Sharing Linked Open Data over Peer-to-Peer Distributed File Systems: The Case of IPFS

Miguel-Angel Sicilia(✉), Salvador Sánchez-Alonso, and Elena García-Barriocanal

Computer Science Department, University of Alcalá, Polytechnic Building, Ctra. Barcelona Km. 33.6, 28871 Alcalá de Henares, Madrid, Spain
{msicilia,salvador.sanchez,elena.garciab}@uah.es

Abstract. Linked Open Data (LOD) is a method of publishing machine-readable open data so that it can be interlinked and become more useful through semantic querying. The decentralized nature of the current LOD cloud relies on location-specific services, which is known to result in problems of availability and broken links. Current approaches to peer-to-peer (P2P) decentralized file systems could be used to support better availability and performance and provide permanent data, while preserving LOD principles. Applications would also benefit from mechanisms that ensure that LOD entities are permanent and immutable, independently of their original publishers. This paper outlines a first prototype design of LOD over the *Interplanetary File System* (IPFS), a P2P system based on Merkel DAGs and a content-addressed block storage model. The fundamental ideas on that implementation are discussed and an example implementation on the early version of IPFS is described, showing the feasibility of such approach and its main differentiating features.

Keywords: IPFS · Linked Open Data · P2P file systems

1 Introduction

Linked Open Data (LOD) is a method of publishing machine-readable structured open data so that it can be interlinked and become more useful and actionable through semantic querying. The current implementation of LOD has resulted in a growing cloud of interlinked *datasets* or "*Web of Data*", that diverse kind of providers (from governments to individuals) expose and give support to, typically using RDF [3]. These providers organize open data exposure around a number of good practices that become progressively adopted at least to a certain extent [18].

LOD is considered by many as an approach to implement open data. In 2010, the Sunlight Foundation collected in ten principles the desirable properties of open (government) data[1], which included accessibility, non-discrimination and permanence. These and other requirements for the public require a robust infrastructure that guarantees sustainability, and it has proven difficult to

[1] http://sunlightfoundation.com/policy/documents/ten-open-data-principles/.

E. Garoufallou et al. (Eds.): MTSR 2016, CCIS 672, pp. 3–14, 2016.
DOI: 10.1007/978-3-319-49157-8_1

achieve it in the current Web of Data, since the LOD cloud is a fully decentralized system that does not feature any built-in redundancy. While open data is in some cases sustained by government policies and programs [21], there is a diversity of providers of diverse size and nature, and availability is not guaranteed for every case.

The current implementation of LOD on top of common Web technology is known to be subject to inherent problems of lack of reliable availability [17,20], which obviously hamper accessibility. This is a natural consequence of the decentralized but location-based approach to publishing in the Web. Many datasets become abandoned or their support discontinued due to a variety of reasons. This is also problematic as machine-readable data is often access via autonomous software programs and not humans, and applications on top of LOD require high availability to avoid those programs to cease function or to be forced to maintain their own proprietary and expensive systems of data caching. Further, many LOD datasets are maintained by organizations that do not have the capacity to sustain the effort of providing the service beyond some project funding. Even worse, many of the datasets are easy targets for different attacks as denial of service [19], as in many cases they do not provide the mechanisms to protect their data against them. Further, the reliance on servers at particular locations may in the future compromise non-discrimination and permanence [10] if some organization decides to revert their open police and restrict access in some way, for example, via throttling. This is also controversial since open data policies [11] are subject to some issues, some of them revolving around the property of the service and its deployment on a particular hosting service.

The above described known issues represent a threat to the success of LOD as an approach for sharing machine-readable data. This situation calls for more robust approaches to data sharing that do not trade the decentralized nature of the LOD cloud. According to its proponents, the *InterPlanetary File System* (IPFS) is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files [2][2]. IPFS and other similar frameworks bring a disruptive approach to the archival of digital resources that is based essentially on independence of location and decentralized storage by networks of untrusted peers (*swarms*). These technologies feature important implications from the technical perspective (as built-in de-duplication), but also from the view of the governance of open data and the current reliance on trusted data providers.

