Metadata of the chapter that will be visualized in SpringerLink

Book Title	Enterprise Information Systems	
Series Title		
Chapter Title	CRAWLER-LD: A Multilevel Metadata Focused Crawler Framework for Linked Data	
Copyright Year	2015	
Copyright HolderName	Springer International Publishing Switzerland	
Corresponding Author	Family Name	**Vale A. Gomes**
	Particle	**do**
	Given Name	**Raphael**
	Prefix	
	Suffix	
	Division	Departamento de Informática
	Organization	PUC-Rio
	Address	Rio de Janeiro, Brazil
	Email	rgomes@inf.puc-rio.br
Author	Family Name	**Casanova**
	Particle	
	Given Name	**Marco A.**
	Prefix	
	Suffix	
	Division	Departamento de Informática
	Organization	PUC-Rio
	Address	Rio de Janeiro, Brazil
	Email	casanova@inf.puc-rio.br
Author	Family Name	**Lopes**
	Particle	
	Given Name	**Giseli Rabello**
	Prefix	
	Suffix	
	Division	Departamento de Informática
	Organization	PUC-Rio
	Address	Rio de Janeiro, Brazil
	Email	grlopes@inf.puc-rio.br
Author	Family Name	**Paes Leme**
	Particle	
	Given Name	**Luiz André P.**
	Prefix	
	Suffix	
	Division	Instituto de Computação
	Organization	UFF
	Address	Niterói, Brazil

Email lapaesleme@ic.uff.br

Abstract

The Linked Data best practices recommend to publish a new tripleset using well-known ontologies and to interlink the new tripleset with other triplesets. However, both are difficult tasks. This paper describes CRAWLER-LD, a metadata crawler that helps selecting ontologies and triplesets to be used, respectively, in the publication and the interlinking processes. The publisher of the new tripleset first selects a set *T* of terms that describe the application domain of interest. Then, he submits *T* to CRAWLER-LD, which searches for triplesets whose vocabularies include terms direct or transitively related to those in *T*. CRAWLER-LD returns a list of ontologies to be used for publishing the new tripleset, as well as a list of triplesets that the new tripleset can be interlinked with. CRAWLER-LD focuses on specific metadata properties, including subclass of, and returns only metadata, hence the classification “metadata focused crawler”.

Keywords (separated by '-') Focused crawler - Tripleset recommendation - Linked data

CRAWLER-LD: A Multilevel Metadata Focused Crawler Framework for Linked Data

Raphael do Vale A. Gomes[1(✉)], Marco A. Casanova[1], Giseli Rabello Lopes[1], and Luiz André P. Paes Leme[2]

[1] Departamento de Informática, PUC-Rio, Rio de Janeiro, Brazil
{rgomes, casanova, grlopes}@inf.puc-rio.br
[2] Instituto de Computação, UFF, Niterói, Brazil
lapaesleme@ic.uff.br

Abstract. The Linked Data best practices recommend to publish a new tripleset using well-known ontologies and to interlink the new tripleset with other triplesets. However, both are difficult tasks. This paper describes CRAWLER-LD, a metadata crawler that helps selecting ontologies and triplesets to be used, respectively, in the publication and the interlinking processes. The publisher of the new tripleset first selects a set *T* of terms that describe the application domain of interest. Then, he submits *T* to CRAWLER-LD, which searches for triplesets whose vocabularies include terms direct or transitively related to those in *T*. CRAWLER-LD returns a list of ontologies to be used for publishing the new tripleset, as well as a list of triplesets that the new tripleset can be interlinked with. CRAWLER-LD focuses on specific metadata properties, including subclass of, and returns only metadata, hence the classification "metadata focused crawler".

Keywords: Focused crawler · Tripleset recommendation · Linked data

1 Introduction

The Linked Data best practices [1] recommend publishers of triplesets to use well-known ontologies in the triplification process and to interlink the new tripleset with other triplesets. However, despite the fact that extensive lists of open ontologies and triplesets are available, such as DataHub,[1] most publishers typically do not adopt ontologies already in use and link their triplesets only with popular ones, such as DBpedia[2] and Geonames.[3] Indeed, according to [2–4], linkage to popular triplesets is favored for two main reasons: the difficulty of finding related open triplesets; and the strenuous task of discovering instance mappings between different triplesets.

This paper describes CRAWLER-LD, a crawler that addresses the problem of recommending vocabulary terms and triplesets to assist publishers in the publication and the interlinking processes. Unlike typical Linked Data crawlers, the proposed crawler then focuses on metadata with specific purposes, illustrated in what follows.

[1] http://datahub.io.
[2] http://dbpedia.org.
[3] http://www.geonames.org.

J. Cordeiro et al. (Eds.): ICEIS 2014, LNBIP 227, pp. 1–18, 2015.
DOI: 10.1007/978-3-319-22348-3_17

In a typical scenario, the publisher of a tripleset first selects a set T of terms that describe an application domain. Alternatively, he could use a database summarization technique [5] to automatically extract T from a set of triplesets. Then, the publisher submits T to CRAWLER-LD, which will search for triplesets whose vocabularies include terms direct or transitively related to those in T. The crawler returns a list of vocabulary terms, as well as provenance data indicating how the output was generated. For example, if the publisher selects the term "`Music`" from WordNet, the crawler might return "`Hit music`" from BBC Music. Lastly, the publisher inspects the list of triplesets and terms returned, with respect to his tripleset, to select the most relevant vocabularies for triplification and the best triplesets to use in the interlinking process, possibly with the help of recommender tools. We stress that the crawler was designed to help recommender tools for Linked Data, not to replace them.

