Heuristics for Fixing Common Errors in Deployed *schema.org* Microdata

Robert Meusel and Heiko Paulheim

University of Mannheim
Research Group Data and Web Science
{robert,heiko}@dwslab.de

Abstract. Being promoted by major search engines such as Google, Yahoo!, Bing, and Yandex, Microdata embedded in web pages, especially using *schema.org*, has become one of the most important markup languages for the Web. However, deployed Microdata is most often not free from errors, which limits its practical use. In this paper, we use the WebDataCommons corpus of Microdata extracted from more than 250 million web pages for a quantitative analysis of common mistakes in Microdata provision. Since it is unrealistic that *data providers* will provide clean and correct data, we discuss a set of heuristics that can be applied on the *data consumer* side to fix many of those mistakes in a post-processing step. We apply those heuristics to provide an improved knowledge base constructed from the raw Microdata extraction.

Keywords: Microdata, schema.org, Data Quality, Knowledge Base Construction

1 Introduction

In the recent years, languages for incorporating structured knowledge into HTML web pages, such as RDFa, Microformats, and Microdata, have been proposed. Out of those, the latter shows the widest adoption [11], in particular due to the *Schema.org* initiative driven by major web search engines such as Google, Bing, Yahoo!, and Yandex.[1]

The main motivation for web site providers to include Microdata is an improved displaying of results by major search engines and by this a improved awareness of their page to the user. Search engines display richer results for web sites described with Microdata.Furthermore, the extraction of a large-scale knowledge base is possible by harvesting data from different sites. One such knowledge base is the Web Data Commons Microdata corpus[2] [11].

In order to fully exploit such capabilities, it is necessary that web site providers adhere to the standards defined by schema.org. For example, a product offer will only appear on an aggregate search site if it uses the correct schema.org classes and properties to annotate the relevant information. Furthermore, a knowledge

[1] `http://schema.org`
[2] `htttp://webdatacommons.org/structureddata/`

base extracted from Microdata will be of higher utility the more strictly the given schema is followed.

In this paper, we analyze common mistakes made in the adoption of schema.org for Microdata. Using the *WebDataCommons* Microdata corpus extracted from the web corpora provided by the Common Crawl Foundation,[3] we perform a quantitative analysis of those mistakes, and we compare the findings to similar analyses carried out on Linked Open Data (LOD). For many of those mistakes, we discuss heuristics to fix them, and apply the fixes to the recent WebDataCommons Microdata corpus. That cleaned corpus contains data that is both syntactically and semantically corrected, and thus represents a more valuable knowledge base. Each heuristic applied is evaluated with respect to its quantitative impact.

The rest of this paper is structured as follows. Section 2 gives an overview of related work. Section 3 shows the quantitative analysis of common mistakes observed in deployed Microdata, and section 4 discusses heuristics for fixing many of those mistakes, as well as the construction of a cleaned up Microdata corpus. We close with a summary and an outlook on future work.

2 Related Work

In [14], the definition of schema.org and its mapping to RDF triples and OWL has been reviewed from a model-theoretic perspective. While that work is rather top-down, starting from the schema definition, we follow a bottom-up approach, making quantitative statements about the actually deployed data – a task named as future work in [14].

Only few works have analyzed the current deployment of RDFa, Microdata, and Microformats in the Web. Mika et al. have presented the first statistics of deployment of the three markup languages in 2011 [12] and 2012 [13], using a non-public web crawl owned by Yahoo!. Bizer et al. [4] present a broader analysis of the current deployment using the public web crawls of the Common Crawl Foundation. They report a strong deployment of markup to describe companies, persons, products, and events, but also note a rather flat usage of properties to describe those items. All those works solely perform an empiric analysis on the current deployment of the different markups and schemas, without a discussion the deviation between the schema definition(s) and the actual usage.

The problem of flatly described items is analyzed in-depth for the class `s:Product`[4] [15]. The authors propose to use regular expressions for extracting features from the title and the description of products marked with Microdata.

A study on validation problems with HTML pages has been done by Chen et al. [5]. They found that only 5% of all web pages are valid according to HTML standards, and analyzed the major problems leading to this invalidity.

For Linked Open Data (LOD), similar works have been carried out [18]. While many of the metrics applied for LOD are rather LOD-specific (such as

[3] `http://commoncrawl.org`

[4] In this paper, we use `s:Foo` as a shorthand notation for `http://schema.org/Foo`.

the presence and correctness of dataset interlinks), some of the typical mistakes apply to both LOD and Microdata, mainly in the categories of *validity* and *consistency.* In the following, we cite some works which perform similar analyses as the one presented in this paper on LOD. Similar to our work for Microdata described in this paper, *LOD Laundromat* project provides cleaned versions of LOD datasets with syntax errors removed [3].