In this paper, we report a first design rationale of deployment mechanisms of LOD graphs on IPFS. We consider the problems of interlinking using content-based storage, versioning and the techniques for bootstraping that kind of alternative LOD version. We also sketch the main envisaged implications of a IPFS based LOD backbone.

The rest of this paper is structured as follows. Section 2 provides some brief background on the technologies involved and the practice of LOD and P2P file systems. Then, Sect. 3 discusses the approach devised to deploy a LOD cloud on

[2] Note this is a reference for the first draft, now superseded by more complete versions.

top of IPFS. Section 4 gives details on how that could be realized with the current state of the tools. Finally, conclusions and outlook are provided in Sect. 5.

2 Background

2.1 Linked Open Data

Linked Open Data (LOD) is a set of conventions to expose open data on the Web, based on adapting the idea of Web links to structured, machine-readable formats as RDF(S) or JSON-LD [14]. A number of tools have been developed to aid in the conversion and exposure of LOD [12] and this has resulted in a diversity of technologies supporting Linked Data.

A number of perceived technical barriers have been identified for the adoption of open data, ranging from the unavailability of a supporting infrastructure to the lack of standards, fragmentation and legacy [9]. These are not different from the problems on reliance on central actors in the Web in general, and they are exacerbated in the case of centralization of providers [4].

Particularly, the fragmentation and heterogeneity of providers results in unavailability of entire datasets [17], non-announced changes in formats or interfaces or simply services being abandoned. This together with lack of performance, scalability or robustness represents a serious risk to application developers and more in general to the accessibility of data. These problems are not inherent to the idea of LOD, but to its current technical and organizational deployment, so that alternatives in the infrastructure layer may bring the required level of robustness both from the perspective of service deployment and of data curation and custody. The read-only and self-mangement nature of P2P networks was identified as a future potential for managing LOD data by Hausenblas and Karnstedt [8], with an understanding that the use case would be different from that of centralized repositories or live look-up systems.

2.2 Peer to Peer Decentralized File Systems

The main concept of P2P technologies is that users contribute part of their computing resources and receive content-centric services in return. Most of the time, P2P services are free of charge, distributed, and there is no concept of operator or central manager. For example, in BitTorrent, each peer contributes parts of the files it locally stores, and is granted download bandwidth by other peers based on how much upload bandwidth it is contributing. Optimizations based on content can also be devised on top of P2P file systems, e.g. by grouping or clustering nodes [15].

The fact that P2P sharing systems rely on a decentralized network of machines with no barriers to entry or exit have resulted in different approaches to incentivizing users while preserving a degree of accessibility.

The InterPlanetary File System (IPFS) is a content-addressable, globally distributed protocol for sharing content that aims to provide a permanent Web.

It combines elements of file-sharing applications such as BitTorrent and version control systems like Git, IPFS can be described as a P2P version controlled file system. It allows for mounting on POSIX file systems also, supporting a seamless interface to applications. The use of P2P technologies for sharing data has already been proposed, as in the case of Biotorrent [13], but more generic frameworks as IPFS would bring better universal management capabilities for it.

3 Proposed Approach

3.1 Publishing Datasets

In the LOD cloud, the common approach to organize data is that of publishing datasets. What one considers as a *dataset* is a matter of convention, but the `VoID` vocabulary[3] provides a useful account if that concept as "a set of RDF triples that are published, maintained or aggregated by a single provider.". The use of RDF is actually an implementation detail, as other formats as JSON-LD are nowadays also commonly used and accepted.

There are two obvious approaches to publish LOD datasets from an archival perspective:

- Publishing entire graphs (whole datasets with many records) as a single object. For example, a georeferenced set of bus stops could be published as a single unit.
- Publishing documents of each independent dereferenceable entity as a IPFS object. For example, in a drug database, the information on each different drug is usually an independently addressable entity in LOD.

