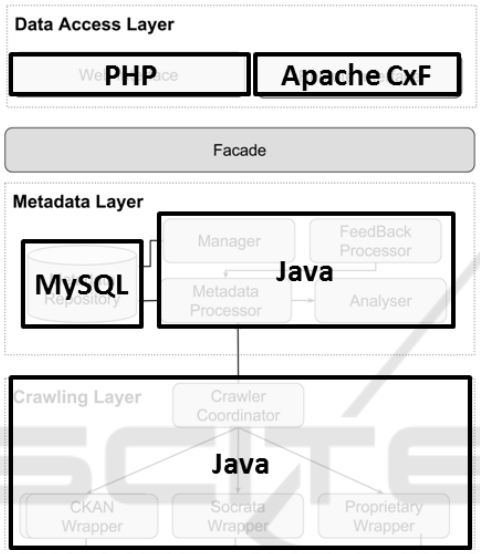

Figure 4: Technologies used to implement the DataCollector modules.

In general, the crawling process spends a lot of time waiting for responses to metadata retrieval requests. To reduce this inefficiency, the *Crawler Coordinator* uses several *CrawlerWorkers* to crawl the registered data sources. In fact, *CrawlerWorkers* are the ones that do the actual crawling work. The *Crawler Coordinator* begins by retrieving the list of subscribed data sources and then it distributes crawling tasks among the different *CrawlerWorkers* instances. Data sources are crawled concurrently, which allows the *CrawlerWorkers* to achieve faster crawl speeds by not being limited by any particular data source constraint, such as a very slow site.

Moreover, the *CrawlerWorkers* employ threads to fetch several datasets in parallel, avoiding to wait for a result to arrive before sending new requests. Making multiple requests at the same time is a well known approach to improve crawling performance (Boldi et al., 2004). When a *CrawlerWorker* visits a data source, it first retrieves the list of available datasets and then it creates a queue with the datasets. Each dataset in the queue is seeing as a metadata extraction task. After have discovered all available datasets, the threads are used to fetch dataset metadata by removing a extraction task from the front of the queue. When the queue is empty, the *CrawlerWorker* has finished its work and updates the *CrawlerCooridnator* with the processed metadata.

Analyser module uses series of *Counters* to gather metrics/statistics which can later be aggregated to produce statistics reports about different facets, such as number of datasets per given data format or total amount of data that covers education domain. Counters are incremented when certain internal condition is valid. In order to evaluate this condition, counters instances receives as input a dataset metadata. The behavior of Counter objects is defined by the following methods:

- *evaluateAndCount*: Increment a counter when a certain condition was evaluated to true;
- *evaluate*: Defines a function to evaluate if the counter need to be incremented;
- *count*: Defines how the counter must be incremented.

There are some predefined Counters and Custom counters can also be defined. The former ones are responsible to gather the analysis statistics presented in Subsection 4.2. When all datasets are analysed, these counters are consolidated to produce a holistic view for the datasets and data sources.

Analyser executes analysis process as presented in the pseudocode 5. At first, it initialize all counter instances and then retrieves a list of all registered data sources. For each data source is also retrieved a list of its datasets. Finally, all of the counters evaluate each data set according their own conditions.

```
AnalysisAlgorithm :
  couters = initializeCounters();
  dataSourceList = retrieveDataSourceList();
  FOR EACH DataSource source IN dataSourceList DO
      datasetList = retrieveDatasetList(source);
      FOR EACH Dataset dataset IN datasetList DO
         FOR EACH Counter counter IN counters DO
            conter.evaluateAndCount(dataset);
         END
      END
  END
  RETURN counters;
```

Figure 5: Analisys Pseudocode.

Users can access the main functionalities offered by the *DataCollector* through the Web interface provided by specific Web Pages, which were implemented with PHP [12] open source technologies. In order to support Remote Interfaces, we have adopted the Apache CxF[13] framework, a Web service stack from

[12]http://www.php.net
[13]http://cxf.apache.org/

the Apache Software Foundation. Apache CxF allows a clean separation from remote front-ends to application implementation. Furthermore, both Web and Remote interfaces interact with the services provided by the *Metadata Layer* through a design pattern, called Facade, that provides a unified interface to a set of interfaces in a subsystem. The Facade module encapsulates the complexity of underlying modules within a single interface, therefore promoting loose coupling to the system.

