A Metadata Focused Crawler 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, Rua Marquês de S. Vicente, 225, Rio de Janeiro, Brasil*
[2]*Instituto de Computação, UFF, Rua Passo da Pátria, 156, Niterói, Brasil*

Keywords: Focused Crawler, Tripleset Recommendation, Linked Data.

Abstract: The Linked Data best practices recommend publishers of triplesets to use well-known ontologies in the triplication process and to link their triplesets with other triplesets. However, despite the fact that extensive lists of open ontologies and triplesets are available, most publishers typically do not adopt those ontologies and link their triplesets only with popular ones, such as DBpedia and Geonames. This paper presents a metadata crawler for Linked Data to assist publishers in the triplification and the linkage processes. The crawler provides publishers with a list of the most suitable ontologies and vocabulary terms for triplification, as well as a list of triplesets that the new tripleset can be most likely linked with. The crawler focuses on specific metadata properties, including *subclass of*, and returns only metadata, hence the classification "metadata focused crawler".

1 INTRODUCTION

The Linked Data best practices (Bizer et al., 2009) recommend publishers of triplesets to use well-known ontologies in the triplication process and to link 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 (Nikolov and d'Aquin, 2011; Nikolov et al. 2012; Martínez-Romero, 2010), 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 a crawler that addresses the problem of finding vocabulary terms and triplesets to assist publishers in the triplification and the linkage processes. Unlike typical Linked Data crawlers, the proposed crawler then focuses on metadata with specific purposes, illustrated in what follows.

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 (Saint-Paul et al., 2005) to automatically extract *T* from a set of triplesets.

Then, the publisher submits *T* to the crawler, 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 linkage 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, Section 7 presents the conclusions.

[1] http://datahub.io

[2] http://dbpedia.org

[3] http://www.geonames.org

do Vale Amaral Gomes R., A. Casanova M., Rabello Lopes G. and André P. Paes Leme L..
A Metadata Focused Crawler for Linked Data.
DOI: 10.5220/0004867904890500
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 489-500
ISBN: 978-989-758-028-4
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

2 RELATED WORK

We first compare the proposed crawler to Linked Data crawlers.

Fionda et al. (2012) 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 (Isele et al., 2010) 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. (2005) 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 Section 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. (2011, 2012) 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. (2010) 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.

The crawler proposed in this work 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 (Manola and Miller, 2004), RDF Schema (Brickley and Guha, 2004) 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 (W3C, 2012) 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
- `rdfs:seeAlso` indicates that the subject is generically related to the object
- `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 (Prud’hommeaux and Seaborne,

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

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

2008) 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).

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 (Alexander et al., 2009) 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 A 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.io 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 `rdfs:seeAlso` 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; (2) for each class found, an estimation of the number of instances in each tripleset visited; and (3) a list relating the VoID data of each tripleset with each one of the terms found in (1).

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 (Leme et al., 2013; Lopes et al., 2013).

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, `musicBrains` and `bbcMusic`. The user might then decide that `opencyc:Music` is a good choice to adopt in the triplification process and that `musicBrains` and `bbcMusic` are good choices for the linkage process.

5 A METADATA FOCUSED CRAWLER

This section discusses the construction of the metadata focused crawler. The reader will find a pseudo-code describing the crawler in Annex 1.

5.1 Crawling Queries

The crawler works with catalogues that use the CKAN framework to identify SPARQL endpoints and RDF dumps. 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.

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

[6] http://schema.org

Given *T*, the crawler proceeds in stages (see Section 5.2) to extract new terms using *crawling queries* over all triplesets listed in the catalogues and using *URI dereferencing*.

The crawling queries find new terms that are related to the terms obtained in the previous stage through the following *crawling properties* (see Section 3) `rdfs:subClassOf`, `owl:sameAs` and `rdfs:seeAlso`. Hence, these queries are respectively called *subclass*, *sameAs* and *seeAlso* queries.

Figure 1 shows one of the templates of the *crawling queries* that obtain terms related to a known term *t* through the crawling property *p*.

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

Figure 1: Template of the SPARQL query to obtain a subset of the crawling results.

Notice that, for the properties `owl:sameAs` and `rdfs:seeAlso`, the crawler also uses the template query of Figure 2. For each term *t* to be crawled, it inverts the role of *t*, as shown in Figure 2, when the predicate *p* is `owl:sameAs` and `rdfs:seeAlso`, since these predicates are reflexive and it is reasonable that the description of the term itself will be explained in that order. However, the crawler does not invert the role of *t* when the predicate *p* is `rdfs:subClassOf`, since this predicate is not reflexive.