One of the closest works is the work on the *Pedantic Web* [8]. The authors identify four categories of mistakes in Linked Open Data, i.e., *incomplete*, *incoherent*, *hijack*, and *inconsistent.* An updated study on a more recent crawl of LOD has been discussed in [17]. Similar to those papers, *Prolod++* [1], among others, can search for typical modeling problems such as data properties with inconsistent data values (e.g., mixing numbers and dates). The work by Ziawasch et al. [2] even goes one step further. Using the deployment of Linked Open Data, their work aims to check whether properties are attached to the "right level" within the hierarchy or if certain properties should be redefined.

In this paper, we specifically analyze to which extent the *schema definition* of schema.org is followed. Similar works also exist for Linked Open Data, e.g., the *DataBugger* framework, which is based on user-formulated tests run against SPARQL endpoints [9] and examines the adherence of instance data to a schema and additional, user-defined constraints. Similarly, *SWIQA* uses patterns and rules, e.g., for defining legal ranges of literals [6]. On the schema level, tools like *OOPS!* [16] search for common violations of modeling best practices.

3 A Quantitative Analysis of Common Errors in schema.org Microdata

For our analysis, we use the the most recent Microdata corpus[5] from WebDataCommons [11]. That original corpus includes over 8.7 billion triples originating from 463 539 pay-level domains (PLDs), where we focus on the large majority of PLDs which make use of the *schema.org* vocabulary (see Section 3.1).[6]

To avoid misleading results which are artifacts due to the selection strategy of the underlying web crawl (not all PLDs are fully crawled), we mainly report numbers aggregated to PLDs instead of triples. This leads to more representative numbers, assuming that an institution (with one or more web masters) is responsible to maintain the pages of a PLD, and that this institution will always apply the same patterns for markup, i.e., they will also repeat the same set of mistakes. Moreover, many websites are generated from databases, and the markup of the information follows a global algorithm, i.e., values from the same

[5] `http://webdatacommons.org/structureddata/2013-11/`

[6] Only 0.1% of all PLDs deploying RDFa use schema.org, and only 2.4% of all LOD sources [11, 17]. Hence, we restrict ourselves to Microdata, where we see a large-scale adoption of schema.org.

database field are always marked up in the same way for one PLD. Different aggregations are only used for making comparisons to other research works.[7]

For the schema description of schema.org, we use the RDF description of schema.org, using version 1.91.[8]

From the *Pedantic Web* paper [8] (see above), we have selected those mistakes that can also occur in Microdata. However, since we base our analysis on an extracted corpus of structured data, not the original embedding web pages, we have no data for some of the categories reported in their paper, such as syntax errors preventing a correct parsing of the contents. Furthermore, some of the categories, such as the misuse of constructs that exist in OWL, but not schema.org, are not applicable to our use case.

3.1 Usage of Wrong Namespaces and Identification of Relevant PLDs

In this paper, we are primarily interested in *schema.org* Microdata. Hence, we first identify all PLDs from the Microdata corpus which deploy such data by looking at the *namespaces* used. To extract the namespaces from the types within our corpus, we worked with a known namespace list, which includes the most common namespaces as done by Bizer et al. [4].

In our case, the two namespaces for *data-vocabulary.org* and *schema.org* are mostly deployed. For all non-fitting namespaces, we consider the substring until the last non-trailing slash as a namespace.[9]

As a result of this extraction, we could identify over $15K$ different namespaces within the whole Microdata corpus. At a first glance, besides the two major namespaces (*data-vocabulary.org* and *schema.org*), we identified obvious typos of those two major namespaces, and a large number of website specific namespaces.

As proposed by [11], and since we are mostly interested in the most common errors, we filtered out all namespaces occurring solely in one PLD. This results in 361 different namespaces, including the two major namespaces. By manually inspecting this set, we could identify 162 namespaces, used by 398 542 PLDs, which obviously were meant to be *schema.org*, but did not use the correct namespace. 149 of those included the substring `schema.org`. The remaining 13 were well-formed URIs[10] whose protocol and authority is within an edit distance of 1 to `http://schema.org`.

Inspecting the most common errors we found that 102 namespaces include a leading `www.`, 19 use the `https` protocol, 11 have missing slashes within the namespace, and four used a wrong capitalization (e.g., `SChema.org`). Despite this

[7] Although the comparisons should be handled with care, since they might be biased by different crawling strategies underlying the corpora at hand.

[8] `http://schema.rdfs.org/`

[9] Note that this might lead to a larger amount of different namespaces in the case of wrong written namespaces or the use of the schema.org extension mechanism, as defined by `http://schema.org/docs/extension.html`

[10] Based on `http://www.ietf.org/rfc/rfc2396.txt`

large variety, overall only 4 909 (1.23%) from 398 542 pay-level domains deploy a wrong namespace meant to be `http://schema.org`.

In the following sections, we will only use those triples which at least include the substring `schema.org` within the namespace. Those 398 542 PLDs, originating from 217.018.636 different URLs, contain 6.4 billion triples describing 1.4 billion different instances (identified by a class), which corresponds to 86.0% of the complete Microdata corpus.