Web Interface and Remote Interface have mechanisms to facilitate datasets discovery through faceted search using multiple criteria, including the name of the data source, data domain, format and license, as can be seen in Figure 6. When the user chooses a dataset of interest, is possible to view the metadata available on the DataCollector about that dataset. In addition, when attempting to download the dataset, it is redirected automatically to the origin website. The Remote Interface use JSON as data exchange format and it employs a pagination-based iterator to deal with the large amount of data. Each request has four possible outputs: *help*, to describe the access possibilities; *next*, indicating the next page with 15 new records; *success*, indicating if the query was successful or not; and *result*, containing one *array* with all datasets found and are described in the our metadata model (vf. Section 3).

6 EVALUATION

In order to validate the the DataCollector and demonstrate its potential to integrate heterogeneous open data portals in a real-world scenario, we developed a Web portal, as a proof-of-concept, that concentrates datasets collected from 14 Brazilian OGD portals.

Figure 6: Graphic Interface.

We focus on Brazilian open data portals promoted by federal, state and municipal governments. The selection of open data portals was performed by an exploratory research on official portals of Brazilian states and its respective bigger cities. As a result, were collected data from the following OGD portals:

- Brazilian Federal Government[14];
- States: Alagoas[15], Federal District[16], Minas Gerais[17], Pernambuco[18], Rio Grande do Sul[19], and São Paulo[20];
- Capital: Curitiba[21], Fortaleza[22], Porto Alegre[23], Recife[24], Rio de Janeiro[25], and São Paulo[26].

In this scenario, the DataCollector enabled a unified view of several Brazilian open data portals and allowed the creation of a single point to access the datasets provided by them. Data consumers profit from this unified view because it becomes easier to search and select datasets distributed in multiple data portals. In its turns, data producers are able to monitor how datasets are being consumed and what problems are being notified about their datasets. In our initial evaluation, the application performance was not considered, but, in general, the response time was satisfactory. The initial crawling process spent few hours in a domestic computer. The entire analysis process take less than 1 min, and for most of datasets, it takes less than 3 seconds to show the metadata and charts. However, customized dataset search may take longer. Improving the performance through the use of materialized views and is one of our future works.

As most of the Brazilian data portals employs CKAN as platform to publish open data, a wrapper to extract metadata from CKAN was developed. This wrapper performs the following tasks: (i) obtains dataset metadata through the REST protocol; (ii) transforms JSON original data to the DataCollector metadata model. Any portal CKAN compliant can be easily integrated to the DataCollector by using the same wrapper.

After the data sources registration and the metadata extraction and transformation, it was possible to

[14] http://dados.gov.br/
[15] http://dados.al.gov.br/
[16] http://www.dadosabertos.df.gov.br/
[17] http://www.transparencia.dadosabertos.mg.gov.br
[18] http://www.dadosabertos.pe.gov.br/
[19] http://dados.rs.gov.br/
[20] http://www.governoaberto.sp.gov.br/view/consulta.php
[21] http://www.curitiba.pr.gov.br/conteudo/dados-abertos-consulta/1498
[22] http://dados.fortaleza.ce.gov.br/catalogo/
[23] http://datapoa.com.br/
[24] http://dados.recife.pe.gov.br/
[25] http://data.rio.rj.gov.br/
[26] http://www.prefeitura.sp.gov.br/cidade/secretarias/ desenvolvimento_urbano/dados_abertos/

perform dataset faceted search and several analysis about the datasets available on the brazilian OGD portals. Table 1 summarizes the analysis results.

Referring to the number of datasets, we found out that the quantity is very diverse, ranging from 33 (Curitiba) to 16,623 (Rio de Janeiro City) datasets. While only 3 Portals provide more than 1 thousand datasets, half of the portals is very small and contains only a subset of government data. For a better understanding, Figure 7 presents the proportion of number of datasets per portals.

Table 1: Dataset General Analysis.