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

Figure 2: Template of the inverted SPARQL query.

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 of Figure 1.

Another case occurs when the relationship between *C* and *C'* is defined in a third schema *S"*. Similarly to the previous example, we need a subclass query over *S"* to discover this relationship between *C* and *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.

Finally, a special type of crawling query is obtained by replacing *p* in Figure 1 with `rdf:type`. However, in this case, only the overall number of instances found and the total number of instances for each tripleset are retrieved and stored in the result set of the crawling process.

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 with the help of the queries and URI dereferencing, as described in Section 5.1, except for `rdf:type`, which is used only to count the number of instances found.

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 crawler creates 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.
```

5.3 Using VoID to Extract More Information about Triplesets

The crawler will eventually collect a large number of terms and count the number of instances of a reasonable number of classes, declared in many triplesets. These data can be used to extract more metadata about a tripleset by parsing its VoID description, as follows.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

For each tripleset *t* in the catalogues the crawler uses, if *t* has a VoID description *V*, the crawler retrieves all objects *o* from triples of the form (*s*, `void:class`, *o*) declared in *V*. The resources retrieved are compared to all resources the crawler already located. Each new resource found is saved and returned as part of the final output of the entire crawling operation, with an indication that it is also related to tripleset *t* through a VoID description.

Although the crawling process has limiting parameters to avoid time-consuming tasks, the processing of VoID descriptions is simple enough and is, therefore, not subjected to limitations.

6 TESTS AND RESULTS

6.1 Organization of the Experiments

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

We evaluated the crawler over triplesets described in Datahub.io. The tool was able to recover 317 triplesets with SPARQL endpoints. But, despite this number, it could run queries on just over half of the triplesets due to errors in the query parser or simply because the servers were not available.

To execute the tests, we separated three set of terms related to the music and publications application domains. In both cases, we focused on three generic ontologies to create the initial crawling terms, WordNet, DBpedia and Schema.org.

WordNet is a lexical database that presents different meanings for the same word. For example, the term `wordnet:synset-music-noun-1` means “an artistic form of auditory communication incorporating instrumental or vocal tones in a structured and continuous manner”. In addition, the term `wordnet:synset-music-noun-2` is defined as “any agreeable (pleasing and harmonious) sounds; "he fell asleep to the music of the wind chimes"”. Both are music, but have different meanings.

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.

We elected these three ontologies as the most generic ones. All three have a collection of terms that covers numerous domains and could be used together to determine an initial set that represents the user intentions. Of course, if a user has good knowledge about a domain, he can adopt more specific ontologies to determine the initial crawling terms. In the examples that follow, we use the abbreviations shown in Table 1.

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/
`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://yago-knowledge/resource/

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
mo:MusicalWork
```

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

[8] http://musicontology.com/

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

Table 2: Related Terms.

Related terms	
Query type	**Description**
(a) Related terms for mo:MusicArtist	
subclass	mo:MusicGroup mo:SoloMusicArtist
instance	103,541 instances, mostly from lastfm
(b) Related terms for mo:MusicalWork	
subclass	mo:Movement
instance	16,833 instances found in multiple databases like dbtune and academic music databases
(c) Related terms for dbpedia:MusicalWork	
subclass	dbpedia:Album dbpedia:Song dbpedia:Single dbpedia:Opera dbpedia:ArtistDiscography and 21,413 classes from yago
sameAs	dbpedia:MusicGenre umbel:MusicalComposition
seeAlso	lastfm:Syfin lastfm:Kipling lastfm:Pandemic lastfm:Ardcore lastfm:Lysis lastfm:Freakhouse lastfm:Saramah lastfm:Akouphen lastfm:Freakazoids lastfm:Cyrenic lastfm:Phender twitter:Ariadne_bullet
instance	145,656 instances
(d) Related terms for dbpedia:Song	
Own URL	dbpedia:EurovisionSongContestEntry
sameAs	schema:MusicRecording
subclass	dbpedia:EurovisionSongContestEntry
seeAlso	lastfm:Apogee lastfm:Brahman lastfm:Anatakikou lastfm:Sakerock lastfm:8otto lastfm:Cro-Magon lastfm:Ladz Plus 7 lastfm resources in Japanese
instance	10,987 instances from multiple language versions of dbpedia, lastfm and others
(e) Related terms for dbpedia:Album	
Own URL	freebase:en.Album opencyc:Album
subclass	nerdeurocom:Album and 17,222 subclasses, mostly from yago
sameAs	schema:MusicAlbum freebase:en.Album dbpedia:Sophomore_Album and some dbpedia:Album classes from other Wikipedia languages
instance	100,090 instances from multiple language versions of dbpedia and others
(f) Related terms for dbpedia:MusicalArtist	
seeAlso	lastfm:Krackhead
sameAs	dbpedia:Musician umbel:MusicalPerformer
subclass	dbpedia:Instrumentalist dbpedia:BackScene and 2,178 subclasses from yago
instance	49,973 instances from multiple language versions of dbpedia
(g) Related terms for dbpedia:Single	
seeAlso	last.fm:Toxin last.fm:Dethrone last.fm:Burdeos last.fm:Sylence twitter:joint_popo last.fm:Toximia last.fm:Alcoholokaust last.fm:Electromatic last.fm:Mighty+Atomics
subclass	3,414 subclasses, the majority from yago
instance	44,623 instances

