# On Managing Prefixes For Vocabularies Published in LOD

Ghislain Auguste Atemezing[1], Bernard Vatant[2], Pierre-Yves Vanderbussche[3] and Raphaël Troncy[1]

[1] EURECOM, Sophia Antipolis, France, `<firstName.lastName@eurecom.fr>`
[2] Mondeca, Paris, France, `<bernard.vatant@mondeca.com>`
[3] Fujitsu, Galway, Ireland, `<py.vanderbussche@fujitsu.com>`

**Abstract.** As the vocabularies are more and more reused in Linked Data ecosystem, there is a need to manage the prefixes associated to their Uniform Resource Identifiers (URIs). Developers and data modellers need a consistent way to look up easily to unique namespaces and retrieve exactly the same URI. This could help to refer easily to the vocabularies just by the prefix, namespace or QName. Who has ever experiment how difficult is to remember the prefix for `http://purl.org/dc/elements/1.1/`, using *dcterms* instead of the official *dce*; while *foaf* is widely accepted to be `http://xmlns.com/foaf/0.1/`. However, the practice is not always the case that a namespace referred to the same couple *prefix, URI*, as developers don't have time or enough control on what some services they use in their daily tasks can help them managing those different prefixes. In this paper, we propose a solution of aligning two services with the aim at managing and harmonizing vocabularies namespaces. We use prefix.cc that provide a look up service for namespaces in general and Linked Open Vocabularies (LOV) that extract metadata for vocabularies and help retrieving them as well. We address three different scenarios: (i) conflicts between prefix.cc and LOV; (ii) prefixes in LOV not present in prefix.cc and (iii) URIs in prefix.cc that are actually LOV-able vocabularies. For each of the above issues, we specify how we solve them with some actions ranging from updating the different services to contacting the editors of the vocabularies for fixing clashes prefixes founded. Finally the new API released by LOV help checking those namespaces in prefix.cc are actually vocabularies to be inserted in the LOV ecosystem.

**Keywords:** vocabulary consumption, Linked Open Vocabularies, prefix.cc, namespaces reconciliation, Vocabulary Management,

## 1 Introduction

The LOV initiative[1] aims at promoting and facilitating the reuse of vocabularies used to describe linked data. LOV provides a dataset, expressed in RDF

[1] `http://lov.okfn.org/dataset/lov/`

language, which identifies and describes such vocabularies using various metadata, including relationships showing how they rely on each other. The Web of data is by definition uncontrolled and constantly evolving. As well as the Web of Data, LOV task has the intrinsic characteristic of being unfinished and incomplete, with a specific scope to include only vocabularies defining classes and properties, also known as Metadata Element Sets [7] and to exclude concept schemes (also known as Value Vocabularies) usually expressed in SKOS [8]. The particularity of LOV is to focus on linked reusable vocabularies designed for the description of linked data. Beyond the simple inventory, it adds to metadata explicitly provided in the vocabularies themselves information harvested either in documentation or through exchanges with vocabulary curators. The result is the definition of relationships between vocabularies, which are automatically inferred, enabling the curators and potential users to assess at a glance the connectivity of a vocabulary with the ecosystem without having to explore it thoroughly. Another feature is the timeline view of vocabulary versions, allowing at a glance to figure out if the vocabulary is recent or old, frequently updated or not, and in the best case to retrieve stored versions.

On the other side, prefix.cc[2] is the service aims at simplifying the tasks in the work of remembering and looking up URI prefixes by RDF developers. It provides ways at looking up prefixes from both from the search box or directly by typing URLs into a browser bar, such as `http://prefix.cc/rdf`. The service allows anyone to add new prefix mappings, leading sometimes to conflicts. However, the only mechanism for multiple conflicting URIs submitted with the same namespace, is to vote by upvoting or downvoting a given URI, with the restriction of only one vote or namespace submission per day.

Nevertheless, prefix.cc has a limitation per its scope, as it is not intended to manage the conflicts among different namespaces and prefixes. It simply reflects the moving, wild, inconsistent prefix/namespace mappings, or trying to help bringing a bit of order and governance in it. At some point, through usage, some prefixes become more that simple local short cuts, they become de facto identifiers built-in applications etc. Our aim is to be possible through services both prefix.cc and LOV be able to distinguish "stable" prefixes that all sensible data providers, vocabulary publishers and applications should stick to and avoid to conflate with. Starting of course with standards such as: `rdf`, `rdfs, owl, skos`, but also some others de facto " *quasi-standards*" such as `dc, dcterms, foaf, sioc, void, geo, yago`, etc.