This paper is organized as follows. Section 2 presents related work. Section 3 summarizes background information about the technology used. Section 4 briefly explains how the crawler works with the help of an example. Section 5 details the crawling process. Section 6 describes experiments that assess the usefulness of the crawler. Finally, Sect. 7 presents the conclusions and future work.

2 Related Work

We first compare CRAWLER-LD with Linked Data crawlers. Fionda et al. [6] present a language, called NAUTILOD, which allows browsing through nodes of a Linked Data graph. They introduced a tool, called *swget* (semantic web get), which evaluates expressions of the language. An example would be: "find me information about Rome, starting with its definition in DBpedia and looking in DBpedia, Freebase and the New York Times databases".

```
swget < dbp:Rome > (< owl:sameAs >)* -saveGraph
-domains {dbpedia.org,rdf.freebase.com,data.nytimes.com}
```

LDSpider [7] is another example of a Linked Data crawler. Similarly to the crawler proposed in this paper, LDSpider starts with a set of URIs as a guide to parse Linked Data. Ding et al. [8] present the tool created by Swoogle to discover new triplesets. The authors describe a way of ranking Web objects in three granularities: Web documents (Web pages with embedded RDF data), terms and RDF Graphs (triplesets). Each of these objects has a specific ranking strategy.

The Linked Data crawlers just described have some degree of relationship with the proposed crawler, though none has exactly the same goals. As explained in the introduction, the proposed crawler focuses on finding metadata that are useful to design new triplesets. Furthermore, rather than just dereferencing URIs, it also adopts *crawling queries* to improve recall, as explained in Sect. 5.1.

We now comment on how the proposed crawler relates to recommender tools for Linked Data. Some generic recommender tools use keywords as input. Nikolov et al. [2, 3] use keywords to search for relevant resources, using the label property of the resources. Indeed, a label is a property used to provide a human-readable version of the

name of the resource.[4] A label value may be inaccurate, in another language or simply be a synonymous of the desired word. There is no compromise with the schema and its relationships. Therefore, the risk of finding an irrelevant resource is high. Martínez-Romero et al. [4] propose an approach for the automatic recommendation of ontologies based on three points: (1) how well the ontology matches the set of keywords; (2) the semantic density of the ontology found; and (3) the popularity of the tripleset on the Web 2.0. They also match a set of keywords to resource label values, in a complex process.

CRAWLER-LD may be used as a component of a recommender tool, such as those just described, to locate: (1) appropriate ontologies during the triplification of a database; (2) triplesets to interlink with a given tripleset. We stress that the crawler was not designed to be a full recommender tool, but rather to be a component of one such system.

3 Background

The Linked Data principles advocate the use of RDF [9], RDF Schema [10] and other technologies to standardize resource description.

RDF describes resources and their relationships through *triples* of the form (*s*, *p*, *o*), where: *s* is the *subject* of the triple, which is an RDF URI reference or a blank node; *p* is the *predicate* or *property* of the triple, which is an RDF URI reference and specifies how *s* and *o* are related; and *o* is the *object*, which is an RDF URI reference, a literal or a blank node. A triple (*s*, *p*, *o*) may also be denoted as "<*s* > <*p* > <*o*>".

A *tripleset* is just a set of triples. In this paper will use *dataset* and *tripleset* interchangeably.

RDF Schema is a semantic extension of RDF to cover the description of classes and properties of resources. OWL [11] in turn extends RDF Schema to allow richer descriptions of schemas and ontologies, including cardinality and other features.

RDF Schema and OWL define the following predicates that we will use in the rest of the paper:

- `rdfs:subClassOf` indicates that the subject of the triple defines a subclass of the class defined by the object of the triple
- `owl:sameAs` indicates that the subject denotes the same concept as the object
- `owl:equivalentClass` indicates that both the subject and the object are classes and denote the same concept
- `rdf:type` indicates that the subject is an instance of the object

For example, the triple

```
<dbpedia:Sweden > < rdf:type > < dbpedia:Country > .
```

indicates that the resource Sweden is an instance of the class Country.

Triplesets are typically available on the Web as SPARQL endpoints [12] or as file dumps (large files containing all the data from a tripleset, or small files containing only the relevant data for a defined term). A third option is through URL dereferencing,

[4] http://www.w3.org/TR/rdf-schema/#ch_label

which means that the resource contains description data about itself so it is possible to discover more data simply by reading the resource content.

More than just a query language similar to SQL, SPARQL is a protocol: it defines the query interface (HTTP), how requests should be made (POST or GET) and how the data should be returned (via a standard XML). Thus, an agent can perform queries on a dataset and acquire knowledge to create new queries and so on.

Finally, VoID [13] is an ontology used to define metadata about triplesets. A VoID document is a good source of information about a tripleset, such as the classes and properties it uses, the size of the tripleset, etc.

Let d be a tripleset and V be a set of VoID metadata descriptions. The classes and properties used in d can be extracted from tripleset partitions defined by the properties `void:classPartition` and `void:propertyPartition` that occur in V. *Class partitions* describe sets of triples related to subjects of a particular class. *Property partitions* describe sets of triples that use a particular predicate. These partitions are described by the properties `void:class` and `void:property` respectively. The set of vocabulary terms used in d can be generated by the union of all values of the properties `void:class` and `void:property`. In some cases, the VoID description of a tripleset does not define partitions, but it specifies a list of namespaces of the vocabularies used by the tripleset with the `void:vocabulary` predicate. One can enrich the set of vocabulary terms used in d with such a list.

4 Use Case

Consider a user who wants to publish as Linked Data a relational database d storing music data (artists, records, songs, etc.). The crawler proposed in this paper will help the user to publish d as follows.

First, the user has to define an initial set T of terms to describe the application domain of d. Suppose that he selects just one term `dbpedia:Music`, taken from DBpedia.