Data Source Name	Dataset Count	Total Dataset Size	Dataset Proportion With Multiple Distributions
Alagoas	1,167	12.8 MB	3.34%
Curitiba	33	239.5 MB	0.00%
Federal District	47	1.2 GB	0.00%
Fortaleza	642	234.9 MB	0.62%
Minas Gerais	40	230.5 MB	20.00%
Pernambuco	66	1.3 GB	100.00%
Porto Alegre	149	1.1 GB	0.67%
Recife	384	1.1 GB	14.84%
Rio de Janeiro (Cap.)	16,623	1.6 GB	0.07%
Rio Grande do Sul	162	1.4 GB	13.58%
São Paulo (Cap.)	37	2.0 GB	0.00%
São Paulo(Sta.)	119	725.9 MB	2.52%
Federal Government	4,051	176.9 GB	35.74%

We also analyzed the size of the datasets, which ranges from some bytes to gigabytes of data. As expected, the biggest portal is the Federal Government portal containing 176.9 GB of data. In contrast, Alagoas provides only 12.8 MB of data, even though it has more than 1 thousand datasets. Moreover, the total size of some portals is heavily affected by small portion of files. The majority of datasets provided by São Paulo City contains less than 10 MB of data, whereas only one dataset has almost 1.5 GB. For a better understanding, Figure 8 presents the distribution of datasets size. As we may observe, most of dthem are small - more than 80% of datasets have less than 10 MB. Only 48 datasets (0.18%) have more than 100 MB.

Concerning data formats, we identified several data formats are being used, ranging from document and multimedia files, such as PDF and PNG, to structured types, such as XML and CSV. At least 47 different data formats were found during the analysis. Table 2 presents an overall proportion of the most representative data formats considering the number of occurrences and dataset size. Despite that, there is no standard data format and the large majority of datasets are

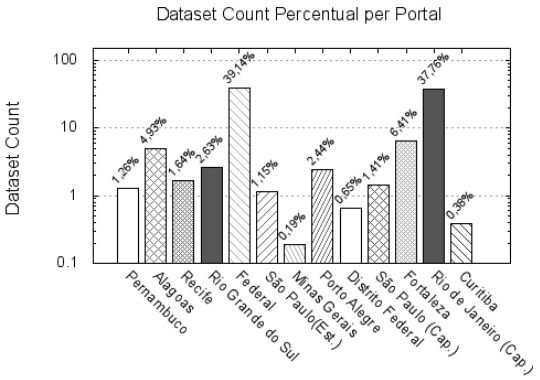

Figure 7: Dataset Count Proportion per Portal.

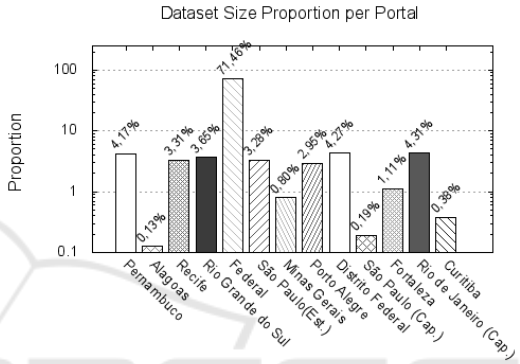

Figure 8: Dataset Size Proportion per Portal.

distributed with structured formats (84.25%). Other datasets are published as documents (14.62%), and others formats (1.13%). Relatively fewer datasets are provided in more than one format. Pernambuco and Federal Government have a higher proportion of multiple dataset distributions (100% and 33.74%). Other portals like Minas Gerais, Rio Grande do Sul, and Recife provide, respectively, 20%, 14.84% and 13.58% of all your data in more than one format. In contrast, the remainder portals provide the majority of its datasets in only one data format.

Considering the dataset domain, several topics and domains were identified, such as Education, Health, Finance and Mobility. To better understand

Table 2: Data Format Proportions.

Formats	Count Proportion	Size Proportion
csv	67.15%	21.84%
pdf	8.18%	0.11%
html	4.79%	0.01%
json	3.82%	0.07%
xml	3.52%	73.24%
shp	2.81%	1.71%
geojson	2.64%	2.14%
kml	2.58%	0.02%
txt	1.49%	0.10%
Other Formats	2.08%	0.76%

the dataset domain landscape, Figure 9 presents a word cloud containing the most frequent keywords. The most required topics are related to Geosciences, Statistics analysis, and Finance. However, this distribution is not the same for all portals. While in Porto Alegre and Recife Mobility is the dominant domain; in Pernambuco and Rio Grande do Sul, the most frequent domains are Employment and Education, respectively. Moreover, there is a lack of standardized terms to classify dataset domains. As a consequence, several different terms are used to represent the same concept, such as transportation and transit (Federal Government), mobility (Recife) and transit (Rio Grande do Sul).