The *graph as an object* has the benefit of easing the publishing process for datasets that are immutable (e.g. the data from a project that is completely frozen and will never change) or for the cases in which the dataset curation cycle involves the publishing of snapshots at regular intervals of time that are intended to be identifiable as different "versions". However, if the dataset is subject to small changes that are wanted to go exposed as they come, a unit of lower granularity is desirable. In that second case, the most straightforward decision is that of using the minimum "retrieval and addressing" unit, i.e. those resources that have a unique, dereferenceable unit.

A convention for publishing datasets in IPFS would be requiring a `VoID` object [1] resolved from a human-readable IPNS address. At the time of this writing, you can only publish a single entry per IPFS node (`nodeId`) using IPNS, but this is likely to change in the future. In any case, the node of the publisher should provide a way to access the `VoID` file in a mutable way via IPNS. A convention may be that of publishing it associated to the "root" folder, under the subfolder `/.well-known/void`, imitating the IANA registered well-known

[3] https://www.w3.org/TR/void/.

URI convention[4]. It should be noted that `VoID` descriptions can also link to provenance information following the WC3 PROV proposed specification [16].

Once the VoID description is available, the two approaches mentioned above have some differences and the next section discusses the details.

A related approach to differentiating types of dataset publishing is described in [5], concretely two main approaches to temporal annotation for Linked Data are discussed: document-centric and sentence-centric. The former refers to annotating "whole RDF documents" which may be interpreted as a whole dataset or the RDF graph of a dereferenceable element. The latter is for a more fine-grained temporal annotation. Here we follow an archival model that encompasses versioning instead of temporal annotation, and takes as units the usual data curation units: entire datasets or entities that have are intended for as independent units of information (which is decided by the publisher implicitly by publishing it with a separate URI).

3.2 Documents and Entry Points

In the *graph as an object* approach, the `VoID` file can be simply published together with a compressed version of the dataset in the same folder. This way, the snapshot archived is self-described. The `VoID` description allows for timestamping the snapshot using `dcterms:modified`. Other Dublin Core terms as `dcterms:relation` can be used to point to the previous version, i.e. linking to the IPFS address of the previous snapshot.

In the case of finer granularity, as mentioned above, a common organization for LOD is that of using dereferenceable URIs for each entity of interest in the dataset. For example, http://dbpedia.org/page/Berlin provides the description of the city of Berlin using triples. Content negotiation can be used to get the information in different formats (e.g. different RDF mappings, XML, JSON).

In this case, each of the documents that has a URI could be added to IPFS independently. This would make them permanent and uniquely identifiable independently from the others. It should be noted that the directory and block structure of IPFS is inherently deduplicating files or fragments that are identical, as they are addressed by hashes of the contents.

However, in this case two important conditions must be met in order to guarantee an appropriate accessibility *inside* the addressing system of IPFS (we will call it IPFS linking):

1. All elements should be reachable from entry points available in the description of the full dataset, e.g. as made available in the `VoID` file with `void:rootResource`. This imposes the requirement of knowing some roots of trees or DAGs (i.e. publishing the roots of a forest structure).
2. Links as IPFS addresses (not the original, "normal" links) should be included *whenever* IPFS files available.

[4] http://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml.

The recent[5] IPLD specification[6] formalizes the concept of content-addressed links. The specification in its current form assumes graphs are DAGs, so the discussion about links presented here is applicable.

It should be noted that IPFS can still be used without converting the RDF links to the corresponding IPFS addresses. Such a simpler model is discussed as an example in Sect. 4.2.