The problem of wrong namespaces in schema.org Microdata is a subproblem of the *dereferencability issues* in [8], although only for schema elements, not for instances (a schema element with a wrong namespace will not be derefencable, but a non-derefencable one may still have a correct namespace). Still, we can compare it to the analysis of *dereferencability of vocabulary elements* for LOD provided in [17]. According to that paper, 80% of all LOD datasets use at least one schema element which is not de-referenceable, which is a much larger fraction than for schema.org Microdata.[11]

3.2 Usage of Undefined Types

Using the definitions from the schema.org website, we identified 24 227 (6.07%) PLDs which make use of undefined types by simply selecting the *type*-triple for each entity and searching its value in this definition. Table 1 lists the ten most common used schema.org types, which are not defined by the official schema. Inspecting a larger fraction of the list of undefined types manually, we could identify three major different types of errors:

Missing Slashes: As already mentioned in the section above, some data providers did not set the slashes correctly, which results in unknown types when parsing the page (e.g. `http://schema.orgStore` on 6 236 different PLDs).

Capitalization: We orientated our analysis on the formal definition given on the web page of schema.org, including the capitalization. Miscapitalization (e.g. `s:localbusiness`) is also a major source of errors, observed for 1 169 PLDs.

Empty Types: A third mistake, according to our observation, are empty or missing types. We identified 228 PLDs, which did not set a type for at least one item on their page. Furthermore, 3 506 PLDs left the type empty within the markup on the HTML page.

[8] reports that for LOD, 38.8% of all documents use undefined types, as opposed to 5.82% of all documents in our corpus.

3.3 Usage of Undefined Properties

Again using the definitions on the *schema.org* website, we could identify 15 597 (3.92%) PLDs which use at least one undefined property. Table 2 shows the most 20 common used properties which are not defined in schema.org, together with

[11] Even if we pessimistically assume that all other namespaces we observe are wrong.

Table 1. Most common used undefined types within schema.org, ordered by number of pay-level domains.

	Type	# PLDs		Class	# PLDs
1	`http://schema.orgStore`	6 236	6	`http://schema.orgApartmentComplex`	767
2	`http://schema.org`	3 507	7	`http://schema.org/product`	566
3	`http://schema.orgAggregateRating`	1 931	8	`http://schema.orgClothingStore`	404
4	`http://schema.orgPerson`	1 738	9	`http://schema.org/Postaladdress`	368
5	`http://schema.org/localbusiness`	1 169	10	`http://schema.orgPostalAddress`	325

Table 2. Most common used undefined properties by type within schema.org, ordered by number of PLDs.

	Type	Property	#PLDs	Comment
1	`s:ImageObject`	`s:contentURL`	5 904	typo: `s:contentUrl`
2	`s:Article`	`s:type`	2 393	not defined
3	`s:BlogPosting`	`s:postId`	1 574	not defined
4	`s:BlogPosting`	`s:blogId`	1 572	not defined
5	`s:BlogPosting`	`s:image_url`	1 509	not defined
6	`s:LocalBusiness`	`s:URL`	1 365	typo: `s:url`
7	`s:VideoObject`	`s:embedURL`	1 299	typo: `s:embedUrl`
8	`s:SoftwareApplication`	`s:operatingSystems`	529	typo: `s:operatingSystem`
9	`s:VideoObject`	`s:thumbnailURL`	464	close: `s:thumbnail`
10	`s:Offer`	`s:currency`	442	close: `s:priceCurrency`
11	`s:LocalBusiness`	`s:rating`	394	close: `s:aggregatedRating`
12	`s:PostalAddress`	`s:AddressLocality`	387	typo: `s:addressLocality`
13	`s:VideoObject`	`s:contentURL`	382	typo: `s:contentUrl`
14	`s:LocalBusiness`	`s:fax`	302	not defined
15	`s:SoftwareApplication`	`s:SoftwareApplicationCategory`	295	close: `s:applicationCategory`
16	`s:SoftwareApplication`	`s:softwareApplicationCategory`	274	close: `s:applicationCategory`
17	`s:PostalAddress`	`s:postalcode`	255	typo: `s:postalCode`
18	`s:Person`	`s:jobtitle`	201	typo: `s:jobTitle`
19	`s:Review`	`s:itemreviewed`	193	typo: `s:itemReviewed`
20	`s:Product`	`s:identifier`	173	close: `s:productID`

the type they are to be used with. In this list, we can identify different types of errors. One main source of errors are spelling mistakes, as in `s:contentURL`, which is only defined as `s:contentUrl`. This error applies to eight out of the top 20 mistakes. Besides completely not defined properties like `s:postId` and `s:blogId`, we also find we also find properties where there is a close match, e.g., `s:priceCurrency` for `s:priceCurrency`.