Figure 9: Dataset Domain Word Cloud.

The lifetime of the data portals was also analysed. As a result of this, we identified two main phases during the lifetime of brazilian OGD portals. First, the creation phase, when the data portal is created and an initial load with several datasets is performed. Second, the update phase, when new datasets are created or existing datasets are updated. In our analysis, we observe in the most cases when new information is added to a portal then a new dataset is created.

7 CONCLUSION AND FUTURE WORKS

The DataCollector solution, presented in this article, aims to allow the cataloging and discovery of datasets available in open data portals, providing a single point of access for open datasets published in heterogeneous portals. The solution presented enables automatic extraction of detailed metadata about the datasets and stores them in accordance with the standard vocabulary for data modeling catalogs. It also provides a set of procedures and mechanisms that allows dataset search using either Web graphic interfaces, as well as remote access mechanisms, such as Web Services, enabling the building of applications for recovery and automatic data processing.

A first evaluation was conducted by cataloging 14 Brazilian open data portals, covering a total of 29,540 datasets. Preliminary results indicate the DataCollector offers a friendly interface and a robust solution for cataloging and dataset access. Based on the services offered by the the DataCollector, we could create a single point access to distributed brazilian open government datasets, helping the identification of relationships between datasets as well as the dataset search. We could perform several analysis on these datasets and we found out several interesting information about the brazilian OGD scenario.

The DataCollector is still under development and new features may be incorporated. As future works, we intend to improve the analysis support offered by the DataCollector with the implementation of a support decision module that allows the creation of OLAP (*Online Analytical Processing*) reports and offers *dashboard* panels for data visualization and data analysis. We also plan to improve the analysis with the inclusion of analysis based on the datasets schemas as well as on the data itself in order to evaluate aspects like data sparseness.

ACKNOWLEDGMENT

This work was partially supported by the Brazilian funding agency FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco) under grant APQ-1088-1.03/15. The authors are indebted to the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) and National Committee of Technological and Scientific Development (CNPq), for the scholarship of Marcelo Iury and Lairson Emanoel.

REFERENCES

Amatriain, X., Pujol, J. M., and Oliver, N. (2009). I like it... i like it not: Evaluating user ratings noise in recommender systems. In *User modeling, adaptation, and personalization*, pages 247–258. Springer.

Barbosa, L., Pham, K., Silva, C., Vieira, M. R., and Freire, J. (2014). Structured open urban data: understanding the landscape. *Big data*, 2(3):144–154.

Boldi, P., Codenotti, B., Santini, M., and Vigna, S. (2004). Ubicrawler: A scalable fully distributed web crawler. *Software: Practice and Experience*, 34(8):711–726.

Chun, S. A., Shulman, S., Sandoval, R., and Hovy, E. (2010). Government 2.0: Making connections be-

tween citizens, data and government. *Information Polity*, 15(1):1.

Data, O. G. (2010). Eight principles of open government data.

Foundation, O. K. (2015a). Datahub.io.

Foundation, O. K. (2015b). Publicdata.eu.

Hotho, A., Jäschke, R., Schmitz, C., and Stumme, G. (2006). *Information retrieval in folksonomies: Search and ranking*. Springer.

Johnson, R. (2004). *Expert one-on-one J2EE design and development*. John Wiley & Sons.

Maali, F., Erickson, J., and Archer, P. (2014). Data catalog vocabulary (DCAT). W3C recommendation, The World Wide Web Consortium.

Miranda, C. (2015). Abra com simplicidade – desmistificando os dados abertos.

Project, C. (2015). Opendatamonitor.eu.

Ribeiro, D. C., Vo, H. T., Freire, J., and Silva, C. T. (2015). An urban data profiler. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 1389–1394. International World Wide Web Conferences Steering Committee.

Sanderson, R., Ciccarese, P., and Young, B. (2015). Web annotation data model. W3C recommendation, The World Wide Web Consortium.

Simon, A. (1997). *Data Warehouse, Data Mining and OLAP*. John Wiley & Sons. USA.

van den Broek, T., Rijken, M., and van Oort, S. (2012). Towards open development data. *Open for Change. Retrieved February*, 25:2013.