The user will then invoke the crawler, passing T as input. The crawler will query the DataHub catalogue of Linked Data triplesets to crawl triplesets searching for new terms that are direct or transitively related to `dbpedia:Music`. The crawler focuses on finding new terms that are defined as subclasses of the class `dbpedia:Music`, or that are related to `dbpedia:Music` by `owl:sameAs` or `owl:equivalentClass` properties. The crawler will also count the number of instances of the classes found.

The crawler will return: (1) the list of terms found, indicating their provenance - how the terms are direct or transitively related to `dbpedia:Music` and in which triplesets they were found; and (2) for each class found, an estimation of the number of instances in each tripleset visited.

The user may take advantage of the results the crawler returned in two ways. He may manually analyze the data and decide: (1) which of the probed ontologies found he

will adopt to triplify the relational database; and (2) to which triplesets the crawler located he will link the tripleset he is constructing. Alternatively, he may submit the results of the crawler to separate tools that will automatically recommend ontologies to be adopted in the triplification process, as well as triplesets to be used in the linkage process [14, 15].

For example, suppose that the crawler finds two subclasses, `opencyc:Love_Song` and `opencyc:Hit_Song`, of `wordnet:synset-music-noun-1` in the ontology `opencyc:Music`. Suppose also that the crawler finds large numbers of instances of these subclasses in two triplesets, `musicBrainz` and `bbcMusic`. The user might then decide that `opencyc:Music` is a good choice to adopt in the triplification process and that `musicBrainz` and `bbcMusic` are good choices for the linkage process.

5 A Metadata Focused Crawler

5.1 Processors

The crawler works with catalogues that use the CKAN framework (such DataHub) to identify SPARQL endpoints and RDF dumps, and the user can also manually add new datasets. It receives as input a set of terms T, called the *initial crawling terms*. Such terms are typically selected from generic ontologies, such as WordNet,[5] DBpedia and Schema.org,[6] albeit this is not a requirement for the crawling process. Given T, the crawler uses a list C of *processors*, in successive *stages* (see Sect. 5.2), to extract new terms from the triplesets listed in the catalogues.

The tool is engineered as a framework, whose pseudo-code is listed in Appendix 1. It passes each *crawling term* t to each processor in C. The processor annotates the provenance of its crawled data and returns a list of terms to be crawled in the next stage, after filtering, based on parameters specified by the user (see Sect. 5.2). Currently, the crawler includes three processors, described in the rest of this section.

Dereference Processor. The first processor is responsible for extracting information of the resource itself. It tries to find new resources using the properties `owl:sameAs`, `owl:equivalentClass` and `rdfs:subclassOf` (see Sect. 3). For each such property, the processor applies a SPARQL query to extract new information. The following template illustrates how each query works, where `p` is one of the above properties and `t` is the crawling term itself; the values assigned to the variable `?item` are resources that to be crawled in a next stage.

```
SELECT distinct ?item
WHERE {< t > p ?item}
```

[5] http://wordnet.princeton.edu/.

[6] http://schema.org.

Given that `owl:sameAs` and `owl:equivalentClass` are reflexive, the processor also applies SPARQL queries generated by a new code template, with the subject and object inverted:

```
SELECT distinct ?item
WHERE {?item p < t>}
```

Property Processor. This processor is responsible for crawling other datasets. It uses a special SPARQL query, which runs over each dataset discovered in DataHub and manually as described Sect. 5.1. The motivation is to extract information that is not directly related to the resources already processed. Given the crawling term *R* that will be processed by CRAWLER-LD and a dataset *D* that uses *R* to describe a fraction of its data. While a conventional crawling algorithm is not able to find *D* since *R* does not have any reference to *D*, CRAWLER-LD, on the other hand, traverses all datasets available and is able to find the relationship between *D* and *R*.

The processor uses the following SPARQL template, where `t` is the resourced being crawled:

```
SELECT distinct ?property ?item
WHERE {
{?item owl:sameAs < t>.}
UNION { < t > owl:sameAs ?item.}
UNION {?item owl:equivalentClass < t>.}
UNION { < t > owl:equivalentClass ?item.}
UNION {?item rdfs:subClassOf < t>.}
?item ?property < t>. }
```

Note that, for each term `t` to be crawled, the template inverts the role of `t` (for the details, see lines 7 and 9 of the code in Appendix (1), when the predicate is `owl:sameAs` and `owl:equivalentClass`, since these predicates are reflexive. However, the crawler does not invert the role of `t`, when the predicate is `rdfs:subClassOf`, since this predicate is not reflexive.

For example, in the specific case of the crawling property `rdfs:subClassOf`, suppose that *C* and *C'* are classes defined in triplesets *S* and *S'*, respectively, and assume that *C'* is declared as a subclass of *C* through a triple of the form

(*C'*, `rdfs:subClassOf`, *C*)

Triples such as this are more likely to be included in the tripleset where the more specific class *C'* is defined than in the tripleset where the more generic class *C* is defined. Hence, after finding a class *C*, the crawler has to search for subclasses of *C* in all triplesets it has access using the template above.

Another case occurs when the relationship between *C* and *C'* is defined in a third ontology *S"*. Similarly to the previous example, we need a subclass query over *S"* to discover that *C'* is a subclass of *C*. *S"* is obtained by dereferencing the URI of *C'*. In most cases the returned tripleset is the complete ontology where *C'* is defined, while in some other cases only a fragment of the ontology where *C'* is defined is returned.