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 similar results: nearly 2,000 resources were found in bbcMusic and musicBrainz:data, which are large databases about the music domain; and the *seeAlso* query pointed to an artist, lastfm:Hadas. As *seeAlso* provides additional data about the subject, we speculate that the result the crawler returned represents a mistake made by the database creator.

Table 2(b) shows the results of Stage 1 for mo:MusicalWork. Note that the crawler found a variety of instances from multiple databases, mainly on universities. On Stage 2, when processing mo:Movement, the crawler found a *seeAlso* reference to lastfm:Altmodisch.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

On Stage 1, when processing mo:Composition, the crawler found 13 instances, but no related terms.

Table 2(c) shows the results of Stage 1 for the first DBpedia term, dbpedia:MusicalWork. The crawler found 5 subclasses from DBpedia and more than 20 thousand subclasses from the yago tripleset. This unusual result is due to the segmentation used by yago. For example, there are subclasses for segmenting records by artist, by historical period, and even by both.

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 3,423 instances, but no new term. The processing of umbel:MusicalComposition returned 1809 instances and dbpedia:MusicGenre retrieved 7,808 new instances.

Table 2(d) shows the results of Stage 1 for dbpedia:Song. The crawler found the most diversified results in terms of query types and query results. It was able to identify resources in different languages (such as Portuguese and Greek), which was only possible because it focused on metadata. Crawlers that use text fields (Nikolov and d'Aquin, 2011) can only retrieve data in the same language as that of initial terms.

On Stage 2, when processing dbpedia:EurovisionSongContestEntry, the crawler obtained three subclasses from yago, a *sameAs* relationship with schema:MusicRecording and found the same result of dbpedia:Song for the *seeAlso* property. The other resource probed on the Stage 2 was schema:MusicRecording which returned no instances or new crawling terms.

Table 2(e) shows the results of Stage 1 for dbpedia:Album. The processing of this term also produced an interesting result. The *sameAs* query found a small number unique relationships, but found some dbpedia:Album in other languages. One may highlight the opencyc:Album class, for which the crawler was able to find 245 instances.

Table 2(f) shows the results of Stage 1 for dbpedia:MusicalArtist. The processing of this term exhibited results similar to those obtained by processing dbpedia:Album, in terms of quantity of subclasses. Therefore, it was possible to recover results in multiple languages.

On Stage 2, when processing dbpedia:Musician, the crawler found over 163 *sameAs* terms, the majority of them pointing to DBpedia in other languages (even in non-latin alphabets). On the other hand, the *seeAlso* query found over 50 terms, but none of them seems related to the subject. When processing umbel:MusicalPerformer, the crawler retrieved one subclass, umbel:Rapper, and over 6,755 instances from a variety of triplesets.

Table 2(g) shows the results of Stage 1 for dbpedia:Single. As for other resources from DBpedia, the crawler was able to find a large number of subclasses from yago tripleset. In addition, it found more than 40 thousand instances from different 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: wordnet:synset-music-noun-1. On Stage 2, the crawler found a new *sameAs* relationship to opencyc:Music.

Finally, we remark that, when we selected the terms to evaluate, we expected to find relationships between DBpedia and Music Ontology, which did not happened. 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. While the queries based on Schema.org practically returned no results, queries on Aktors returned enough 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

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

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.

The VoID processing, as discussed in Section 5.3, was not able to find any new information. In fact, in a more detailed analysis, it was clear that VoID seems to be a neglected feature. From the initial 317 triplesets, only 102 had the VoID description stored in Datahub.io, and only 8 had any triple with the property `void:class` (which were not related to our test domains).

