

# Studying Linked Data Accessibility Healthiness for the Long Tail of the Data Web

Johannes Frey<sup>1,2,\*</sup>, Marvin Hofer<sup>1,3</sup> and Sebastian Hellmann<sup>1</sup>

<sup>1</sup>Knowledge Integration and Linked Data Technologies (KILT/AKSW) / DBpedia Association @ Institute for Applied Informatics, Leipzig, Germany, <https://aksw.org/Groups/KILT>

<sup>2</sup>KMI Competence Center @ Institute for Applied Informatics, Leipzig University, Germany, <https://kmi-leipzig.de>

<sup>3</sup>Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), Dresden/Leipzig, Germany, <https://scads.ai>

## Abstract

In this paper, we explore the accessibility healthiness of Linked Data within the context of the Data Web, focusing on the long tail of data sources. Unlike the traditional web, Linked Data lacks a driving infrastructure to enhance accessibility, leading to negative impacts on data consumers, adoption, and the creation of large-scale infrastructures. We investigate challenges posed by issues such as link rot, unparseable content, downtime, and timeouts that hinder effective access to Linked Data. The study involves a novel Linked Data client that logs debugging information, providing insights into the efficiency and effectiveness of accessing Linked Data. The research also includes discussions on the methods and approach taken, IRI identity mismatch handling, crawling results, and Linked Data parsing statistics. Through extensive analysis of HTTP response status codes and accessibility issues, the paper quantifies common problems but also proposes methods for enhancing Linked Data accessibility in order to retrieve consistent sub-graphs from the Data Web.

## Keywords

Linked Data, Accessibility Issues, Web Crawling, Long Tail, Data Quality

## 1. Introduction

Linked Data was proposed as a way of creating a giant global graph (GGG) of interconnected data on the Web. The idea was that by using shared vocabularies and standard protocols, disparate datasets could be connected to form a single, unified resource that could be navigated and explored in a way similar to the WWW with web browsers.

The evolution of the data itself but also the hosting environment and circumstances lead to accessibility issues such as link rot, unparseable content, downtime, or timeouts when trying to access it. In the traditional web, Google's search engine offers specific browsing entry points based on detailed information needs and enhances accessibility by caching sites and incentivizing proper syntax and standards (e.g. schema.org). Such a driving infrastructure is missing for the Web of Data. Accessibility issues are negatively affecting consumers and adoption, but also hinder the creation of large-scale infrastructures for a better usability of the Data Web, impacting


---


MEPDAW'23: Managing the Evolution and Preservation of the Data Web, November 06–07, 2023, Athens, GR

\*corresponding author

✉ [frey@infai.org](mailto:frey@infai.org) (J. Frey); [hofer@informatik.uni-leipzig.de](mailto:hofer@informatik.uni-leipzig.de) (M. Hofer); [hellmann@infai.org](mailto:hellmann@infai.org) (S. Hellmann)

ORCID [0000-0003-3127-0815](https://orcid.org/0000-0003-3127-0815) (J. Frey); [0000-0003-4667-5743](https://orcid.org/0000-0003-4667-5743) (M. Hofer)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

areas such as data management (e.g. entity indexing, sameAs link clustering) and preservation (archiving, crawling) of Linked Data. While Linked Data crawls - that also explore accessibility aspects - have already been subject to previous work, we exclusively focused on Linked Data according to Tim Berners-Lee's original Design Issues<sup>1</sup>, thus excluding SPARQL endpoints and RDF dataset dumps, but allowed anything – in particular embedded JSON-LD – that follows the rules. Particularly, we collected crawling seed IRIs from several million domains to assess the long tail, which deserves better exploration, especially given the context of establishing aforementioned usability infrastructure. Furthermore, we implemented a novel Linked Data client which extensively logs and stores debugging information as necessary first step to study the efficiency and effectiveness of accessing Linked Data.

## 2. Related Work

Linked Data Crawls have been the subject of several research efforts typically aimed at building large collections of Linked Open Data resources from the Web. In this section, we summarize, to the best of our knowledge, the most notable Linked Data and RDF crawling efforts with public access.