Instance Counter Processor. The last processor extracts information about the quantity of instances available in each dataset for each crawling term. It runs queries over all datasets, using the same principle as the property processor. To reduce the bandwidth, the processor uses grouping functions to query datasets:

```
SELECT distinct (count(?instance) AS ?item)
WHERE {?instance rdf:type < %s > . }
```

Unfortunately, grouping functions are only available in SPARQL version 1.1 [16] and above. Therefore, the processor also crawls the remaining datasets using the following query, which spend more bandwidth:

```
SELECT distinct ?item
WHERE {?item rdf:type < %s > . }
```

5.2 Crawling Stages

The crawler simulates a breath-first search for new terms. Stage 0 contains the initial set of terms. The set of terms of each new stage is computed from those of the previous stage, as described in Sect. 5.1.

The *crawling frontier* is the set of terms found which have not yet been processed. To avoid circular references, we used a hash map that indicates which terms have already been processed.

Since the number of terms may grow exponentially from one stage to the next, we prune the search by limiting:

- The number of stages of the breath-first search
- The maximum number of terms probed
- The maximum number of terms probed in each tripleset, for each term in the crawling frontier
- The maximum number of terms probed for each term in the crawling frontier

For each new term found, the processors create a list that indicates the provenance of the term: how the term is direct or transitively related to an initial term and in which tripleset(s) it was found. That is, the crawler identifies the sequence of relationships it traversed to reach a term, such as in the following example:

```
wordnet:synset-music-noun-1 - > owl:sameAs ->
opencyc:Music - > rdfs:subClassOf ->
opencyc:LoveSong - > instance ->
500 instances.
```

6 Tests and Results

6.1 Organization of the Experiments

We evaluated the crawler over triplesets described in DataHub. The tool was able to recover 1,042 triplesets with SPARQL endpoints. However, despite this number, it could run queries on just over 35 % of the triplesets due to errors in the query parser or simply because the servers were not available.

To create the initial crawling terms, we used three generic ontologies, WordNet, DBpedia and Schema.org, as well as ontologies specific to the application domain in question.

Table 1. Namespace abbreviation.

Abbreviation	Namespace
`akt`	http://www.aktors.org/ontology/portal#
`bbcMusic`	http://linkeddata.uriburner.com/about/id/entity/http://www.bbc.co.uk/music/
`dbpedia`	http://dbpedia.org/resource/ or http://dbpedia.org/ontology/
`dbtune`	http://dbtune.org/
`freebase`	http://freebase.com/
`freedesktop`	http://freedesktop.org/standards/xesam/1.0/core#
`lastfm`	http://linkeddata.uriburner.com/about/id/entity/http://www.last.fm/music/
`mo`	http://purl.org/ontology/mo/
`musicBrainz`	http://dbtune.org/musicbrainz/
`nerdeurocom`	http://nerd.eurecom.fr/ontology#
`opencyc`	http://sw.opencyc.org/2009/04/07/concept/en/
`schema`	http://schema.org/
`twitter`	http://linkeddata.uriburner.com/about/id/entity/http://twitter.com/
`umbel`	http://umbel.org/
`wordnet`	http://wordnet.rkbexplorer.com/id/
`yago`	http://www.yago-knowledge/resource/

WordNet is a lexical database that presents different meanings for the same word. For example, the word "music" is given two different meanings, denoted by two distinct terms: `wordnet:synset-music-noun-1` means "an artistic form of auditory communication incorporating instrumental or vocal tones in a structured and continuous manner", while `wordnet:synset-music-noun-2` means "any agreeable (pleasing and harmonious) sounds; "he fell asleep to the music of the wind chimes"".

DBpedia is the triplified version of the Wikipedia database. The triplification process is automatically accomplished and the current English version already has 2.5 million classified items.

Schema.org is the most recent ontology of all three. It focuses on HTML semantics and was created by Google, Bing and Yahoo. Therefore, Schema.org is now used by many triplesets.[7] Schema.org is also developing other ways to increase the search results by creating a mapping with other ontologies, such as DBpedia and WordNet.

In the examples that follow, we use the abbreviations shown in Table 1.

6.2 Results

The experiments involved two domains, Music and Publications, and used the following parameters:

- Number of stages: 2
- Maximum number of terms probed: 40
- Maximum number of terms probed for each term in the crawling frontier: 20
- Maximum number of terms probed in each tripleset, for each term in the crawling frontier: 10

Music Domain. We chose Music as the first domain to evaluate the crawler and elected three ontologies, DBpedia, WordNet and Music Ontology,[8] to select the initial crawling terms. The Music Ontology is a widely accepted ontology that describes music, albums, artists, shows and some specific subjects.

The initial crawling terms were:

```
mo:MusicArtist          dbpedia:Album
mo:MusicalWork          dbpedia:MusicalArtist
mo:Composition          dbpedia:Single
dbpedia:MusicalWork     wordnet:synset-music-noun-1
dbpedia:Song
```

In what follows, we will first comment on the results obtained in Stage 1, for each initial term. Then, we will proceed to discuss how the new terms obtained in Stage 1 were processed in Stage 2.

Table 2(a) shows the results of Stage 1 for `mo:MusicalArtist`. On Stage 2, for each of the terms `mo:MusicGroup` and `mo:SoloMusicArtist`, the crawler obtained different results: while `mo:MusicGroup` recovered over 1.5 million instances over three datasets, `mo:SoloMusicArtist` did not find any new result.

Table 2(b) shows the results of Stage 1 for `mo:MusicalWork`. Note that the crawler found a variety of instances from multiple databases. On Stage 2, when processing `mo:Movement`, the crawler did not find any new instance or class.

Table 2(c) shows the results of Stage 1 for the first DBpedia term, `dbpedia:MusicalWork`. The crawler found 5 subclasses from DBpedia and over a million instances in 13 datasets, with 8 being DBpedia in different languages (such as French, Japanese, Greek and others), which was only possible because it focused on metadata.