The prevalence of undefined properties in LOD has also been investigated in [8], where the authors report that 72.4% of all documents use undefined properties, while in our corpus, there are 9.69% of all documents. In [17], it is reported that 80.75% of all documents use non-dereferencable vocabulary elements, i.e., either undefined properties or types.

3.4 Confusion of ObjectProperties and DatatypeProperties

In our corpus (using only types and properties which are defined by the website) 163 404 PLDs make use of ObjectProperties. Over half of those sites, namely 92 449 (56.58%) use those properties with a literal value at least once. This percentage is large in comparison to LOD where only 8% of all documents use

Table 3. Most common ObjectProperties used with a literal.

	Domain	Property	#PLDs	Actual Domain
1	s:PostalAddress	s:addressCountry	10 249	s:Country
2	s:Product	s:manufacturer	7 933	s:Organization
3	s:Review	s:author	7 807	s:Organization, s:Person
4	s:BlogPosting	s:author	7 089	s:Organization, s:Person
5	s:Article	s:author	5 491	s:Organization, s:Person
6	s:WebPage	s:mainContentOfPage	5 441	s:WebPageElement
7	s:Article	s:creator	4 567	s:Organization, s:Person
8	s:Product	s:brand	4 402	s:Brand, s:Organization
9	s:AutoDealer	s:address	2 437	s:PostalAddress
10	s:Recipe	s:author	2 392	s:Organization, s:Person
11	s:ImageObject	s:thumbnail	2 233	s:ImageObject
12	s:Review	s:itemReviewed	1 564	s:Thing
13	s:Organization	s:address	1 284	s:PostalAddress
14	s:AggregateRating	s:itemReviewed	1 177	s:Thing
15	s:Blog	s:author	1 171	s:Organization, s:Person
16	s:Event	s:location	1 086	s:Place, s:PostalAddress
17	s:WebPage	s:author	991	s:Organization, s:Person
18	s:Offer	s:seller	845	s:Organization, s:Person
19	s:VideoObject	s:thumbnail	818	s:ImageObject
20	s:Book	s:author	619	s:Organization, s:Person

object properties with literal objects [8], as opposed to 24.35% of all documents in our corpus.

Table 3 lists the 20 most commonly misused ObjectProperties by the number of PLDs making use of them.[12] Within this list, literals are mostly used to describe objects of the types `s:Organization`, `s:Person`, and `s:PostalAddress`.

The reverse case is neglectable. While 356 274 PLDs of the corpus use DatatypeProperties, only 810 (0.2%) of those sites use an instance and not a literal for at least one datatype property. This number is low compared to the numbers reported by [8] for LOD, i.e., 2.2% of all documents use datatype properties with non-literal objects, as opposed to 0.56% of the documents in our corpus.

3.5 Datatype Range Violations

For DatatypeProperties, eight different datatypes are defined in schema.org: `Text` (the most general type), `URL` for all kinds of links, `Boolean` for binominal values, `Date`, `DateTime`, and `Time` for temporal values, and `Number` and `Integer` for numeric values. It is notable that, in some cases, more than one datatype is allowed. For example, the property `s:discount` expects either a `Number` or a `Text` as value. For the given values of each datatype property within our corpus, we tried to parse them into one of the defined datatypes (e.g. for the property `s:deathDate`, we tried to parse the literal into a date) using the type guessing code from the *Mannheim Search Join Engine* for parsing web tables [10]. The type guesser uses defensive heuristics, e.g., for URL, we only checked if the literal starts with something like `http`, `www`, `ftp`, or `sftp`, and even includes more possible types for dates than the proposed ISO 8601 standard.

[12] We have excluded all properties which are also used with literals in the examples provided at `http://schema.org`.

Table 4. Most common datatype property values with non-parseable values, sorted by number of PLDs.

	Domain	Property	#PLDs	Expected Datatype
1	s:BlogPosting	s:datePublished	7 890	s:Date
2	s:Event	s:startDate	4 877	s:Date
3	s:Article	s:dateCreated	4 807	s:Date
4	s:Review	s:datePublished	2 691	s:Date
5	s:Event	s:endDate	2 422	s:Date
6	s:Article	s:datePublished	2 247	s:Date
7	s:ImageObject	s:uploadDate	2 097	s:Date
8	s:AggregateRating	s:reviewCount	1 644	s:Number
9	s:Product	s:url	926	s:URL
10	s:NewsArticle	s:datePublished	750	s:Date
11	s:Article	s:dateModified	610	s:Date
12	s:AggregateRating	s:ratingCount	552	s:Number
13	s:VideoObject	s:uploadDate	481	s:Date
14	s:Person	s:url	409	s:URL
15	s:UserComments	s:commentTime	401	s:Date
16	s:Organization	s:url	390	s:URL
17	s:JobPosting	s:datePosted	369	s:Date
18	s:Person	s:birthDate	321	s:Date
19	s:OpeningHoursSpecification	s:opens	295	s:Time
20	s:OpeningHoursSpecification	s:closes	271	s:Time

From the 356 274 PLDs using datatype properties, the parser was not able to parse the literal to one of the defined datatypes on at least one property in 34 324 (9.63%) PLDs. Table 4 shows the top 20 properties with non-parseable literals. Obviously, most difficulties exist for dates: when investigating the data manually, we found various strings that were interpretable as dates for human beings, but which did not follow a known standard. Also for some PLDs, we could not parse the values for `s:reviewCount` properly. Here, not only a number was given in the literal, but also the unit, e.g. ”10 votes“.