The the most recent iteration of the **Billion Triple Challenge** - the **BTC-2019** dataset [1], involved crawling of over 2.6 million documents from 394 pay-level domains of the Web. The authors retrieved more than 2.1 billion unique quads and 256 million distinct triples. The crawl has been performed using LD-Spider 1.3 in a breadth-first manner based on 442 URLs from DyLDO.

The **Dynamic Linked Data Observatory** (DyLDO) project [2] performs weekly crawls on Linked Open Data since 2012. Based on a fixed seed list (containing 95,737 URIs from 652 domains), it dereferences RDF data in a first round. Subsequently, all discovered IRIs are used to perform another crawl, persisting the retrieved RDF data, HTTP headers, and redirects. The crawler applies breadth-first search and performs 2-5 more rounds, whereas each round, dereferences all unseen IRIs of the next hop based on a frontier list from the previous round. DyLDO was developed to assess the temporal stability of Linked Data resources; the availability and functioning of the Linked Data mechanisms as well as data evolution for particular RDF resources can be analyzed over time. Although the authors used techniques aiming at covering a wide cross-section of domains in the initial seed, the design is focused on completing the crawling given limited time and resource constraints and the seed list is capturing an over 12 years old state of the LOD cloud.

**LOD Laundromat** [3] is a tool that crawls and cleans RDF dumps. The seed was created using a combination of manual and automated methods. The authors added dump URLs using the CKAN API, from e.g. Datahub, but also added several datasets where they knew the location of the dumps. The dump files are retrieved with a custom and fault-tolerant crawler/parser, and VoID triples in the dumps are used to (recursively) discover new datasets and their dump file locations. Moreover, users can submit URLs locating to either an RDF dump or a VoID description of a dataset. However, as of August 2023, the service URL <http://lodlaundromat.org> did not host any data or project related information anymore and the GitHub page states that it

<sup>1</sup><https://web.archive.org/web/20090308030513/http://www.w3.org/DesignIssues/LinkedData.html>

is closed for maintenance since July 2021. Fortunately, a subset of the data (650K RDF documents summing up to 524 GB of compressed HDT [4] data and over 3.3 billion triples) is available in **LOD-a-lot** [5], which has been used as a basis for this work.

**Web Data Commons** [6] is a large-scale project that aims to extract structured data from the Common Crawl. Since 2012, the project released several structured data dumps based on semantic annotations in the crawled HTML files, including Microdata, Microformat, RDFa, as well as Schema.org from embedded JSON-LD [7]. Out of 1.5 billion URLs with semantic markup (of over 3 billion URLs from the October 2022 crawl) that were hosted on 14 million domains, over 19 billion typed entities and 86 billion triples were extracted<sup>2</sup>.

**DBpedia Archivio** [8] is an augmented ontology archive that automatically crawls, discovers, versions, and archives ontologies. In order to discover OWL and SKOS ontologies, it performs follow-your-nose Linked Data on (transitive) dependencies/imports in ontologies from previous iterations of Archivio crawls, but also employs vocabulary usage reports in VoID files, ontology repositories, and user inclusion requests. As such it tries to crawl the Web of Ontologies, a subset of the LOD cloud, and a study in 2022 [9] has shown that it improves the accessibility of the terminological context (property & class IRIs) for 80% of the triples in LOD-a-lot respectively 45% the used terms.

To the best of our knowledge, none of the efforts specifically focus on and evaluate the long tail of Linked Data.

## 3. Methods and Approach

### 3.1. Linked Data Access

Linked Data access follows the principles of web architecture<sup>3</sup>, a multitude of standards, protocols, and rules, including the use of URIs (Uniform Resource Identifiers) and IRIs (Internationalized Resource Identifier) to identify resources, and data models, like RDF (Resource Description Framework), to represent data.

As the main actors of the Linked Data web architecture, the implementation of clients and servers plays an important role in enabling the publication, discovery, and consumption of Linked Data. A Linked Data server provides access to resources through HTTP(S) IRIs. Clients, on the other hand, are responsible for consuming and processing Linked Data from servers and thus retrieve a local sub-graph of the globally accessible Linked Data graph. The Linked Data consumption process involves the following major phases between client and server.