[7] http://schema.rdfs.org/mappings.html.

[8] http://musicontology.com/

Table 2. Related terms.

Related terms	
Query type	Description
(a) Related terms for `mo:MusicArtist`	
subclass	`mo:MusicGroup`, `mo:SoloMusicArtist`
instance	2,647,957 instances from over four datasets
(b) Related terms for `mo:MusicalWork`	
subclass	`mo:Movement`
instance	1,166,365 instances found in multiple databases
(c) Related terms for `dbpedia:MusicalWork`	
subclass	`dbpedia:Album`, `dbpedia:Song`, `dbpedia:Single`, `dbpedia:Opera`, `dbpedia:ArtistDiscography`
instance	939,480 instances from 13 datasets
(d) Related terms for `dbpedia:Song`	
equivalentclass	`schema:MusicRecording`
subclass	`dbpedia:EurovisionSongContestEntry`
instance	35,702 instances from 9 datasets
(e) Related terms for `dbpedia:Album`	
equivalentclass	`schema:MusicAlbum`
instance	871,348 instances from 13 datasets
(f) Related terms for `dbpedia:MusicalArtist`	
instance	424,152 instances from 19 datasets
(g) Related terms for `dbpedia:Single`	
instance	305,041 instances from 10 datasets

Crawlers that use text fields [2] can only retrieve data in the same language as that of initial terms.

The first three terms, `dbpedia:Album`, `dbpedia:Song` and `dbpedia:Single`, will be analyzed in the next paragraphs since they are also in the initial set of terms.

On Stage 2, the processing of `dbpedia:Opera` returned no results and the processing of `dbpedia:ArtistDiscography` returned 48,784 instances, but no new term.

Table 2(d) shows the results of Stage 1 for `dbpedia:Song`. The crawler was able to find a relationship with other generic dataset (Schema.org) and also found a variety of resources from DBpedia in different languages.

On Stage 2, when processing `dbpedia:EurovisionSongContestEntry`, the crawler found 7,807 instances from 7 datasets. The other resource probed on the Stage 2 was `schema:MusicRecording`, which returned 38,464 instances and no new crawling terms.

Table 2(e) shows the results of Stage 1 for `dbpedia:Album`. The processing of this term also found `schema:MusicAlbum` and a large number of instances. On

Stage 2, the tool was able to find 662,409 instances of `schema:MusicAlbum`, but no new resource.

Table 2(f) shows the results of Stage 1 for `dbpedia:MusicalArtist`. The tool was not able to find any new related resource, but it found a large number of datasets that have instances of this class.

Table 2(g) shows the results of Stage 1 for `dbpedia:Single`. The tool found more than 300 thousand instances from triplesets in many languages.

The last term probed in Stage 1 was `wordnet:synset-music-noun-1`. The crawler found a *sameAs* relationship with an analogue term from another publisher: http://www.w3.org/2006/03/wn/wn20/instances/synset-music-noun-1.

Finally, we remark that, when we selected the terms to evaluate, we expected to find relationships between DBpedia and the Music Ontology, which did not happen. In addition, we found much better results using terms from DBpedia than from the Music Ontology, which is specific to the domain in question. The definition of links between the Music Ontology and DBpedia could increase the popularity of the former. For example, if the term `mo:MusicArtist` were related to the term `dbpedia:MusicalArtist`, crawlers such as ours would be able to identify the relationship. Also, matching or recommendation tools would benefit from such relationship.

Publications Domain. For the second domain, we focused on two ontologies, Schema.org and Aktors,[9] which is commonly used by publications databases. We selected the following terms:

```
schema:TechArticle
schema:ScholarlyArticle
akt:Article-Reference
akt:Article-In-A-Composite-Publication
akt:Book, akt:Thesis-Reference akt:Periodical-Publication
akt:Lecturer-In-Academia
akt:Journal
```

The results were quite simple. Both ontologies (Schema.org and Aktors) returned a small number of instances, but with no complex structure. A quick analysis showed that almost all triplesets were obtained from popular publications databases (such as DBLP, IEEE and ACM) by the same provider (RKBExplorer), which used the Aktors ontology. In addition, the Aktors ontology is not linked to other ontologies, which lead to an almost independent cluster in the Linked Data cloud.

Processing Times. Table 3 shows the processing time for each experiment. In general, the time spent to process each term was direct related to the number of terms found (some exceptions apply due to bandwidth issues). The experiment was performed on a virtual machine hosted by Microsoft Azure[10] with 56 GB and two AMD Opteron™ 4171 processors.

[9] http://www.aktors.org

[10] http://azure.microsoft.com/

Table 3 shows that the minimum time was 4 min, when no new terms were found, but the maximum time depended on the number of new terms in the crawling frontier and how the network (and the endpoints) responded.

Finally, we observe that the processing time can be optimized, provided that: (1) the endpoints queries have lower latency; (2) the available bandwidth is stable across the entire test; (3) cache features are used.

Table 3. Performance evaluation.

Term	Proc. time (minutes)
Music domain	
mo:MusicArtist	11
mo:MusicalWork	8
mo:Composition	4
dbpedia:MusicalWork	22
dbpedia:Song	11
dbpedia:Album	8
dbpedia:MusicalArtist	4
dbpedia:Single	4
wordnet:synset-music-noun-1	11
Publications domain	
schema:TechArticle	4
schema:ScholarlyArticle	4
akt:Article-Reference	4
akt:Article-In-A-Composite-Publication	8
akt:Book	5
akt:Thesis-Reference	5
akt:Periodical-Publication	4
akt:Lecturer-In-Academia	5
akt:Journal	4

Table 4. Number of terms found using *swget*.