The second condition establishes that if an independent IPFS address for the resource exists, it must be used. But this may not be the case in many situations, notably when the links are to other datasets, that may have not (yet) been moved to IPFS. For intra-dataset links, this requires a bootstrapping step in which all the links are changed to the IPFS counterparts. The problem with this is that it entails two issues:

- If the RDF graph is a directed acyclic graph (DAG), the process of adding the documents to IPFS should start from the leaves of the graph and move up to nodes pointing to them. This may require mechanisms for dealing with the graphs *out of core* for large datasets.
- The approach does not work for non-DAG cases, as the IPFS address (the hash) of the document changes with even the smallest change in it.

The latter restriction calls for maintaining the links in a separate index system. While this at first sight conflicts with the usual idea of having links embedded in the documents, it is a model used in the early hypermedia models, and it is also used as a representation in scalable parallel graph architectures as Apache Spark [6].

3.3 Graph Evolution

The approach described so far that involves IPFS linking assumes a static graph, which fits well with the IPFS property of immutable objects. However, many datasets evolve naturally, being DBPedia a prominent example. This is sometimes known as dynamic datasets [5].

A practical approach for this kind of evolution is summarized in the following principles:

1. Deletions are simply impossible due to the permanent nature of IPFS. This fits well with the IPFS approach, and avoids "broken links".
2. New versions of a document entail submitting a new version and having thus a new IPFS address.

The problem with this update is that maintaining a record of the latest changes requires some mechanisms for applications to be aware of them. Several options are available. A possibility is that of using backlinks to previous versions,

[5] Note that this paper was writing before that specification was published in Github, so some of the ideas on this paper may need re-working to be fully IPLD compatible.

[6] https://github.com/ipld/specs/tree/master/ipld.

in the style of `owl:DeprecatedClass` that allows to point to a substitute as `owl:equivalentClass`. The problem is that this needs to be implemented from the new to the previous version (since the previous one is immutable), and has problems of accessibility (how applications could find out the most recent version) and of evolution (the links to the older version should be changed).

These problems point out again to the need to adopt a separation of IPFS links and resources. A simple mechanism could be that of implementing releases or snapshots as index files. This can be done by maintaining an index for each version of the dataset with all the IPFS addresses of the resources for the dataset and the given timestamp, and a separate file with all the links (arcs in the graph). Having complete transformation of the links could be done by specifying links as triples `<ipfs-addr-src, ipfs-addr-dest, URI>` with the source and destination addresses in IPFS, and the normal URI of the link, so that the triple(s) with the original URIs can be re-interpreted by applications by simple substitution. The IPLD format can be adapted to fit under this structure, since in its current form links are embedded in the source document, and there would be a need to adapt to non-embedded links.

A deletion can be achieved by simply making the document inaccessible from entry points. Changes could be marked as substitutions, i.e. pairs of IPFS addresses `<ipfs-addr-old, ipfs-addr-new>`, or simply substituting the old by the new (this complicates versioning back). The IPFS links could be maintained in the same or in a separate but linked file, so that retrieving the full graph entail retrieving the link file and the master index, but the retrieval of the documents with the descriptions can be done separately, delayed or lazily, depending on the needs of the application.

4 Example Implementation

A proof of concept design prototype was built using IPFS version 0.4.3. We have selected two representative cases to illustrate how the above presented approach can be deployed.

4.1 Example Snapshot Dataset

The Ordnance Survey of the UK government makes available their 1:50 000 Scale Gazetteer in the form of LOD[7]. This dataset contains at the time of this writing 258,404 different named places (including farms, antiquities and hills among other types) that are described with only 9 predicates in a total of 2,362,412 triples. Gazetteers are complements to maps that historians use to locate the places, mainly for places that no longer exist and names that are no longer used or whose spelling has significantly altered. Due to the nature of gazetteers, the frequency of update is expected to be low.

[7] http://data.ordnancesurvey.co.uk/datasets/50k-gazetteer.

It should be noted that the Linked Data Cloud diagram[8] does not consider this particular dataset as independent but integrated into the "*Ordnance Survey Linked Data*"[9] dataset, even though the dataset is independently described by their publishers also, in a `VoID` description available in the URI pointed to the `foaf:homepage` of the dataset description in the master file.