1. IRI dereferencing: The client sends a GET request to the server identified by the HTTP(S) IRI and follows redirects.
2. Representation selection: The server responds with a representation of the resource in a particular format (such as plain RDF serialization formats, RDFa, or JSON-LD).
3. Representation parsing: The client parses the representation (the payload contained in the HTTP body) to extract the RDF or other structured data from it.

---

<sup>2</sup><http://webdatacommons.org/structureddata/2022-12/stats/stats.html>

<sup>3</sup><https://www.w3.org/standards/webarch/>

4. Follow-your-nose: The client might dereference any IRI it finds in the structured data to retrieve additional resources.

### 3.2. Access Mechanisms and Pitfalls

Several mechanisms allow for flexibility in requesting and serving Linked Data and therefore lead to an increased **variety** and **complexity** for both server and clients. Furthermore, implementations and setups may not adhere fully to expectations and specifications, which can result in accessibility **failures** when trying to fetch data from the GGG. A third area, that is highly relevant for access, is retrieval performance and throughput, i.e. how much data can be requested in what time.

- **Redirects and Links.** Servers can use redirects (HTTP code 3xx) as well as `link rel="alternate" . . . \` in the HTTP header and the HTML `<meta>` tag to point to the data document. The correct server configuration is a common pitfall; long(er) redirect chains decrease the performance of servers and clients; loops prevent accessibility.
- **Serialization Variety.** There is no single mandatory format specified in Linked Data, rather a multitude of RDF serialization formats (e.g. Turtle, N-Triples, RDF/XML) and HTML embedded formats such as RDFa or JSON-LD exist that the server could use in response; clients should support them to be able to retrieve all parts of the GGG.
- **Content Negotiation.** As not all servers/clients can deal with all formats, clients may send a prioritized list of formats in the *Accept* header. The server should select a supported format in favor of the client's request. However there are no guarantees on how the server selects the response format, thus requiring the client to be flexible and employing try-and-error heuristics; especially manual proxy/rewrite rule configurations on the server side are a common source of errors.
- **Parsing.** The retrieved serialization can contain erroneous elements, syntactical errors, or a deviant serialization format that was incorrectly reported, requiring fault-tolerant parsing methods (e.g. skipping erroneous parts).
- **Access Limits & Performance.** Servers can be at capacity resulting in timeout errors or they can apply rate limits indicated with HTTP 429 error, clients need to obey the robots.txt and the *Retry-after* Headers (which might not be correctly computed by the server). Overall, low speed of responding to requests (due to server capacity or configured limits) can be a large bottleneck, when accessing the GGG.
- **IRI normalization** URIs/IRIs can be represented and encoded in different forms, but merging interlinked subgraphs of the GGG fetched from several servers requires string equality. IRI normalization can be necessary to make use of the links (see Section 3.3).

To create a practical Linked Data client<sup>4</sup>, we used a bottom-up approach and made changes to the configuration and code based on what we learned from Section 4 & 5.

### 3.3. Handling IRI identity mismatch

Many RDF tools such as graph databases, RDF libraries, and reasoners often require exact matches for IRIs in order to correctly identify them as the same, whereas Linked Data does not

---

<sup>4</sup><https://github.com/dbpedia/ld-crawl>

**Table 1**

URL seed sampling for LOD-a-lot (LAL) and DBpedia External Links (DEL).

	Domains	Distinct URLs	2k+ dom.	2k+ URLs	2k- URLs	$\Sigma$ Sampled URLs
LAL	1,453,373	2,911,686,622	1,492	2,984,000	10,714,576	13,698,576
DEL	4,208,042	33,529,222	1,152	2,304,000	18,107,679	20,411,679

as it has mechanisms like redirects (e.g. for HTTPS upgrades).

While some of the IRI representation variety can be *normalized* by syntactical transformation on the IRI string which is tackled in RFC standards, others need to be further *canonicalized* by dereferencing and matching the IRI in the delivered RDF response.