The remainder of this paper is structured as follows. In Section 2, we motivate the challenge of aligning and reconciling prefixes form both services . In Section 3, we present the process of detecting the conflicts where the same prefix claims having two different namespaces. We also provide some actions undertaken in such cases as well as how we update prefix.cc from the prefix/-namespace from LOV. The goal is to have the intersection of both services equal to the smallest subset. In Section 4, we check in prefix.cc those URIs that could be LOV-able vocabularies, to be added in LOV. By doing so, we present

[2] `http://prefix.cc/about`

the results of using an API checker related to LOV Bot[3] to help the work of finding more vocabularies spread on the wild. We discuss some lessons learned and recommendations (Section 5) before concluding and outlining future work (Section 6).

## 2 Motivation

Many developers uses prefix.cc to look up for prefix/namespaces while consuming RDF data. At the same time, it is more and more recommended to reuse vocabularies in Linked Data. Prefix.cc and LOV are respectively addressing those two issues separately. Many other tools are created around the dataset[4] of prefix.cc, like Triple-Checker [5], a tool aims at finding typos and common errors in RDF data; while some tools for assisting in the process of publishing Government Linked Data, such as Datalift [9] already integrate LOV in their workflow, at the modelling and reusing vocabularies stage.

The purpose of this work is to make possible the two services "working" together to manage the prefixes/namespaces of the vocabularies published in Linked Data. One of the benefits is to solve different types of conflicts which could occurred during the process of aligning namespaces in both services, and ultimately find a more stable solution to have both services updated and synchronized when any type of event (adding/removing) or (upvoting/downvoting) are likely to appear respectively in LOV or prefix.cc.

## 3 Aligning LOV with Prefix.cc

In this section, we present how we perform the alignment and mapping tasks between the two services LOV and prefix.cc. The main goals are to align Qnames to a unique URI in LOV side and make sure that all the vocabularies in LOV are actually inserted in prefix.cc.

### Technological Requirements

We need to perform SPARQL queries that grabs the all the file of prefix.cc `http://prefix.cc/popular/all.file.vann` in the FROM clause, and compares it to the contents of the LOV SPARQL endpoint[6] via a SERVICE[7] call. To be more generic and standards-compliant, the queries could be run with the Jena ARQ command-line tool to produce a CSV, JSON file that could be easily consumed either by the prefix.cc backend via phpMyAdmin or by LOV backend.

[3] http://lov.okfn.org/dataset/lov/bot/
[4] http://prefix.cc/popular/all.file.txt
[5] https://github.com/cgutteridge/TripleChecker
[6] http://lov.okfn.org/endpoint/lov
[7] http://www.w3.org/2009/sparql/docs/fed/service

### 3.1 First Task: prefixes in LOV not present in Prefix.cc

We first have to compute $< LOV > INTERSECTS < PREFIX.CC >$ and then $< LOV > MINUS\{< LOV > INTERSECTS < PREFIX.CC >\}$. The following SPARQL query finds namespace URIs in LOV that don't exist in prefix.cc, along with their LOV prefix.

```
 PREFIX vann: <http://purl.org/vocab/vann/>
SELECT ?prefix ?lovURI
FROM <http://prefix.cc/popular/all.file.vann> {
 SERVICE <http://lov.okfn.org/endpoint/lov> {
   SELECT ?prefix ?lovURI {
     [] vann:preferredNamespacePrefix ?prefix;
       vann:preferredNamespaceUri ?lovURI;
   }
 }
 FILTER(NOT EXISTS { [] vann:preferredNamespaceUri ?lovURI })
 OPTIONAL {
   [] vann:preferredNamespacePrefix ?prefix;
     vann:preferredNamespaceUri ?pccURI;
 }
}
ORDER BY ?prefix
```

The first results[8] shown the following: $card(LOV) \bigcap card(PREFIX.cc) = 188$[9] and $card(Diff(LOV, PREFIX.cc)) = 133$[10] prefixes in LOV not yet registered in prefix.cc. At this point, a first batch of 80 prefixes/namespaces from LOV where safely imported in prefix.cc since there were no conflicts. For the remaining conflicting ones, they needed more in-deep process and analysis.

### 3.2 Second Task: Dealing with Conflicts between Prefix.cc and LOV

In the process of alignment, there were two types of conflicts identified and where an adequate solutions where provided:

- **Clashes:** Cases where we have in both services the same prefix but different URIs.
- **Disagreements on preferred namespace:** where for the same URI, we found different prefixes.