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).

Table 3 shows that the minimum time was 14 minutes, 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; (4) cache features are used; (3) queries are optimized to reduce the number of requests.

Table 3: Performance evaluation.

Term	Proc. time (minutes)
Music domain	
`mo:MusicArtist`	70
`mo:MusicalWork`	28
`mo:Composition`	14
`dbpedia:MusicalWork`	183
`dbpedia:Song`	163
`dbpedia:Album`	173
`dbpedia:MusicalArtist`	167
`dbpedia:Single`	186
`wordnet:synset-music-noun-1`	24
Publications domain	
`schema:TechArticle`	29
`schema:ScholarlyArticle`	47
`akt:Article-Reference`	14
`akt:Article-In-A-Composite-Publication`	28
`akt:Book`	14
`akt:Thesis-Reference`	14
`akt:Periodical-Publication`	28
`akt:Lecturer-In-Academia`	14
`akt:Journal`	14

6.3 A Comparison with *Swget*

We opted for a direct comparison between the proposed crawler 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 Section 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 Section 6.2, specifying which crawled property *swget* should follow. Table 4 shows the number of terms *swget* found, for each term and crawling property.

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

Term	*subclass*	*sameAs*	*seeAlso*	*type*
mo:MusicArtist	4	0	0	3
mo:MusicalWork	7	0	0	3
mo:Composition	0	0	0	3
dbpedia:MusicalWork	16	1	0	3
dbpedia:Song	6	1	0	3
dbpedia:Album	6	1	0	3
dbpedia:MusicalArtist	9	1	0	3
dbpedia:Single	6	1	0	3

Based on the experiments with *swget* and the crawler, we compiled the list of terms shown in Table 5 of Annex 2. We excluded the terms retrieved from `yago` to avoid unbalancing the experiment in favor of the crawler. Then, we manually inspected the terms and marked, in Table 5, those that pertain to the Music domain and those that *swget* and the proposed crawler found.

The results detailed in Annex 2 can be summarized by computing the precision and recall obtained by *swget* and our crawler for the list of terms shown in Table 5:

- *swget*: precision = 35% recall = 24%
- crawler: precision = 95% recall = 91%

These results should be interpreted as follows.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Swget achieved a much lower precision since it finds more generic and more specific terms at the same time, while our crawler 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.

The crawler achieved a better recall since, in part, it was able to, given two classes defined in different triplesets, uncover relationships between the classes described in a third tripleset. Indeed, *swget* processed `umbel:MusicalPerformer` using properties `rdfs:subClassOf` and `owl:sameAs`. Our expectation was that it would be able to find the class `dbpedia:MusicalWork`, as our crawler was, which did not happen. A quick analysis showed that the relationship between both classes was not described in any of the original triplesets, but in a third tripleset, `http://linkeddata.uriburner.com/`.

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 ours.

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.

VoID description. The VoID processing seems to be an adequate solution to a faster access to tripleset information. Despite the VoID expressivity, most triplesets used in our experiments had a simplistic VoID description available. Hence, our crawler hardly found new data using the VoID descriptions.

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 found 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 was 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`.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

7 CONCLUSIONS AND FUTURE WORK

This paper presented a metadata focused crawler for Linked Data. The crawler starts with a small set *T* of generic RDF terms and enriches *T* by identifying subclasses, equivalences (`sameAs` property) and related terms (`seeAlso` property).

In general, the metadata focused crawler introduced in this paper helps simplify the triplification and linkage 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; to recommend triplesets to be used in the linkage process; and to increase the quality of VoID descriptions.

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 may also optimize the crawling process by combining the crawling queries into a single query and by using caching to avoid bandwidth issues.

ACKNOWLEDGEMENTS

This work was partly funded by CNPq, under grants 160326/2012-5, 303332/2013-1 and 57128/2009-9, and by FAPERJ, under Grants E-26/170028/2008 and E-26/103.070/2011.

REFERENCES

Alexander, K. Cyganiak, R., Hausenblas, M., Zhao, J., 2009. Describing linked datasets - on the design and usage of void, the ‘vocabulary of interlinked datasets’. Proc. Workshop on Linked Data on the Web (LDOW’09), Madrid, Spain.

Bizer, C., Heath, T, Berners-Lee, T., 2009. Linked Data - The Story So Far, *Int’l. Journal on Semantic Web and Info. Sys.*, 5 (3), pp. 1-22.