A similar analysis was presented in [8] for LOD. Here, the authors examined whether the lexical syntax of literals matched the lexical form. They report that 4.6% of all literals have a mismatch between their declared type and their lexical form, as opposed to 12.06% of all documents in our corpus. Here, the most dominant source of problems were also dates, with prominently 26.6% of all `xsd:dateTime` literals being malformed.

3.6 Property Domain Violations

For each property, *schema.org* defines a domain and a range. It is important to note that the semantics for schema.org are different than for LOD. Schema.org uses `s:domainIncludes` and `s:rangeIncludes` to define *disjunctive*, not *conjunctive* enumerations of domains and ranges as in RDFS and OWL [14]. We assume the enumerations of possible domains and ranges to be *complete*, and count each typed subject or object as a mistake if it has a type which is not contained in the domain or range enumeration (or an `rdfs:subclassOf` thereof), respectively, although disjointness is not explicitly defined in schema.org.

In total, 15 949 PLDs (4.0%) expose domain violations. Table 5 lists the 20 most common domain violations. Column four shows the types the property

Table 5. Most common used defined properties with a domain violation, ordered by number of PLDs.

	Class	Property	#PLDs	Is property of type
1	s:Product	s:price	2 480	s:Offer
2	s:LocalBusiness	s:addressLocality	1 437	s:PostalAddress
3	s:LocalBusiness	s:addressRegion	1 143	s:PostalAddress
4	s:Product	s:availability	1 163	s:Offer
5	s:Product	s:video	1 032	s:CreativeWork
6	s:Article	s:ratingValue	983	s:Rating
7	s:Article	s:ratingCount	943	s:Rating
8	s:WebPage	s:title	868	s:JobPosting
9	s:LocalBusiness	s:streetAddress	766	s:PostalAddress
10	s:Event	s:price	731	s:Offer
11	s:LocalBusiness	s:postalCode	687	s:PostalAddress
12	s:Event	s:telephone	565	s:Person, s:Organization, s:Place
13	s:WebPage	s:location	550	s:PostalAddress
14	s:Place	s:startDate	545	s:Event, s:Role, s:Season, s:Series
15	s:Event	s:email	510	s:Person, s:Organization
16	s:Product	s:category	508	s:Offer
17	s:Place	s:endDate	489	s:Event, s:Role, s:Season, s:Series
18	s:Product	s:priceCurrency	390	s:PostalAddress
19	s:Review	s:ratingValue	344	s:Rating
20	s:Blog	s:breadcrumb	336	s:WebPage

was actually defined for. Inspecting this list, we found that most of the properties which are used actually have `s:PostalAddress`, `s:Offer`, and `s:Rating` as their domain. Looking at the types these properties are used with, we can find a unique direct link between the defined and the used type. In most cases, those types are needed to describe the original item further (e.g. Offer to describe prices, availability of a Product). It seems that the data providers used a "shortcut", without modeling the in-between instance, as defined by the schema. For example, the triple[13]

1. `_:1 s:ratingValue ''5'' .`

is used for an `s:Article` instead of the set of triples

1. `_:1 s:aggregateRating _:2 .`
2. `_:2 a s:AggregateRating .`
3. `_:2 s:ratingValue ''5'' .`

3.7 ObjectProperty Range Violations

We used the schema given at the website to gather a list of all ObjectProperties and their ranges, including all the subtypes and supertypes. This means, e.g. for the the property `s:bloodSupply` expecting an object of type `s:Vessel`, we recursively included all subtypes (e.g. `s:Artery` and `s:Vein`), as well as all supertypes (`s:AnatomicalStructure`, `s:MedicalEntity`, and `s:Thing`) of this object. An instance of any of those types in the object position, respectively, was counted as correctly typed.

[13] Following [7], we use blank nodes for instances extracted from Microdata.

Table 6. Most common type, object property range violation ordered by number of PLDs.