**IRI Syntax Normalization** is a set of rules that transform an IRI into a normal form that allows equality checks. The RFC<sup>5</sup> includes case normalization, character normalization, percent-encoding normalization, path segment normalization, and scheme-based normalization. However, we argue that this is not sufficient for Linked Data in practice.

**Canonicalization & Consistently Dereferenceable IRIs (CDIRI).** An identity mismatch that needs additional canonicalization occurs when the normalized IRI  $i_n$  of the IRI  $i$  that was supposed to be dereferenced does not occur or is not described as an element of the dereferenced Linked Data document for  $i_n$ . Subsequently, we define an IRI as the consistently dereferenceable (CDIRI) for  $i$  if its normal form  $i_n$  matches the normalized IRI of the resource (most commonly used in subject position) of the data or in simple words - What you request is what you get. In our implementation, we treat the CDIRI as the canonical IRI for the referenced resource or its parts if several CDIRIs are present, which is the case for fragment (#) IRIs. Determining the CDIRIs and using them as a replacement for all IRIs in third-party Linked Data, which provide owl:sameAs or other relationships to  $i$  for a resource  $r$  and thus make the resulting merged local graphs for  $i$  and  $r$  connected (e.g. such that a SPARQL pattern like  $r$  owl:sameAs  $i$ .  $i$  ?p ?o2 would succeed). In the following snippet, we show 2 examples of RDF resources, listing the actual Entity IDs in the RDF and used IRIs in third-party Linked Data (CDIRIs underlined):

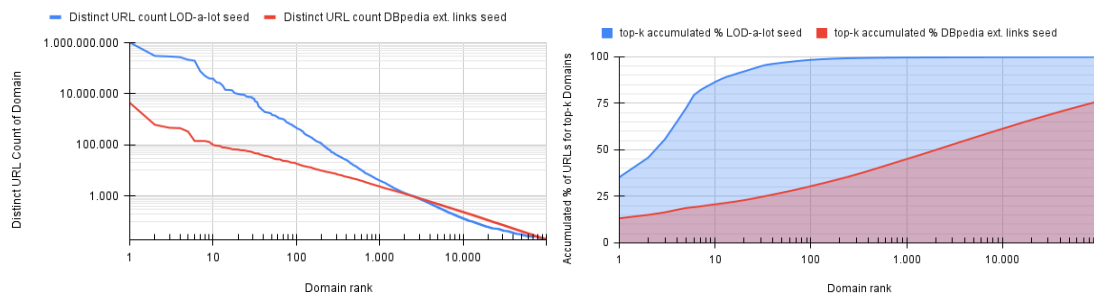
1. <http://dbpedia.org/resource/Björk>, <https://dbpedia.org/resource/Björk> and <https://dbpedia.org/page/Björk> (html view) redirect, resolve, or link to RDF using <http://dbpedia.org/resource/Björk> as entity ID
2. <http://d-nb.info/gnd/1140180746> and <https://d-nb.info/gnd/1140180746> nowadays redirect/resolve to RDF using <https://d-nb.info/gnd/1140180746> as entity ID

Note, that the DNB identifiers were switched to HTTPS several years ago. However, datasets linking to the legacy DNB HTTP identifiers are still widespread. W.r.t. DBpedia, sometimes the HTML page IRI is confused with the entity ID. Working with CDIRIs thus supports cleaning up the connection of sub-graphs provided by different Linked Data providers and increases accessibility.

We developed the Pinguin<sup>6</sup> canonicalization algorithm for Linked Data clients. It collects all intermediate IRIs, *normalizes* them and creates a surjective mapping of IRIs to the CDIRI as well as validates the CDIRI by resolving it again.

<sup>5</sup>Normalization of URIs in RFC3986 Section 6, and for IRIs in RFC3987 Section 5

<sup>6</sup>named after the random resonance of Ping URI to the German word Pinguin



**Figure 1:** Left: Distinct IRI count vs. Domain rank (log-log scale), Right: Aggregated IRI percentage per Domain rank

## 4. Evaluation

### 4.1. IRI Seed Selection

We picked two set of IRIs that cover a huge spectrum of domains to study the long tail.