Brickley, D., Guha, R.V. (eds.), 2004. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation* 10 February 2004.

Ding, l., Pan, R., Finin, T., Joshi, A., Peng, y., Kolari, P., 2005. Finding and ranking knowledge on the semantic web. Proc. 4th Int’l. Conf. on the Semantic Web, Springer-Verlag, pp. 156-170.

Fionda, V., Gutierrez, C., Pirró, G., 2012. Semantic navigation on the web of data: specification of routes, web fragments and actions. Proc. 21st Int’l. Conf. on World Wide Web, pp. 281-290.

Isele, R., Harth, A., Umbrich, J., Bizer, C., 2010. LDspider: An open-source crawling framework for the Web of Linked Data. Proc. Int’l. Semantic Web Conf. (Posters), Shanghai, China.

Leme, L.A.P.P., Lopes, G.R., Nunes, B.P., Casanova, M.A., Dietze, S., 2013. Identifying candidate datasets for data interlinking. Proc. 13th Int’l. Conf. on Web Engineering, Aalborg, Denmark (July 8-12, 2013), pp. 354-366.

Lopes, G.R., Leme, L.A.P.P., Nunes, B.P., Casanova, M.A., Dietze, S., 2013. Recommending Tripleset Interlinking through a Social Network Approach. Proc. 14th Int’l. Conf. on Web Information System Engineering, Nanjing, China (Oct. 13-15, 2013), pp. 149-161.

Manola, F., Miller, E., 2004. RDF Primer, *W3C Recommendation* 10 February 2014.

Martínez-Romero, M., Vázquez-Naya, J., Munteanu, C., Pereira, J., Pazos, A., 2010. An approach for the automatic recommendation of ontologies using collaborative knowledge. Proc. 14th Int’l. Conf. on Knowledge-based and Intelligent Information and Engineering Systems, Part II, Springer, pp. 74-81.

Nikolov, A., d'Aquin, M., 2011. Identifying Relevant Sources for Data Linking using a Semantic Web Index. Proc. Workshop on Linked Data on the Web. Volume 813 of CEUR Workshop Proceedings, CEUR-WS.org.

Nikolov, A., d'Aquin, M., Motta, E., 2012. What should I link to? Identifying relevant sources and classes for data linking. Proc. Joint Int’l. Semantic Technology Conference, pp. 284-299.

Prud’hommeaux, E., Seaborne, A., 2008. SPARQL Query Language for RDF, *W3C Recommendation* 15 January 2009.

Saint-Paul, R., Raschia, G., Mouaddib, N., 2005. General purpose database summarization. Proc. 31st Int’l. Conf. on Very Large Data Bases. VLDB Endowment, pp. 733-744.

W3C OWL Working Group, 2012. OWL 2 Web Ontology Language □Document Overview (Second Edition). *W3C Recommendation* 11 December 2012.

Wang, J., Wen, J., Lochovsky, F., Ma, W., 2004. Instance-based schema matching for web databases by domain-specific query probing. Proc. 30th Int’l. Conf. on Very Large Data Bases. Vol. 30. VLDB Endowment, pp. 408-419.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

APPENDIX

1 Crawler Pseudo-code

```
                                   genericCrawlingQuery(d, S, t, p; R);
input:        d   -  direction of the query (“direct” or “reverse”)
              S   -  a SPARQL Endpoint or a RDF Dump to be queried
              t   -  a crawling term
              p   -  a predicate
output:       R   -  a set of terms crawled from t
begin
      if d == “direct”
       then   R :=   execute  SELECT distinct ?item  WHERE   { ?item  p  <t> } over S
       else   R :=   execute  SELECT distinct ?item  WHERE   { <t>  p  ?item } over S
      return R;
end
```

```
CRAWLER(maxLevels, maxTerms, maxFromTerm, maxFromSet; T, C; 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
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
                    add t to Q;
                    /*           crawling by dereferencing                              */
                    S := downloaded RDF content obtained by dereferencing t;
                    R1 := empty;
                    for each predicate p in { rdfs:subClassOf,owl:sameAs,rdfs:seeAlso } do
                    begin
                        if p == “rdfs:subClassOf” then d := “direct” else d := “inverse”;
                        genericCrawlingQuery( d, S, t, p; RTEMP );
                        if (RTEMP not empty)
                              then   begin add (t, p, RTEMP, S) to P;
                                           R1 := concatenate(R1, RTEMP);
                                     end
                    end
                    /*           crawling by direct querying the triplesets in C          */
                    R2 := empty;
                    for each tripleset S from the catalogues in C do
                    begin
                        RS := empty;
                        for each predicate p in { rdfs:subClassOf,owl:sameAs,rdfs:seeAlso } do
                        begin
                                    genericCrawlingQuery( “direct”, S, t, p; RTEMP );
                                    if (RTEMP not empty)
                                    then   begin add (t, p, RTEMP, S) to P;
                                                 RS := concatenate(RS, RTEMP);
                                           end
                        end
                        if (RS not empty)
                              then   begin add (t, S) to D;
                                           truncate RS to contain just the first maxFromSet terms;
                                           R2 := concatenate(R2, RS);
                                     end
                    end
                    RT := concatenate(R1, R2)
                    for each u in RT do
                    begin
                        #termsFromTerm := #termsFromTerm +1;
                        #terms := #terms +1;
                              if ( #termsFromTerm > maxFromTerm or #terms > maxTerms ) then exit;
                              add u to nextLevel;
                        end
            end
                end
        return Q, P, D;
    end
```