The `VoID` description of the dataset is available[10] for all the datasets, but also the fragment related only to the gazetteer. The approach thus to detect changes in the dataset can be that of looking at the `dct:modified` predicate that contains the latest modification date. It is also possible to monitor other elements in the `VoID` description that entail changes as `void:triples` but modification time seems the most sensible alternative.

The approach to publish and update the dataset should then be that of:

- Monitoring changes in the `VoID` description as described above.
- When a change is detected, download the new snapshot pointed to by the `void:dataDump` predicate.
- Include the `VoID` file within the folder of the dataset.
- Add the resulting structure in *path* to IPFS with `ipfs add -r path`.
- The resulting IPFS URI generated from the content of the data and its dataset metadata is retained in a file for enabling location, browsing (e.g. `ipfs ls`) and broader indexing capabilities (as commented below).

This simple approach is sufficient for low-frequency updates of LOD datasets, and it could be used to move that part of the LOD cloud into IPFS provided that the datasets conform to the minimal conventions on using `VoID` that are used for the detection of changes. Of course, this creates full copies of the graphs, which only makes sense in some cases. Also, this simple approach does not convert RDF links to IPFS links.

We have tested that approach using a simple script using `rdflib` in Python, that could be applied to any dataset following similar conventions. Once the data is downloaded, a SPARQL engine can be setup using the same libraries by importing all the `.nt` files downloaded. It should be noted that this approach does not store copies of ontology or vocabulary versions, which should be done separately if the full semantics are to be made permanent with the copy of the data.

In the domain of open data, a problem of P2P systems is that they do not provide built-in capabilities for discovery of datasets, which require building an index of static URIs on top of IPFS itself. At the time of this writing this is still a debated topic, but some prototypes as *Noetic*[11] already exist. In any case, a master file of the snapshots of the dataset indexed using IPNS could be a viable alternative to implement search tools by following common URI standards.

[8] http://lod-cloud.net/.
[9] https://datahub.io/dataset/ordnance-survey-linked-data.
[10] http://data.ordnancesurvey.co.uk/.well-known/void.
[11] https://github.com/doesntgolf/ipfs-search.

4.2 Example URI-Based Dataset

As a second example, we have chosen Europeana, the European digital library that provides an access point to millions of cultural objects (paintings, books, etc.) that have been digitized throughout Europe, gathered from hundreds of individual cultural institutions. This represents an aggregation, that is typically maintained by using the standardized OAI-PMH protocol. The Europeana LOD pilot [7] currently implements a SPARQL endpoint to a regularly updated copy of the Europeana database.

In this case, replicating the entire dataset for a single change would be inefficient, and maintaining snapshots do not appear to be a good candidate, as record updates are determined by the providers, which are independent institutions as museums or archives. The most sensible approach here is that of following the OAI-PMH approach, that works with temporal deltas of the datasets, so that in each harvesting cycle, only the changed or deleted records are collected.

Also, as the collection is diverse and large, it is unlikely that users retrieve it in full but only some particular section of the records using some content criteria. This matches well the idea in P2P networks of a possible content-clustering of data copies. Figure 1 shows the structure of a record, including the entity referring to the real physical object and the description of the view of the provider (`ore:Proxy`) about it. That structure is the logical unit of transfer, so it makes sense to be the unit to be added to IPFS as independent objects.