**Source 1 (LAL) from LOD-a-lot:** For the first source of IRI seeds for crawling, we used LOD-a-lot [5], a compact archive of RDF dumps. It contains more than 28.36 billion triples with 3.21 billion distinct subjects. Following a previous study [9], the data still contains noisy and erroneous data (e.g. IRIs containing prefixes although supposed to be absolute, usage of subject IRIs in property position etc.). As we are interested in IRIs that enable us to retrieve RDF via Linked Data, we filtered LOD-a-lot for all `rdf:type` statements (3,321,354,308 triples) and then collected all distinct IRIs (2,911,686,622) that occur in the subject position of those. This also removed all invalid and not absolute IRIs as confirmed via the Java 11 URL checker.

**Source 2 (DEL) from DBpedia:** As second source, we used the DBpedia external links 2022-09 (DEL) dataset<sup>7</sup>, which contains all hyperlinks to external websites from the Wikitext of articles of 137 Wikipedia versions. The links point to over 33.5 million distinct IRIs (2 invalid IRIs were removed).

Since almost 3 billion URLs would pose a significant challenge (in terms of time, traffic, and storage requirements) for a crawling experiment, we decided to study the distribution of URLs per domain (FQDN - fully qualified domain name), in order to understand whether we can shrink it without limiting the number of domains. The rational behind this is, that our focus is on studying the accessibility of Linked Data for the domains, instead of a full analysis of the (payload) data of the domain.

As can be seen in Figure 1 (left), the distribution of the URL counts per domain for the top-100k domains (the 100,000 domains having the most URLs) follows a power law distribution. This type of statistical distribution is characterized by the pattern that a small number of items occur frequently (called the "head"), while a large number of items occur rarely (denoted as the "long tail"). In our case, the items compare to the domains that occur in the host part in the seed URLs.

The curve of LAL is steeper compared to DEL in log-log-scale, implying that the URL counts decrease more rapidly for LAL as the rank increases, indicating a greater inequality in the

<sup>7</sup><https://databus.dbpedia.org/dbpedia/generic/external-links/2022.09.01>



distribution. Moreover, LAL top-k domains have more URLs per domain than DEL until rank 2,802. In Figure 1 (right) is shown, how much portion of the overall amount of URLs of the datasets are contained in the accumulative counts of the top-k ranks. While for LAL the top-15 domains accumulate over 90 % of all LAL URLs, for DEL they only contain approximately 21 %. In other words, the head of LAL is much shorter with higher IRI counts compared to DEL. In order to reduce the number of URLs, we decided to sample URLs of the head. We decided to limit every domain to 2000 URIs. Subsequently, we used Waterman’s Algorithm R for random sampling to pick 2000 URLs for each of the top-k domains that contain more than 2000 URLs, which splits the domains into a subset of domains that is sampled 2k+ and a subset that is not 2k- (see Table 1). For LAL, this resulted in sampling URLs for the top-1492 domains that cover 99.63 % of the URIs and reducing its number of 2k+ URLs from 2.901 billion to 2.948 million, whereas for DEL the top-1152 domains covering 45.99 % were reduced from 15.422 million to 2.304 million. As a result, 13.699 (LAL) respectively 20.412 million URLs (DEL) were selected for the seeds. The resulting seeds are quite complementary since only 46,810 (0.14 %) of the URLs and 115,313 of the domains (2.08 %) overlap between the two sources.

## 4.2. Crawling Results

From the samples described in Section 4.1, we removed the IRI fragment identifiers and deduplicated the resulting IRIs. We then used our custom crawler implementation that can deal with scheduling millions of domains for parallel crawling.