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

2 A Comparison Between *Swget* and the Proposed Crawler for the Music Domain

Terms retrieved by *swget* or crawler		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
22	dbpedia:Song	Y	Y	Y
23	dbpedia:Single	Y	Y	Y
dbpedia:MusicalArtist		-	-	-
24	dbpedia:Artist	N	Y	N
25	schema:MusicGroup	Y	Y	N
26	dbpedia:Sculptor	N	Y	N
27	dbpedia:Painter	N	Y	N
28	dbpedia:Actor	N	Y	N
29	dbpedia:ComicsCreator	N	Y	N
30	dbpedia:Comedian	N	Y	N
31	dbpedia:FashionDesigner	N	Y	N
32	dbpedia:Writer	N	Y	N
33	dbpedia:Person	N	Y	N
dbpedia:Song		-	-	-
	(No new term retrieved *swget*)			
dbpedia:Album		-	-	-
	(No new term retrieved *swget*)			
dbpedia:Single		-	-	-
	(No new term retrieved *swget*)			
mo:MusicArtist		-	-	-
34	mo:SoloMusicArtist	Y	Y	Y
35	foaf:Agent	Y	Y	N
36	mo:MusicGroup	Y	Y	Y
37	foaf:Person	Y	Y	N
38	foaf:Organization	Y	Y	N
mo:MusicalWork		-	-	-
39	mo:Movement	Y	Y	Y
40	frbr:Work	N	Y	N
41	frbr:ScholarlyWork	N	Y	N
42	frbr:ClassicalWork	N	Y	N
43	frbr:LegalWork	N	Y	N
44	frbr:LiteraryWork	N	Y	N
45	frbr:Endeavour	N	Y	N
46	wordnet:Work~2	N	Y	N
mo:Composition		-	-	-
	(No term retrieved)			
(Terms retrieved only by crawler)				
47	umbel:MusicalComposition	Y	N	Y
48	schema:MusicRecording	Y	N	Y
49	freebase:en.Album	Y	N	Y
50	opencyc:Music	Y	N	Y
51	opencyc:Album	Y	N	Y
52	nerdeurocom:Album	Y	N	Y
53	schema:MusicAlbum	Y	N	Y
54	dbpedia:Sophomore_Album	Y	N	Y
55	dbpedia:Musician	Y	N	Y
56	umbel:MusicalPerformer	Y	N	Y
57	umbel:Rapper	Y	N	N
58	dbpedia:Instrumentalist	Y	N	Y
59	dbpedia:BackScene	N	N	Y
60	dbpedia:MusicGenre	Y	N	Y
61	freebase:en.Album	Y	N	Y
	36 items from lastfm	Y	N	Y
	2 items from twitter	N	N	Y

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

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 the crawler"):
  - Y = term retrieved by *swget* or the *crawler*
  - N = term not retrieved by *swget* or the *crawler*
- Terms retrieved by *swget* or crawler:
  Retrieved terms: 99
  Relevant terms that were retrieved (identified by "Y" in column "MV"): 66
- Terms retrieved by *swget*:
  Retrieved terms: 46
  Relevant terms that were retrieved (identified by rows with the pattern (Y,Y,-)): 16
  Precision = 16 / 46 = 0.35
  Recall = 16 / 66 = 0.24
- Terms retrieved by the crawler:
  Retrieved terms: 63
  Relevant terms that were retrieved (identified by rows with the pattern (Y,-,Y)): 60
  Precision = 60 /63 = 0.95
  Recall = 60/66 = 0.91