	Domain	Property	#PLDs		Domain	Property	#PLDs
1	s:WebPage	s:mainContentOfPage	6 230	11	s:Review	s:reviewRating	179
2	s:Article	s:aggregateRating	1 696	12	s:JobPosting	s:jobLocation	173
3	s:BlogPosting	s:author	1 460	13	s:JobPosting	s:address	151
4	s:Product	s:offers	405	14	s:Product	s:review	131
5	s:Place	s:address	396	15	s:Recipe	s:reviews	108
6	s:Review	s:aggregateRating	298	16	s:Article	s:author	103
7	s:LocalBusiness	s:address	283	17	s:Dentist	s:address	95
8	s:Product	s:aggregateRating	259	18	s:Movie	s:director	76
9	s:Place	s:geo	257	19	s:Event	s:location	66
10	s:Organization	s:address	219	20	s:SoftwareApplication	s:aggregateRating	63

From the 163 404 PLDs making use of ObjectProperties, 14 089 (8.62%) PLDs violate the defined range of at least one object property on their pages. Table 6 lists the 20 most common range violations. This list does only include those PLDs which use the object properties with an object as range, and not with a literal, as those mistakes are covered in Section 3.4.

The most common *mistake* is made for the property s:mainContentOfPage for the type s:WebPage, expecting an object of type s:WebPageElement. Here webmasters in 92.5% of the cases maintain an object of type s:Blog as value. Semantically, this might make sense, as the Blog is part of the web page, but based on the schema, s:Blog is a subtype of s:CreativeWork and by that no subtype of s:WebPageElement. For the property s:aggregateRating of type s:Article, we found that the major reason for the range violation results from undefined types, resulting from spelling mistakes, e.g. s:aggregatedrating.

In [8], a similar analysis has been carried out for LOD, using reasoning to find inconsistencies between a type assigned to an instance, and the expected type according to the domain/range of its properties. On average, 2.4% of all LOD documents are reported to show one such inconsistency, as opposed to 3.2% of the document in our corpus.

3.8 Hybrid Properties

Last, we have a look at properties which are defined as DatatypeProperties as well as ObjectProperties in schema.org. In comparison to LOD, where this phenomenon is only rarely applied [8], 24 such properties exist in schema.org, e.g. s:category, s:citation, s:defaultValue, s:image, s:option, and s:query to name just a few.[14] While those are not a mistake w.r.t. schema.org, they lead to an unclean knowledge base when applying RDFS/OWL semantics.

Table 7 lists all of the hybrid properties which are deployed within our corpus with the number of PLDs making use of them at least once. From the 24 available hybrid properties, only 10 are present within our corpus. The most common used property is s:image, whose range can be an URL or an

[14] The complete list of properties can be found at http://webdatacommons.org/structureddata/2013-11/stats/fixing_common_errors.html.

Table 7. List of all deployed hybrid properties, ordered by the number of pay-level domains using them at least once.

	Property	#PLDs		Property	#PLDs
1	s:image	133 819	6	s:citation	24
2	s:logo	10 929	7	s:license	13
3	s:model	2 231	8	s:eligibleRegion	12
4	s:category	825	9	s:toLocation	1
5	s:screenshot	305	10	s:fromLocation	1

Table 8. Distribution of deployed object or datatypes for hybrid properties by percentage of PLDs making use of those values. Most outstanding values are marked bold. The table lists only properties used by more than one PLD.

Property	# Different PLDs	Object	Text	URL	Number	Date
s:image	133 819	0.08%	15.22%	**84.68%**	0.01%	0.00%
s:logo	10 929	1.59%	3.41%	**95.00%**	0.00%	0.00%
s:model	2 231	0.14%	**76.82%**	0.82%	14.27%	7.94%
s:category	825	0.12%	**98.32%**	1.20%	0.24%	0.12%
s:screenshot	305	3.25%	5.84%	**90.91%**	0.00%	0.00%
s:citation	24	14.29%	50.00%	32.14%	3.57%	0.00%
s:license	13	0.00%	**76.92%**	23.08%	0.00%	0.00%
s:eligibleRegion	12	0.00%	**100.00%**	0.00%	0.00%	0.00%

`s:ImageObject`, the same holds for `s:logo`. The third most common used hybrid property is `s:model`, where the schema expects a textual description of an item of type `s:ProductModel`. A still broadly used property is `s:category`. The schema allows a textual description, as well as a `s:Thing` and more specific a `s:PhysicalActivityCategory`.

For those 10 properties, we again used our datatype guesser (see section 3.5) to find out what kind of value is mostly used for those properties. Table 4 lists for each of the properties the percentage of value types used by PLDs for this property. Whenever it was not possible to find a more specific datatype, the datatype `s:Text` was guessed. The table reveals that most of the cases, objects are not the dominantly deployed value types for those properties. Among the properties, `s:citation` stands out. Here, we cannot define the major used value type, as beside `s:CreativeWork` and `s:Text` (as defined by the schema), 32% of the PLDs use a URL as value.

4 Heuristics for Fixing Deployed schema.org Microdata

Since we cannot rely that data *providers* will fix their Microdata, we follow the approach of repairing the data on the *consumer* side. In the following, we introduce a set of simple heuristics for fixing schema.org Microdata, and quantify their coverage. With those heuristics, many of the mistakes discussed above can be fixed rather easily.