[8] The results shown here were conducted in two weeks between March, 2nd and March, 20th 2013, as this work is always a continuous task. As of the starting of the experiments, card(LOV) = 321 vocabularies, card(Prefix.cc) = 925.

[9] http://www.eurecom.fr/~atemezin/iswc2013/experiments/firstAlignments/intersection-prefixLOV-02-03.csv

[10] http://www.eurecom.fr/~atemezin/iswc2013/experiments/firstAlignments/inLovNotINPrefixcc-02-03.csv

**Clashes:** The following query identifies those clashes (30) vocabularies. In Table 1, we identify seven different type of issues to deal with, such as (i) real conflicts, (ii) URIs are 404, (iii) URIs are obsolete versions and (iv) both URIs redirecting to the same resource.

```
PREFIX vann: <http://purl.org/vocab/vann/>
SELECT ?prefix ?lovURI ?pccURI
FROM <http://prefix.cc/popular/all.file.vann> {
  SERVICE <http://lov.okfn.org/endpoint/lov> {
    SELECT ?prefix ?lovURI {
      [] vann:preferredNamespacePrefix ?prefix;
        vann:preferredNamespaceUri ?lovURI;
    }
  }
  FILTER(NOT EXISTS { [] vann:preferredNamespaceUri ?lovURI })
  [] vann:preferredNamespacePrefix ?prefix;
    vann:preferredNamespaceUri ?pccURI;
}
ORDER BY ?prefix
```

| Type of issues | # Vocabularies | Percentage |
|---|---|---|
| Real conflicts | 6 | 20% |
| pccURI is 404 | 4 | 13,3% |
| pccURI and lovURI redirect to same resource | 8 | 26,67% |
| lovURI already in prefix.cc as secondary | 7 | 23,3% |
| pccURI is an obsolete version | 3 | 10% |
| lovURI is an older version | 1 | 3,3% |
| lovURI is 404 | 1 | 3,3% |

**Table 1.** Types of issues encountered on the clashes vocabularies

**Disagreements on namespaces:** The general idea is that if vocabulary editors have not included explicitly a `vann:preferredNamespacePrefix` in their description, the curators of LOV are free to change it and put whatever seems appropriate. At the same time, in prefix.cc, having multiple prefixes for the same namespace IRI in not a problem. However, we computed those prefixes in LOV that have different prefixes in prefix.cc. The following query retrieves the URIs falling in those disagreements:

```
PREFIX vann: <http://purl.org/vocab/vann/>
SELECT ?prefix ?lovURI ?prefixcc
FROM <http://prefix.cc/popular/all.file.vann> {
  SERVICE <http://lov.okfn.org/endpoint/lov> {
    SELECT ?prefix ?lovURI {
      [] vann:preferredNamespacePrefix ?prefix;
        vann:preferredNamespaceUri ?lovURI;
    }
  }
  FILTER (?pccURI = ?lovURI && ?prefix != ?prefixcc)
  OPTIONAL {
    [] vann:preferredNamespacePrefix ?prefixcc;
      vann:preferredNamespaceUri ?pccURI;
  }
}
ORDER BY ?prefix
```

From the results of this task (61 cases), we have three actions to perform:

- add the lovPrefix to pccPrefix (e.g: adding `geod:http://vocab.lenka.no/geo-deling#`) to existing `ngeoi` in *pccPrefix*.)
- add more URIs to the existing prefix in pccPrefix (e.g: adding `prov:http://purl.org/net/provenance/ns#`) to existing `hartigprov, prv` in *pccPrefix*)
- change a prefix in LOV (e.g: lovPrefix `dc` for `http://purl.org/dc/terms` not in the list $\{dcterm, dcq, dct, dcterms\}$. So, we use dcterms in LOV instead.
- No changes when the lovPrefix is contains in the set of pccPrefix.

### 3.3 Social Aspects

It is also part of the alignment process to contact the authors, creators or maintainers (if exist) of vocabularies to involve them as well in the process of changing prefixes, and agree with them to fix some issues regarding their vocabularies[11]. By doing so, we use any social platform (Google +, Twitter, etc.) and email contacts provided in the metadata. Table 2 summarizes some cases of real conflicts where the LOV curators have to find and contact the editors of the vocabularies for negotiation.