Term	Subclass	SameAs	Equivalentclass	Type
mo:MusicArtist	6	0	0	0
mo:MusicalWork	8	0	0	0
dbpedia:MusicalWork	21	0	0	0
dbpedia:Song	7	0	1	0
dbpedia:Album	6	0	1	0
dbpedia:MusicalArtist	10	0	0	0
dbpedia:Single	6	0	0	0

6.3 A Comparison with *Swget*

We opted for a direct comparison between CRAWLER-LD and *swget* for three reasons. First, there is no benchmark available to test Linked Data crawlers such as ours and it is nearly impossible to manually produce one such (extensive) benchmark. Second, *swget* is the most recent crawler available online. Third, it was fairly simple to setup an experiment for *swget* similar to that described in Sect. 6.2 for the Music domain.

Briefly, the experiment with *swget* was executed as follows. Based on the examples available at the *swget* Web site, we created the following template to run queries (where `t'` is the term to be probed and `q'` the current crawling property):

```
t' -p < q' > < 2-2>
```

The above query means "given a term `t'`, find all resources related to it using the predicate `q'`, expanding two levels recursively.

Then, we collected all terms *swget* found from the same initial terms of the Music domain used in Sect. 6.2, specifying which crawled property *swget* should follow. Table 4 shows the number of terms *swget* found, for each term and crawling property.

Based on the experiments with *swget* and CRAWLER-LD, we compiled a list of terms shown in Appendix 2. Then, we manually inspected the terms and marked those that pertain to the Music domain and those that *swget* and CRAWLER-LD found.

The results can be summarized by computing the precision, recall and balanced F-measure (F_1) [17] obtained by *swget* and CRAWLER-LD for the list of terms thus obtained:

- *swget*: Precision = 29.17 % Recall = 100 % F_1 = 45.16 %
- CRAWLER-LD: Precision = 100 % Recall = 78.57 % F_1 = 88.00 %

These results should be interpreted as follows. *Swget* achieved a much lower precision since it finds more generic and more specific terms at the same time, while CRAWLER-LD only searches for the more specific terms. This feature creates undesirable results for the purposes of focusing on an application domain. For example, using `rdfs:subClassOf` as predicate and `dbpedia:MusicalWork` as object, *swget* returned `dbpedia:Work`, a superclass at the first level. At the next stage, *swget* then found resources such as `dbpedia:Software` and `dbpedia:Film`, each of them subclasses of `dbpedia:Work`, but unrelated to the Music domain.

Swget achieved a larger recall value. However, *Swget* found only two related resources that CRAWLER-LD could not find. The basic reason is that the crawling resources were subclasses of domain resources, which implied that CRAWLER-LD could not reach the resources, while *swget* was able to.

Analyzing the overall quality of the crawlers using balanced F-measure, CRAWLER-LD outperformed *swget*, obtaining an F1 result almost twice as large as that of *swget*. Thus, in this experiment, CRAWLER-LD was able to find a better balance between recall and precision values than *swget*.

Swget was also unable to identify the same number of instances as CRAWLER-LD. While CRAWLER-LD searched a large number of datasets, *swget* tried to obtain data

only by reading the resource's content, which does not describes instances from any other dataset. This behavior should not be regarded as defect of *swget*, though, but a consequence of working with a general-purpose crawler, rather than a metadata focused crawler, such as CRAWLER-LD.

6.4 Lessons Learned

In this section, we highlight the main lessons learned from the results of our experiments. We first enumerate some aspects that may influence the crawling results, such as the settings of the parameters and the availability of sufficient information about the crawled triplesets.

Parameter Setting. Since, in our crawler, the set of terms of each new stage is computed from that of the previous stage, the number of terms may grow exponentially. We defined some parameters to prune the search. Hence, the user must adequately set such parameters to obtain results in reasonable time, without losing essential information.

Choosing the Initial Crawling Terms. In the Music domain experiments, we started with terms from three different triplesets, DBpedia, WordNet and Music Ontology, the first two being more generic than the last one. It seems that the resources defined in the Music Ontology are not interlinked (directly or indirectly) with the more popular triplesets. This limitation is related to the fact that some triplesets do not adequately follow the Linked Data principles, in the sense that they do not interlink their resources with resources defined in other relevant triplesets.

Ontologies Describing the Domain of Interest. Our crawler proved to return more useful when there are relationships among the metadata. In the experiments using the publications domain, our crawler returned a simplified result because all triplesets related to the initial crawling terms used the same ontology to describe their resources. In general, the larger the number of triplesets in the domain, the more useful the results of our crawler will be.

Reducing the Number of Request. Our crawler demands a high number of requests for each dataset, and creating ways to reduce this number would improve the performance. Our approach, primarily implemented on *property processor,* combines all queries inside one using the *UNION* clause and processing the result set locally.

We now highlight some improvements obtained by our metadata focused crawler, when compared to traditional crawlers.

Discovering Relationships Between Resources of Two Triplesets Described in a Third One. Using our crawler, we can find cases in which a relationship between two resources *r* and *r'*, respectively defined in triplesets *d* and *d'*, was described in another tripleset *d''*. This happens, for example, when the ontologies used by *d* and *d'* are only stored in a different dataset *d''*. In these cases, it is necessary to crawl all triplesets, other than *d* and *d'*, to find the relationship between *r* and *r'*. A traditional crawler following links from *d* would not find any link between *r* and *r'* because it is only declared in *d''*.

Crawling with SPARQL Queries. Our crawler returns richer metadata than a traditional crawler since it uses SPARQL queries, executed over all triplesets. In particular, our crawler discovers not only the links between resources, but also the number of instances related to the crawling terms.