4.1 Identifying and Fixing Wrong Namespaces

According to our observation from the previous section, we identified a set of heuristics to fix the most common namespace mistakes:

1. Removal of the leading `www.` before `schema.org`
2. Replacement of `https://` by `http://`
3. Conversion of the whole domain name to lower case
4. Removal of any additional sequence between `http://` and `schema.org`
5. Addition of an extra slash after `schema.org`, if none is present.

Using these rules in the given order, we are able to fix 147 out of 148 of wrongly spelled `schema.org` namespaces. The remaining namespace had a duplication of the top-level domain `.org` and could not be fixed by these heuristics.

4.2 Handling Undefined Types and Properties

Apart from mistakes resulting from errors within the namespace and missing slashes, our analysis in Section 3.2 and 3.3 has revealed that a large number of undefined types and properties are caused by spelling errors, in particular wrong capitalization (e.g. `s:contentURL` and `s:jobtitle`). Thus, whenever parsing Microdata entities from web pages, we suggest to not take capitalization into account, and replace each schema element with the properly capitalized version. This approach has also been proposed for consuming LOD in [8].

Applying this heuristic (together with the fixing of namespaces as above) to the undefined/unknown types, it is possible to replace them by correct types for 17 192 (71.0%) of all PLDs using undefined types. Likewise, we can replace undefined properties on 10 281 (65.92%) of the PLDs exposing that problem. However, we can observe a long tail distribution here, i.e., the remaining 29.0% (34.08%) PLDs account for 73.89% (91.82%) of all undefined types (properties, resp.). Those long-tail errors are typically hard-to-detect typos or types and properties that have been made up freely.

4.3 Handling ObjectProperties with a Literal Value

As shown in section 3.4, the main objects which are modeled by the webmasters as literals are `s:Organization`, `s:Person`, and `s:PostalAddress`. Thus, we randomly inspected 715 such property value for the properties `s:author`, `s:creator`, and `s:address`, to get a better understanding. From this analysis, we saw that the majority of literals for `s:Person` and `s:Organization` are person and organization names or URLs, while `s:PostalAddress` is usually represented by a textual representation of the address.

From this observation, we derive the following strategy for fixing literal valued ObjectProperties: Given a triple

1. `_:1 s:op l .`,

where `s:op` is an ObjectProperty, and `l` is a literal, replace the triple by

1. `_:1 s:op _:2 .`
2. `_:2 a s:t .`
3. `_:2 (s:name|s:url) l .`

Here, `s:t` is the range of `s:op`, or the least abstract common supertype of all ranges, if there are more than one. If `l` is a valid URL, then it is set as the `s:url` of the newly created instance, otherwise, it is used as its `s:name`.[15]

With this heuristic, we are able to replace *all* misused `ObjectProperties` on 92 449 PLDs with a semantically correct set of triples. Note that using this heuristic might change the overall distribution of types within the corpus, as it will create a larger number of new entities (e.g., of type `s:PostalAddress`). For example, mapping all `s:address` literal values to a new `s:PostalAddress` would create around 14 million new entities of this type, which would be an increase of 11%. Inspecting this shift more closely will be subject to future work.

4.4 Handling Property Domain Violations

As discussed in section 3.3, properties used on objects that they are not defined on are often caused by "shortcuts" taken by the data provider. Picking up the example above, the data provider used the triple

```
1. _:1 s:ratingValue ‘‘5’’ .
```

instead of the set of triples

```
1. _:1 s:aggregateRating _:2 .
2. _:2 a s:AggregateRating .
3. _:2 s:ratingValue ‘‘5’’ .
```

where `_:1` is of type `s:Article`. In order to expand the wrong triple to the correct set of triples, we need to guess what the data provider meant. To that end, we use the following approach: Given two triples

```
1. foo:x s:r foo:y .
2. foo:x a s:t
```

where `s:t` is *not* the domain of `s:r`, we try to find a relation R and a type T within schema.org such that one of the following two patterns is fulfilled:

```
1. R s:domainIncludes s:t .
2. R s:rangeIncludes T .
3. s:r s:domainIncludes T .
```

```
1. R s:rangeIncludes s:t .
2. R s:domainIncludes T .
3. s:r s:domainIncludes T .
```

If there is *one unique* solution for *only one of the two* pattern, we replace the erroneous triple with the solution we found. In a second step, we unify all newly created entities of one type into one entity. Thus, given that in the above example, there was also a `s:ratingCount` defined, we would end up with only instance of *s:AggregateRating* with both the `s:ratingValue` and the `s:ratingCount` properties from the original `s:Article`.

With that heuristic, we could replace 1 098 out of 3 767 properties used with types they are not defined for, which corresponds to 5 011 (31.42%) of all PLDs. In 986 cases, no solution could be found for any of the two patterns; in the remaining 1 683 cases, the solution found was not unique.

[15] Note that `s:name` is more generic than, e.g., the name of a person. It is comparable to `rdfs:label` in RDF.