## 4 Finding Vocabularies in Prefix.cc

In this scenario, we want to find out in prefix.cc, which of the couples (prefix, URI) could be potentially a vocabulary to be further assess to be included in LOV catalogue. To address this issue, we first compute all the differences on prefix.cc NOT in LOV, i.e. $PREFIX.CC$ MINUS ($LOV < INTERSECT > PREFIX.CC$), performing the following SPARQL query:

[11] As of March 20th, 2013; the first process of aligning LOV and prefix.cc ended with all the vocabularies of LOV inserted in prefix.cc.

| prefix | lovURI | pccURI | Remark |
|---|---|---|---|
| sp | http://data.lirmm.fr/ontologies/sp# | http://spinrdf.org/sp# | contact editor at LIRMM (*sp* ⇒ *osp*) |
| scot | http://scot-project.net/scot/ns# | http://scot-project.org/scot/ns# | contact editors at lovURI |
| media | http://purl.org/media# | http://purl.org/microformat/hmedia/ | contact editors for negotiation |
| pro | http://purl.org/spar/pro/ | http://purl.org/hpi/patchr# | contact editors for negotiation |
| swp | http://www.w3.org/2004/03/trix/swp-1/ | http://www.w3.org/2004/03/trix/swp-2/ | contact editors, fix on LOV side |
| wo | http://purl.org/ontology/wo/core# | http://purl.org/ontology/wo/ | contact editors |
| idemo | http://rdf.insee.fr/def/demo# | http://rdf.insee.fr/graphes/def/demo# | to resolve with INSEE |

**Table 2.** LOV prefix.cc conflicts resolution yielding to contact vocabularies editors for negotiation

```
PREFIX vann: <http://purl.org/vocab/vann/>
SELECT  DISTINCT ?pccURI
FROM <http://prefix.cc/popular/all.file.vann> {
   [] vann:preferredNamespacePrefix ?prefixcc;
   vann:preferredNamespaceUri ?pccURI.
  FILTER (NOT EXISTS {
  SERVICE <http://lov.okfn.org/endpoint/lov> {
    SELECT ?prefixcc ?pccURI {
      [] vann:preferredNamespacePrefix ?prefixcc;
        vann:preferredNamespaceUri ?pccURI;
    }
  }
  })
  }  order by ?prefixcc
```

The result gives us a file of 742 containing the URIs to be checked[12].

## 4.1 LOV Check API

We have implemented an API that could help the users to perform remotely a useful service of LOV for checking vocabularies. The Check API[13] allows a user to run the LOV BOT over a distant vocabulary. It takes as parameter the vocabulary URI to process and the time out (integer) specified to stop the process. The result of this action is a dictionary with 26 main keys; from which we are interested in using only 8 of them, which are:

– **uri** (string) – uri of the vocabulary.
– **namespace** (string) – namespace of the vocabulary.

[12] http://www.eurecom.fr/~atemezin/iswc2013/experiments/output/botAnalysis_experiment4.js
[13] http://lov.okfn.org/dataset/lov/apidoc/

- **prefix** (string) – prefix of the vocabulary
- **inLOV** (boolean) – indicates if the vocabulary is already in the Linked Open Vocabularies ecosystem.
- **nbClasses** (int) – Number of classes defined in the vocabulary namespace.
- **nbProperties** (int) – Number of properties defined in the vocabulary namespace.
- **dateIssued** (string) – Vocabulary date of issue.
- **title** (Taxonomy) – List of titles with language information if available.

The code below gives a sample output of the response of our algorithm for `http://ns.aksw.org/Evolution/` the retrieving some metadata describing it as a potential vocabulary with properties and classes, title and the prefix.

```
[caption={Sample output of a response of the Check API}]
{
            "dateIssued": "None",
            "inLOV": false,
            "namespace": "http://ns.aksw.org/Evolution/",
            "nbClasses": 14,
            "nbProperties": 9,
            "pccURI": "http://ns.aksw.org/Evolution",
            "prefix": "ns0",
            "title": [
                {
                    "dataType": null,
                    "language": null,
                    "value": "OntoWiki Evolution Pattern Ontology"
                }
            ],
            "uri": "http://ns.aksw.org/Evolution/"
        },
```

### 4.2 Experiments

We wrote a small script calling the LOV Check API on the URIs in prefix.cc for determining the candidates vocabularies to be inserted in LOV, using the algorithm in 1. Due to some instabilities of the network, we ran four times the experiments to determine from which results who should assess. Table 3 gives an overview of the number of URIs with respectively the attribute "inLOV=false" (TP), "inLOV=true" (FP) and the errors occurred (Null returned, http/proxy or time out reached by the API).