Identifying Resources in Different Languages and Alphabets. Our crawler was able to identify resources in different languages, even in different alphabets, through the *sameAs* and *seeAlso* queries.

Performing Simple Deductions. Using the provenance lists the crawler generates, one may perform simple deductions, using the transitivity of the subclass property, perhaps combined with the *sameAs* relationship. For example, suppose that the crawler discovered that `opencyc:Hit_music` is a subclass of `opencyc:Music`, which in turn has a *sameAs* relationship with `wordnet:synset-music-noun-1`. Then, one may deduce that `opencyc:Hit_music` is a subclass of `wordnet:synset-music-noun-1`.

7 Conclusions and Future Work

This paper presented CRAWLER-LD, a metadata focused crawler for Linked Data. The crawler works in stages, starting with a small set T_0 of generic RDF terms and creating a new set of terms T_{i+1} by enriching T_i with related terms. The tool is engineered as a framework that currently includes three processors, implementing different crawling strategies.

In general, the metadata focused crawler introduced in this paper helps simplify the triplification and the interlinking processes, thereby contributing to the dissemination of Linked Data. Indeed, the results of the crawler may be used: to recommend ontologies to be adopted in the triplification process; and to recommend triplesets to be used in the linkage process.

Finally, the overall crawling process is open to several improvements. For example, we may use summarization techniques to automatically select the initial set of terms. We could also use some sort of caching system to improve overall performance.

Acknowledgements. A This work was partly funded by CNPq, under grants 160326/2012-5, 303332/2013-1 and 57128/2009-9, by FAPERJ, under grants E-26/170028/2008, E-26/103.070/2011 and E-26/101.382/2014, and by CAPES. Microsoft Azure for Research program was also valuable providing capable machines for experimentations.

Appendix 1: Framework Pseudo-Code

CRAWLER-LD(maxLevels, maxTerms, maxFromTerm, maxFromSet; T, C, PR; Q, P, D)

CRAWLER-LD(maxLevels, maxTerms, maxFromTerm, maxFromSet; T, C, PR; Q, P, D)

Parameters: *maxLevels* - maximum number of levels of the breath-first search
maxTerms - maximum number of terms probed
maxFromTerm - maximum number of new terms probed from each term
maxFromSet - maximum number of terms probed from a tripleset, for each term
input: *T* - a set of input terms
C - a list of catalogues of triplesets
PR - a list of processors
output: *Q* - a queue with the terms that were crawled
P - a provenance list for the terms in *Q*
D - a provenance list of the triplesets with terms in *Q*

```
begin   Q, P, D := empty;
        #levels, #terms := 0;
        nextLevel := T;
        while #levels < maxLevels and #terms < maxTerms do
        begin
            #levels := #levels + 1;
            currentLevel := nextLevel;      /* currentLevel and nextLevel are queues of terms */
            nextLevel := empty;
            for each t from currentLevel do
            begin

                    terms += terms;
                    if ( #terms > maxTerms ) then exit;
                    add t to Q;
                    resourcesForEachDataset := (dataset,resourceList) := empty
                    for each p from PR do
                    begin
                    /* use t on the processor p and save the results for each dataset */
                     call (dataset,resultList) := p(t,P,D)
                     add p to resourceForEachDataset
                    end
                    /* limiting results phase */
                    resourcesFromTerm := empty
                    for each dataset d from resourcesForEachDataset
                    begin
                     resultList := results from dataset D on term t;
                     truncate resultList to contain just the first maxFromSet terms;
                     resourcesFromTerm := concatenate(resultList, resourcesFromTerm);
                    end
                    truncate resourcesFromTerm to contain just the first maxFromTerm terms;
                     nextLevel := concatenate(resourcesFromTerm, nextLevel);
            end /* t loop */
        end /* level loop */
        return Q, P, D;
end /* algorithm */
```

Appendix 2: A Comparison Between Swget and CRAWLER-LD for the Music Domain

	Terms retrieved by *swget* or CRAWLER-LD	MV	SW	RC
	(Terms retrieved by *swget*)			
dbpedia:MusicalWork		-	-	-
1	dbpedia:Song	Y	Y	Y
2	dbpedia:Single	Y	Y	Y
3	dbpedia:Album	Y	Y	Y
4	dbpedia:Work	N	Y	N
5	dbpedia:ArtistDiscography	Y	Y	Y
6	dbpedia:Opera	Y	Y	Y
7	dbpedia:EurovisionSongContestEntry	Y	Y	Y
8	owl:Thing	N	Y	N
9	dbpedia:Software	N	Y	N
10	dbpedia:RadioProgram	N	Y	N
11	dbpedia:Cartoon	N	Y	N
12	dbpedia:TelevisionSeason	N	Y	N
13	dbpedia:Film	N	Y	N
14	dbpedia:Website	N	Y	N
15	dbpedia:CollectionOfValuables	N	Y	N
16	dbpedia:WrittenWork	N	Y	N
17	dbpedia:Musical	Y	Y	N
18	dbpedia:Artwork	N	Y	N
19	dbpedia:LineOfFashion	N	Y	N
20	dbpedia:TelevisionShow	N	Y	N
21	dbpedia:TelevisionEpisode	N	Y	N
dbpedia:MusicalArtist		-	-	-
22	dbpedia:Artist	N	Y	N
23	schema:MusicGroup	Y	Y	N
24	dbpedia:Sculptor	N	Y	N
25	dbpedia:Painter	N	Y	N
26	dbpedia:Actor	N	Y	N
27	dbpedia:ComicsCreator	N	Y	N
28	dbpedia:Comedian	N	Y	N
29	dbpedia:FashionDesigner	N	Y	N
30	dbpedia:Writer	N	Y	N
31	dbpedia:Person	N	Y	N
dbpedia:Song		-	-	-
32	schema:MusicRecording	Y	Y	Y
33	dbpedia:MusicalWork	Y	Y	N
dbpedia:Album		-	-	-
34	schema:MusicAlbum	Y	Y	Y
dbpedia:Single		-	-	-
	(No new term retrieved *swget*)			
mo:MusicArtist		-	-	-
35	mo:SoloMusicArtist	Y	Y	Y
36	foaf:Agent	N	Y	N
37	mo:MusicGroup	Y	Y	Y
38	foaf:Person	N	Y	N
39	foaf:Organization	N	Y	N
40	foaf:Group	N	Y	N
mo:MusicalWork		-	-	-
41	mo:Movement	Y	Y	Y
42	frbr:Work	N	Y	N
43	frbr:ScholarlyWork	N	Y	N
44	frbr:ClassicalWork	N	Y	N
45	frbr:LegalWork	N	Y	N
46	frbr:LiteraryWork	N	Y	N
47	frbr:Endeavour	N	Y	N
48	wordnet:Work~2	N	Y	N
mo:Composition		-	-	-
	(No term retrieved)			
	(Terms retrieved only by CRAWLER-LD)			
	(No term retrieved)			