5 Conclusion and Outlook

In this paper, we have identified the most common mistakes made by providers of *schema.org* Microdata. Beside more obvious mistakes as spellings errors within namespaces, types or property names, we have identified various confusions within the usage of values of ObjectProperties and DatatypeProperties, and the violation of domain and range constraints defined for schema.org. Additionally, we have investigated the parseability of values, e.g., numbers or dates.

For the issues identified, we have performed a quantitative analysis and compared the numbers to similar analyses carried out on Linked Open Data. The comparison shows that Microdata is cleaner than LOD w.r.t. simple errors such as the usage of undefined types or properties, while schema conformance (such as respecting domain/range restrictions) is higher for LOD.

One main finding is that the majority of information marked-up using Microdata with *schema.org* can be parsed following the recommended schema. We have proposed a set of simple heuristics that can be applied by data consumers to fix a large fraction of wrong markup in a post-processing step. With those heuristics, we were able to curate an improved, cleaned up version of the WebDataCommons Microdata corpus, which corrects many of the syntactic and semantic errors made on the data providers' side. This new corpus is a higher quality knowledge base, derived from Microdata deployed on the web, and fixing data provided at tens of thousands of PLDs.[16]

Many of our heuristics are still simple, and there is a room for improvement. For example, we are currently not trying to guess matching properties for misspelled ones beyond capitalization errors. Furthermore, our method for creating new objects for literal-valued ObjectProperties is rather simple. In particular for complex objects, such as addresses, it could be strongly improved by training extractors that decompose the given literal into a street, a city, ZIP code, etc. Furthermore, our heuristic for domain violation so far only works if there is a *unique* solution, but a more relaxed version looking for *likely* solutions (e.g., patterns that are more commonly deployed than others) could fix even more mistakes. Similar solutions could be applied for fixing ObjectProperty range violations, which are currently not addressed by our approach.

Another interesting observation we made was that some classes and properties – such as `s:Game` – were already widely used in the corpus *before* they became a standard. With our methods, we can identify such widely used cases and provide quantitative evidence to discussions on missing classes and properties in the data schema.

While in this paper, we have taken a *synchronic* approach, looking only at the state of the data deployment at the current time, we aim at extending our analysis with a *diachronic* perspective, looking at the changes over time. This would reveal insights data quality change over time, as well as on the pace at which changes in the data schema (such as deprecations) are adopted.

[16] The corpus is available for download at `http://webdatacommons.org/structureddata/2013-11/stats/fixing_common_errors.html`

References

1. Abedjan, Z., Gruetze, T., Jentzsch, A., Naumann, F.: Profiling and mining rdf data with prolod++. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on. pp. 1198–1201. IEEE (2014)
2. Abedjan, Z., Lorey, J., Naumann, F.: Reconciling ontologies and the web of data. In: Proceedings of the 21st International Conference on Information and Knowledge Management (CIKM). pp. 1532–1536. Maui, Hawaii, USA (2012)
3. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: Lod laundromat: A uniform way of publishing other peoples dirty data. In: ISWC (2014)
4. Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., Völker, J.: Deployment of rdfa, microdata, and microformats on the web – a quantitative analysis. In: ISWC (2013)
5. Chen, S., Hong, D., Shen, V.: An experimental study on validation problems with existing html webpages. In: Proceedings of the 2005 International Conference on Internet Computing, ICOMP'05 (2005)
6. Fürber, C., Hepp, M.: Swiqa–a semantic web information quality assessment framework. In: ECIS (2011)
7. Hickson, I., Kellogg, G., Tennison, J., Herman, I.: Microdata to rdf – second edition (2014), http://www.w3.org/TR/microdata-rdf/
8. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: Linked Data on the Web (2010)
9. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd international conference on World Wide Web. pp. 747–758 (2014)
10. Lehmberg, O., Ritze, D., Ristoski, P., Eckert, K., Paulheim, H., Bizer, C.: Extending tables with data from over a million websites. In: Semantic Web Challenge (2014)
11. Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, rdfa and microformat dataset series. In: ISWC (2014)
12. Mika, P.: Microformats and RDFa deployment across the Web . http://tripletalk.wordpress.com/2011/01/25/rdfa-deployment-across-the-web/ (2011)
13. Mika, P., Potter, T.: Metadata statistics for a large web corpus. In: LDOW 2012: Linked Data on the Web. CEUR Workshop Proceedings, Vol. 937, CEUR-ws.org (2012), http://ceur-ws.org/Vol-937/
14. Patel-Schneider, P.F.: Analyzing Schema.org (2014)
15. Petrovski, P., Bryl, V., Bizer, C.: Integrating product data from websites offering microdata markup. In: 4th Workshop on Data Extraction and Object Search (DEOS2014) @ WWW (2014)
16. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. International Journal on Semantic Web and Information Systems (IJSWIS) 10(2), 7–34 (2014)
17. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: ISWC (2014)
18. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S., Hitzler, P.: Quality assessment methodologies for linked open data. Submitted to Semantic Web Journal (2013)