According to Figure 1, `Experiment4` gives stable results with less network errors. Therefore, we stick on this experiment to report our findings and analysis. We found that 227 (43, 48%) are vocabularies in the sense of LOV, because they have at least one property or one class, and the rest, i.e 297 (56, 51%) might have some problems (or event are not vocabularies at all) as they have no classes nor

| | TP(inLOV=false) | FP(inLOV=true) | Errors |
|---|---|---|---|
| Experiment1 | 525 | 44 | 173 |
| Experiment2 | 403 | 26 | 313 |
| Experiment3 | 351 | 28 | 363 |
| Experiment4 | 522 | 44 | 176 |

**Table 3.** Experiments looking for stable results of finding vocabularies in prefix.cc.

properties. Regarding the presence of prefixes names, we found 140 ($61,67\%$) already present in the vocabularies.

In fact, those 227 vocabularies could be easily integrated in LOV. In this list, we found schema Vocabularies like `rdf, rdfs, owl` that are used to built the rest of vocabularies, but are not integrated in LOV catalogue.

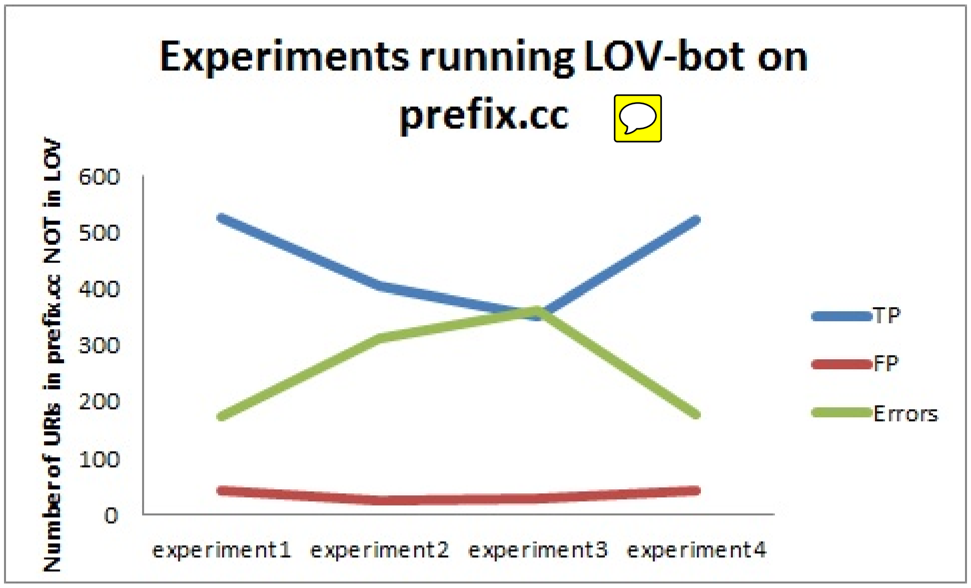

**Fig. 1.** Experiments using the LOV API to check the uris in prefix.cc are vocabularies

**Algorithm 1** finding vocabulary in LOV from prefix.cc algorithm

```
1: Open $notInLOV.jsonfile$ containing the prefix.cc URIs **not** in LOV
2: initialize $item$ as List
3: Initialize $output - result$ as collection of $item$
4: **for** each $pccURI \in notInLOVfile$ **do**
5:     $uri \leftarrow$ value of $pccURI$
6:     $uriv \leftarrow$ construct-valid $uri$
7:     call LOV-Check API with parameter $uriv$
8:     try/catch HTTPError, URLError, IOError, ValueError
9:     **while** no error raised **do**
10:        initialize $item$ to an empty List
11:        append $pccURI, prefix, inLOV, namespace, title, dateIssued, nbClasses, nbProperties$
           in $item$ List
12:        append $item$ to $output - result$
13:    **end while**
14: **end for**
15: **print** $output - result$
```

## Anaysing Errors

From the list of URIs that were not LOV-able vocabularies, we wanted to do more analysis by checking the RDF files using the tool Triple-Checker. Our aim is to be sure if we didn't left out some potential vocabularies, or if they are probably other types of errors, such as parsing errors. Table 4 reveals us 4 categories founded:

– Potential vocabularies (12.20%);
– Those that are not clearly vocabularies e.g: RDF datasets, html pages (6.45%)
– Different errors such as loading the files, parsing them or proxy errors (78.30%)
– Others, which are mainly parsing errors (3,05%), which could be either vocabularies or datasets if fixed.