The prototype implementation used the `Sickle` Python library[12] was used to write a simple OAI-PMH client. The initial harvesting produced the first version of the dataset in this case by iterating the records returned by the `ListRecords` verb request and submitting them to IPFS via `ipfs add`. This generates a list of initial IPFS files which URIs are listed in the initial index file for the dataset with the corresponding OAI-PMH identifiers. The timestamp used in the harvesting

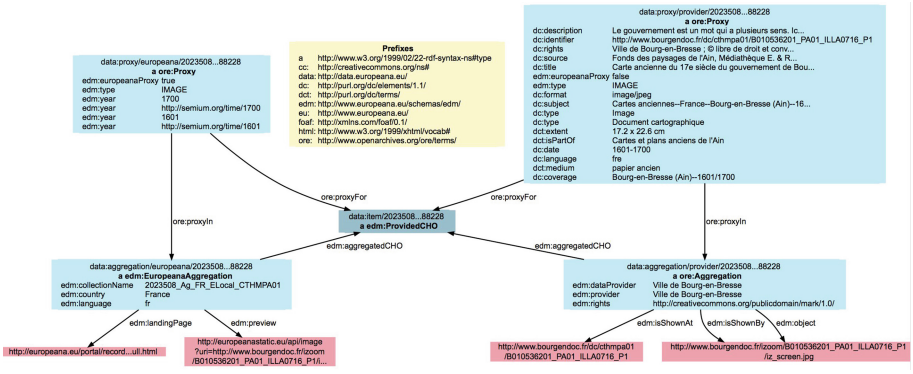

Fig. 1. Example structure of a Europeana CHO

[12] https://pypi.python.org/pypi/Sickle.

request is also included in the index file. The `VoID` file for the dataset is currently not available, but its URI could also be included in the index file.

Subsequent harvesting cycles create a new version of the index file and of the modified records, so that the IPFS addresses of the modified records are changed to the newly created files. A deletion involves simply removing the address of the record from the file. A back link to the previous index is also provided that allows for a retrieval of the history of revisions.

This approach maintains permanent storage of records and the modification granularity is that of a record, which appears a reasonable approach, since this is likely the unit of change at the provider's systems. However, it does not implement IPFS linking, as the URIs used internally are still the original ones.

Regarding linking, the original RDF links are maintained, but when adding or updating the records, they are inspected for the presence of intra-dataset links, i.e. those following the URI design pattern:

http://data.europeana.eu/item/collectionID/itemID

These are then used to generate a separate link file. Both the index and the files are added to IPFS in the same folder. Concretely, the current Web page of Europeana Linked Data states that "When applicable, the Europeana URIs for these [`ProvidedCHO`] objects also link, via `owl:sameAs` statements, to other linked data resources about the same object". This is a typical case for having a separate structure of links for the integration of different provider's metadata views on the same cultural objects.

Such approach could be expanded in the future for between-dataset links, e.g. once other datasets as DBPedia are eventually in IPFS.

5 Conclusions and Outlook

Decentralized P2P file systems as IPFS have the potential to remove technical barriers to the exposure of Linked Open Data (LOD) by providing a infrastructure for the publishing of data that detaches datasets from their institutions when considering sustainability. However, they do not provide explicit support for interoperability and good practices, which remain a concern that can only be solved via agreement and community curation.

In this paper, we have sketched a potential implementation pattern for common cases of sharing LOD datasets that require a limited deployment effort.

There are a number of additional important non-technical implications that should be stressed in adopting IPFS-like technology:

- Datasets become a property of the commons, as there is no way to revoke the publishing of datasets. This is a benefit from the perspective of openness but it would require careful consideration in licensing.
- Availability and performance become a feature of the P2P network, in which nodes decide to host some of the datasets, increasing global and local availability.

- Consequently, there is no upfront cost or investment to publish the dataset, but there is also a lack of control of the quality of service for a particular dataset. In any case, institutions may choose to IPFS-"*pin*" their datasets (or other's dataset that are considered of interest) to ensure local copies.
- P2P systems feature mechanisms to incentivize sharing, and this may result in a lack of neutrality in access that need to be addressed.
- Permanent storage requires versioning and linking among versions, which requires some sort of indexing layer to make dataset versions and derivatives easily findable, and eventually the separation of links and content.
- Semantics and interoperability remain a matter of consensus and adoption of good practices, as demonstrated in the transfer of LOD practices and ideas into the IPFS deployment.