Notes:

- Column Headers / Values:
  "MV" ("Manual Validation"):
  - Y = term relevant for the Music domain
  - N = term not relevant for the Music domain

  "SW" ("Retrieved by *swget*") and
  "RC" ("Retrieved by CRAWLER-LD"):
  - Y = term retrieved by *swget* or CRAWLER-LD
  - N = term not retrieved by *swget* or CRAWLER-LD
- Terms retrieved by *swget* or CRAWLER-LD:
  Retrieved terms: 48
  Relevant terms that were retrieved (identified by "Y" in column "MV"): 14
- Terms retrieved by *swget*:
  Retrieved terms: 48
  Relevant terms that were retrieved (identified by rows with the pattern (Y,Y,-)): 14
  Precision = 14 / 48 = 0.2917
  Recall = 14 / 14 = 1.0
  F_1-Measure = 2 * ((0.2917*1.0) / (0.2917+1.0))
  = 0.4516
- Terms retrieved by CRAWLER-LD:
  Retrieved terms: 11
  Relevant terms that were retrieved (identified by rows with the pattern (Y,-,Y)): 11
  Precision = 11 / 11 = 1.0
  Recall = 11 / 14 = 0.7857
  F_1-Measure = 2 * ((1.0*0.7857) / (1.0+0.7857))
  = 0.8800

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far, lnt'l. J. Seman. Web Info. Sys. **5**(3), 1–22 (2009)
2. Nikolov, A., d'Aquin, M.: Identifying relevant sources for data linking using a semantic web index. In: Proceedings Workshop on Linked Data on the Web. vol. 813 of CEUR Workshop Proceedings, CEUR-WS.org (2011)
3. Nikolov, A., d'Aquin, M., Motta, E.: What should I link to? identifying relevant sources and classes for data linking. In: Proceedings Joint Int'l Semantic Technology Conference, pp. 284–299 (2012)
4. Romero, M.M., Vázquez -Naya, J.M., Munteanu, C.R., Pereira, J., Pazos, A.: An approach for the automatic recommendation of ontologies using collaborative knowledge. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010, Part II. LNCS, vol. 6277, pp. 74–81. Springer, Heidelberg (2010)
5. Saint-Paul, R., Raschia, G., Mouaddib, N.: General purpose database summarization. In: Proceedings 31st Int'l Conference on Very Large Data Bases. VLDB Endowment, pp. 733–744 (2005)
6. Fionda, V., Gutierrez, C., Pirró, G.: Semantic navigation on the web of data: specification of routes, web fragments and actions. In: Proceedings of the 21st Int'l Conference on World Wide Web, pp. 281–290 (2012)
7. Isele, R., Harth, A., Umbrich, J., Bizer, C.: LDspider: an open-source crawling framework for the web of linked data. In: Proceedings Int'l Semantic Web Conference (Posters), Shanghai, China (2010)
8. Ding, L., Pan, R., Finin, T.W., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Gil, Y., Motta, E., Benjamins, V., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 156–170. Springer, Heidelberg (2005)
9. Manola, F., Miller, E.: RDF Primer, W3C Recommendation 10 February 2014 (2004)
10. Brickley, D., Guha, R.V. (eds.): RDF vocabulary description language 1.0: RDF schema. In: W3C Recommendation 10 February 2004 (2004)
11. W3C OWL Working Group: OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation 11 December 2012
12. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF, W3C Recommendation 15 January 2009
13. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets - on the design and usage of void, the 'vocabulary of interlinked datasets'. In: Proceedings Workshop on Linked Da-ta on the Web (LDOW 2009), Madrid, Spain (2009)
14. Leme, L.A.P., Lopes, G.R., Nunes, B.P., Casanova, M.A., Dietze, S.: Identifying candidate datasets for data interlinking. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 354–366. Springer, Heidelberg (2013)
15. Lopes, G.R., Leme, L.A.P.P., Nunes, B.P., Casanova, M.A., Dietze, S.: Recommending tripleset interlinking through a social network approach. In: Proceedings 14th International Conference on Web Information System Engineering, Nanjing, China (Oct. 13–15, 2013), pp. 149–161 (2013)
16. Garlik, S.H., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language. World Wide Web Consortium (2013)
17. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	··· under matter to remain	
Insert in text the matter indicated in the margin		New matter followed by or
Delete	through single character, rule or underline or through all characters to be deleted	or
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	