| **Total URIs** | 295 | 100% |
|---|---|---|
| Vocabularies | 36 | 12.20% |
| RDF data | 02 | 0.67% |
| Loading/404 errors | 182 | 61.69% |
| Proxy errors | 27 | 9.15% |
| Web Pages containers | 9 | 3.05% |
| No triples found | 8 | 2.71% |
| Parsing errors | 9 | 3.05% |
| 50X, 400 errors | 22 | 7.45% |

**Table 4.** Analysis of the URIs with no classes and no properties while using the LOV-Bot API

## 5 Discussion

Prefixes are important because they are part of the DNA of a vocabulary. For this reason, their choices should be taken seriously into consideration as well as defining a policy for the URI of the vocabulary namespace. We recommend the following guidance to the editors and modellers of vocabularies:

– Choose a meaningful prefix, related as much as possible to the vocabulary
– Choose a short combination of characters, preferably between two and eight characters.
– When publishing a vocabulary, use a service like prefix.cc to claim the URI you assign for the vocabulary and define in the metadata properties for the `preferredNamespacePrefix` and the `preferredNamespaceURI`.
– Use stable URIs for vocabularies and use the principles of "Cool URIs don't change"[14] also applied in URIs vocabulary design decisions.
– Suggest your vocabulary when released or published to LOV via the suggestion page.[15]

## 6 Conclusion

In this paper, we have presented a way to manage vocabularies prefixes using two services, LOV and prefix.cc. We have shown that in the process of mapping namespaces with prefixes, some conflicts have to be managed contacting the editors themselves through social networks or mail contacts provided in the Vocabulary specification. The methods and results shown in this work could be easily automated by any other tool using either LOV or prefix data.

One direction of the future work is to have a new strategy on the LOV-BOT API to take into account vocabularies published in n3 and turtle formats. This could leads to change the strategy on how we get the files, by first testing the turtle ones, and on how we get the namespaces. This latter could be more improved using more checking functions on the explicit `vann:preferredNamespace`, similarity algorithm to get the closer namespace given a URI and more statistical approach to compute the classes and properties.

The other direction of this work is to create a small agent in LOV side fetching each day to automate this process described above in live.

## Acknowledgments

This work is partially supported by the project Datalift founded by the French Research Agency (ANR) under grant number ANR-10-CORD-009. The Linked Open Vocabularies initiative is hosted by the Open Knowledge Foundation. The authors are very grateful for the support and help of Richard Cyganiak, author and maintainer of prefix.cc service.

[14] `http://www.w3.org/Provider/Style/URI.html`
[15] `http://lov.okfn.org/dataset/lov/suggest/`

## References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5:1–22, 2009.
2. D. Brickley. Rdf vocabulary description language 1.0: Rdf schema. *http://www.w3. org/tr/rdf-schema/*, 2004.
3. J. Demter, S. Auer, M. Martin, and J. Lehmann. Lodstats – an extensible framework for high-performance dataset analytics. In *Proceedings of the EKAW 2012*. 29
4. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag New York Inc, 2007.
5. A. Ferrara, A. Nikolov, and F. Scharffe. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 7(3):46–76, 2011.
6. T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space: Theory and Technology*, volume 1. Morgan & Claypool Publishers, 2011.
7. W. W. Y. J. Isaac, A. and M. Zeng. Library linked data incubator group: Datasets, value vocabularies, and metadata element sets. *W3C Incubator Group Report. Online: http://www.w3.org/2005/Incubator/lld/XGR-lld-vocabdataset-20111025/*, October 2011. [accessed 25-April-2013].
8. M. B. W. M. Miles, A. and D. Brickley. SKOS core: simple knowledge organisation for the web. *International Conference on Dublin Core and Metadata Applications*, pages 3–4, December 2005.
9. F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, S. Villata, B. Bucher, F. Hamdi, L. Bihanic, G. Képéklian, F. Cotton, J. Euzenat, Z. Fan, P.-Y. Vandenbussche, and B. Vatant. Enabling linked-data publication with the datalift platform. In *26th Conference on Artificial Intelligence (AAAI-12)*, 2012.
10. P.-Y. Vandenbussche and B. Vatant. Metadata recommendations for linked open vocabularies. *http://lov.okfn.org/dataset/lov/suggest/*, 2012. [Online pdf file for recommendations; accessed 25-April-2013].
11. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and Maintaining Links on the Web of Data. In *International Semantic Web Conference (ISWC'09)*, 2009.