It is still too early to value if the adoption of file systems as IPFS will become widespread and how they would tackle with the problem of distributing the storage responsibilities with some form of incentives to sharing. But in any case, they represent a new playground for experimenting with new ideas and approaches to open data that make it more transparent and independent from their original curators.

References

1. Alexander, K., Hausenblas, M.: Describing linked datasets-on the design and usage of VoID, the vocabulary of interlinked datasets. In: Linked Data on the Web Workshop (LDOW 2009), in Conjunction with 18th International World Wide Web Conference (WWW 2009) (2009)
2. Benet, J.: IPFS-Content Addressed, Versioned, P2P File System. arXiv preprint arXiv:1407.3561 (2014)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. In: Emerging Concepts, Semantic Services, Interoperability and Web Applications, pp. 205–227 (2009)
4. Cabello, F., Franco, M.G., Haché, A.: The social web beyond 'Walled Gardens': interoperability, federation and the case of Lorea/n-1. PsychNology J. **11**(1), 43–65 (2013)
5. Fernández, J. D., Polleres, A., Umbrich, J.: Towards efficient archiving of dynamic linked open data. In: Proceedings of DIACHRON, pp. 34–49 (2015)
6. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graph processing in a distributed dataflow framework. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014), pp. 599–613 (2014)
7. Haslhofer, B., Isaac, A.: Data.europeana.eu: The Europeana Linked Open Data pilot. In International Conference on Dublin Core and Metadata Applications, pp. 94–104 (2011)
8. Hausenblas, M., Karnstedt, M.: Understanding linked open data as a web-scale database. In: 2010 Second International Conference on Advances in Databases Knowledge and Data Applications (DBKDA), pp. 56–61. IEEE (2010)
9. Janssen, M., Charalabidis, Y., Zuiderwijk, A.: Benefits, adoption barriers and myths of open data and open government. Inf. Syst. Manag. **29**(4), 258–268 (2012)

10. Johnson, J.A.: From open data to information justice. Ethics Inf. Technol. **16**(4), 263–274 (2014)
11. Khayyat, M., Bannister, F.: Open data licensing: more than meets the eye. Inf. Polity **20**(4), 231–252 (2015)
12. Konstantinou, N., Spanos, D.E.: Methodologies and software tools. In: Materializing the Web of Linked Data, pp. 51–71. Springer International Publishing, Switzerland (2015)
13. Langille, M.G., Eisen, J.A.: BioTorrents: a file sharing service for scientific data. PLoS One **5**(4), e10071 (2010)
14. Lanthaler, M., Gtl, C.: On using JSON-LD to create evolvable RESTful services. In: Proceedings of the Third International Workshop on RESTful Design, pp. 25–32. ACM (2012)
15. Liu, G., Shen, H., Ward, L.: An efficient and trustworthy P2P and social network integrated file sharing system. IEEE Trans. Comput. **64**(1), 54–70 (2015)
16. Missier, P., Belhajjame, K., Cheney, J.: The W3C PROV family of specifications for modelling provenance metadata. In: Proceedings of the 16th International Conference on Extending Database Technology, pp. 773–776. ACM (2013)
17. Rajabi, E., SanchezAlonso, S., Sicilia, M.A.: Analyzing broken links on the web of data: an experiment with DBpedia. J. Assoc. Inf. Sci. Technol. **65**(8), 1721–1727 (2014)
18. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11964-9_16
19. Wong, A., Liu, V., Caelli, W., Sahama, T.: An architecture for trustworthy open data services. In: Jensen, C.D., Marsh, S., Dimitrakos, T., Murayama, Y. (eds.) Trust Management IX. IFIP, pp. 149–162. Springer International Publishing, Switzerland (2015)
20. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: a survey. Semant. Web **7**(1), 63–93 (2015)
21. Zuiderwijk, A., Janssen, M.: Open data policies, their implementation and impact: a framework for comparison. Govern. Inf. Q. **31**(1), 17–29 (2014)