**Crawling Parameters:** Several parameters can customize the crawling process. We describe the most important parameters of our setup below and refer to our paper repository<sup>8</sup> for transparency and reproducibility. A single request’s *timeout* is set to 10 seconds. The default request delay is 100ms. The crawler obeys Retry-After headers up to a delay of 10s. The system follows a maximum of 10 *redirects*. IRIs of one domain are requested sequentially, and the total timeout for one domain is `max time=2*iriCountForDomain+13` seconds otherwise a `DomainTooSlow` exception will occur. The crawler is configured in this way, because we consider this an appropriate setting performance-wise for an effective GGG, given that network and server performance increased significantly since the inception of Linked Data in 2006. After encountering more than 50 exceptions of either `Java IOException` or request timeout exceptions (signaled by `Java ConnectTimeoutException`, `HttpConnectTimeoutException` or `HttpTimeoutException`) in a sequence for one domain, the crawling of that domain is stopped. Subsequently, all remaining IRIs of that domain are discarded and marked with the `TooManyFailuresInARowException`. Moreover, we only store resource payloads that are smaller than 10 megabytes. Each requests contains the following *Accept* request header (adapted from Apache Jena, but giving priority to line-based and triple formats for fault-tolerant parsing (pruning invalid lines)).

```

1 application/n-triples;q=1, text/turtle;q=0.9, application/n-quads;q=0.9,
2 application/ld+json;q=0.8, application/rdf+xml;q=0.7, application/trig;q=0.7,
3 */*;q=0.5 # covers text/html with embedded JSON-LD

```

Table 2 displays the analyzed log data of four seed partitions. The table is split into two parts. The first part shows the distribution of HTTP response status codes, while the second part displays the accessibility issues that prevented the completion of the HTTP request.

<sup>8</sup><https://purl.org/diamoneonic/papers/lod-crawl-2023>

**Table 2**

HTTP retrieval statistics for the four IRI seed partitions. The table is divided into two parts. The first part displays the distribution of HTTP response status codes. The second part breaks down accessibility issues preventing completion of the HTTP request; JAVA=builtin Java HTTP request Exceptions, CUS=Custom exceptions reported by our accessibility debugging client, e.g., Private IP IRI

ID	Description	Head		Head		Tail		Tail		Total	%
		LAL2k+	%	DEL2k+	%	LAL2k-	%	DEL2k-	%		
—	IRIs (unique w/o #)	2,553k	100	2,300k	100	10,713k	100	17,581k	100	33,147k	100
—	All HTTP Codes	1,498k	58.7	1,912k	83.1	2,276k	21.2	5,024k	28.6	10,710k	32.3
200	Ok	672k	26.3	981k	42.7	896k	8.4	2,843k	16.2	5,393k	16.3
3xx	Redirects	128k	5.0	193k	8.4	163k	1.5	324k	1.8	809k	2.4
4xx	Client Err. Response	657k	25.7	697k	30.3	1,202k	11.2	1,747k	9.9	4,304k	13.0
-403	Forbidden	365k	14.3	182k	7.9	54k	0.5	297k	1.7	898k	2.7
-404	Not Found	259k	10.1	466k	20.2	29k	0.3	1,321k	7.5	2,075k	6.3
-429	Too Many Requests	2k	0.1	6k	0.3	19k	0.2	16k	0.1	43k	0.1
5xx	Server Err. Response	37k	1.4	36k	1.6	14k	0.1	88k	0.5	174k	0.5
Xxx	Other HTTP Codes	77k	3.0	75k	3.3	29k	0.3	197k	1.1	379k	1.1
—	Accessibility Issues	1,055k	41.3	388k	16.9	8,437k	78.8	12,557k	71.4	22,437k	67.7
CUS	UnreliableDomain	514k	48.8	80k	20.5	4,565k	54.1	3,593k	28.6	8,752k	39.0
JAVA	ConnectException	392k	37.2	18k	4.5	2,761k	32.7	3,897k	31.0	7,068k	31.5
JAVA	IOException	105k	9.9	189k	48.7	5k	0.1	2,982k	23.7	3,281k	14.6
JAVA	RequestTimeout	5k	0.5	7k	1.9	1,041k	12.3	1,839k	14.6	2,894k	12.9
CUS	InvalidRedirectIRI	6k	0.5	22k	5.6	22	~0	35k	0.3	62k	0.3
CUS	RedirectLoop	5k	0.5	9k	2.3	3k	~0	11k	0.1	28k	0.1
CUS	MaxRetryAfterTime	1k	0.1	10k	2.5	43	~0	7k	0.1	18k	0.1
CUS	MaxResourceSize	1k	0.1	3k	0.7	58	~0	11k	0.1	16k	0.1
CUS	PrivateIpSkipped	0	0.0	0	0.0	495	~0	19	~0	514	~0
JAVA	Other JAVA Errors	25k	2.4	51k	13.1	61k	0.7	182k	1.5	319k	1.4

**Table 3**

Linked Data parsing statistics. Accepted formats: plain RDF (incl. JSON-LD) and Embedded JSON-LD.

Description	Head		Head		Tail		Tail		Total	%
	LAL2k+	%	DEL2k+	%	LAL2k-	%	DEL2k-	%		
status 200 w. body	672k	26.3	967k	42.0	896k	8.4	2,843k	16.2	5,378k	16.22
- workaround parsable	251k	9.8	189k	8.2	394k	3.7	643k	3.7	1,476k	4.45
- contain CDIRI	205k	8.0	131k	5.7	120k	1.1	569k	3.2	1,025k	3.09
- found unique triples	16,391k	—	14,849k	—	11,128k	—	18,797k	—	42,367k	—
- found unseen IRIs	2,027k	—	640k	—	1,736k	—	2,973k	—	4,299k	—

In terms of HTTP status codes, it is worth noting that status code 200 typically has an average recall rate of only 16%, with the highest value being 42.7% for DEL2k+ and the lowest for LAL2k- at a mere 8.4%. The second most frequently returned status code is 404 Not Found, with an overall percentage of 6.3%. It's also important to mention that 2.4% or 809 thousand IRIs end up causing too many redirects and therefore remain unresolved with a 3xx code. When it comes to overall accessibility, DEL was slightly more accessible than LAL, with 83.1% for the 2k+ seed and 28.6% for the 2k- seed, as opposed to LAL's 58.7% for the 2k+ and 21.2% for the 2k-.



Out of the 33.1 million IRIs that were requested, 67.7% failed due to accessibility issues. The crawler encountered two types of issues preventing the completion of the HTTP request: those related to the remote server and those related to our crawling requirements. The former includes DNS record inaccessibility and resource reachability, while the latter is caused by exceeding manually set thresholds such as payload size (`MaxResourceSize`). The table displays that over 95% of the total exceptions come from the top four accessibility issues. The most common issue is the `UnreliableDomain` exception (ranging from 20 to 54%), which can be triggered by `DomainTooSlow` and `TooManyFailuresInARow` exceptions. The second most common exception is the `ConnectException` (31.5%) which indicates the unreachability of a host due to DNS retrieval or a missing web server under the resolved IP address and port. The `IOException` (14.6%) indicates that the remote server unexpectedly closed an HTTP request, e.g., the response stream. Further, it was found that 12.9% of the issues were related to `RequestTimeouts` that lasted for more than 10 seconds. Two exceptions related to IRI resolution are `InvalidRedirectIRI` and `PrivateIPSkipped`. `InvalidRedirectIRI` denotes that the initial well-formed IRI redirects to an invalid IRI, while `PrivateIPSkipped` indicates skipped requests due to the IRI resolving to a private IP range. Other JAVA Errors, which account for 1.4%, are primarily caused by issues with the JAVA HTTP library, such as HTTP message syntax or SSL problems.

Table 3 analyzes the body of successfully fetched resources (status code 200). 15k responses with code 200, but an empty body, could not be analyzed. 26% of LAL2k+ and 42% of DEL2k+ IRIs had payload in the body. A significantly lower portion but with a similar ratio between LAL and DEL was measured for the tails with 8.4 respectively 16.2%. In a subsequent step, we used our failure-tolerant parser to extract Linked Data from the payload. It was configured to parse all plain RDF formats (including plain JSON-LD, but excluding RDFa) and JSON-LD embedded in HTML. For LAL2k+ and LAL2k- 37 vs. 44% of the content could be parsed making up a only a total of around 10 vs. 4% of the IRIs leading to Linked Data. Since DEL is likely to contain more resources that are not described using Linked Data, it is not surprising that these numbers are also low with 8 vs. 3.7%. As final measure of the accessibility of a Linked Data IRI is row 3. The existence of the CDIRI in the parsed document is necessary to retrieve information about it and further navigate in the GGG (exploiting incoming and outgoing links). For LAL this is the case for 8 vs. 1.1 % of the IRIs in contrast to 5.7 vs. 3.2%.

## 5. Conclusion

To shed light on Linked Data accessibility healthiness of the long tail, we sent requests to IRIs from 5,661,415 distinct domains. We compared accessibility statistics for IRIs from head and tail of LOD-a-lot (containing IRIs from dumped RDF resources) to Wikipedia external links (containing manually curated IRIs, mostly intended for browsers, not necessarily with RDF data) extracted with DBpedia. We discovered, that the head and tail of these Wikipedia links also were significantly impacted by link rot or very slow speeds, with only 43% and 16% respectively returning HTTP 200 status codes. For LAL, we actually expected a much lower and divergent number (26% and 8%) since on the one hand it contains a portion of dumped resources that never were accessible via Linked Data and on the other hand it was published before 2017 - several years earlier than Wikipedia - (based on the assumption that age increases link rot).

Addressing the identified accessibility challenges, an infrastructure for improving resource availability and a refined Linked Data consumption strategies seem potential steps toward fostering a more usable and accessible Linked Data web.

**Acknowledgements:** This work was partially supported by grants from the German Federal Ministry for Economic Affairs and Climate Action (BMWK) to the projects KISS (01MK22001A) and OpenFlaaS (100594042), by the European Research Council for the project ScienceGRAPH (819536) as well as by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI

## References

- [1] J. Herrera, A. Hogan, T. Käfer, BTC-2019: the 2019 billion triple challenge dataset, in: ISWC 2019, volume 11779 of *LNCS*, Springer, 2019, pp. 163–180. doi:10.1007/978-3-030-30796-7\_11.
- [2] T. Käfer, A. Abdelrahman, J. Umbrich, P. O’Byrne, A. Hogan, Observing linked data dynamics, in: *The Semantic Web: Semantics and Big Data*, ESWC 2013, volume 7882 of *LNCS*, Springer, 2013, pp. 213–227. doi:10.1007/978-3-642-38288-8\_15.
- [3] W. Beek, et al., Lod laundromat: A uniform way of publishing other people’s dirty data, in: ISWC, Springer, 2014, pp. 213–228. doi:10.1007/978-3-319-11964-9\_14.
- [4] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, M. Arias, Binary rdf representation for publication and exchange (hdt), in: *Journal of Web Semantics*, volume 19, Elsevier, 2013, pp. 22–41. doi:10.1016/j.websem.2013.01.002.
- [5] W. Beek, J. D. Fernández, R. Verborgh, Lod-a-lot: A single-file enabler for data science, in: *SEMANTICS 2017*, ACM, 2017, pp. 181–184. doi:10.1145/3132218.3132241.
- [6] R. Meusel, P. Petrovski, C. Bizer, The webdatacommons microdata, rdfs and microformat dataset series, in: ISWC 2014, volume 8796 of *LNCS*, Springer, 2014, pp. 277–292. doi:10.1007/978-3-319-11964-9\_18.
- [7] A. Brinkmann, A. Primpeli, C. Bizer, The web data commons schema.org data set series, in: *WWW 2023 Companion*, ACM, 2023, pp. 136–139. doi:10.1145/3543873.3587331.
- [8] J. Frey, D. Streitmatter, F. Götz, S. Hellmann, N. Arndt, DBpedia archivo: A web-scale interface for ontology archiving under consumer-oriented aspects, in: *Semantic Systems*, volume 12378 of *LNCS*, Springer, 2020, pp. 19–35. doi:10.1007/978-3-030-59833-4\_2.
- [9] J. Frey, D. Streitmatter, N. Arndt, S. Hellmann, Reproducibility crisis in the LOD cloud? studying the impact of ontology accessibility and archiving as a counter measure, in: ISWC 2022, volume 13489 of *LNCS*, Springer, 2022, pp. 91–107. doi:10.1007/978-3-031-19433-7\_6.