ABSTRACT
We collect evidence to answer the following question: Is the quality of the XML documents found on the Web sufficient to apply XML technology like XQuery, XPath and XSLT? XML collections from the Web have been previously studied statistically, but no detailed information about the quality of the XML documents on the Web is available to date. We address this shortcoming in this study. We gathered 180K XML documents from the Web. Their quality is surprisingly good; 85.4% is well-formed and 99.5% of all specified encodings is correct. Validity needs serious attention. Only 25% of all files contain a reference to a DTD or XSD, of which just one third is actually valid. Well-formedness errors and validity errors are studied in detail. Our study is well documented, easily repeatable and all data is publicly available [48][21]. This paves the way for a periodic quality assessment of the XML Web.

Keywords
Standardization, XML, XML Web, Schemas, Data Quality.

1 INTRODUCTION
Querying and extracting data from the Web has been an ongoing research issue since the birth of the Web. Querying the Web invariably involves extracting data, processing it and loading it into a representation of a model. Models range from inverted indexes mapping words and phrases to URLs as used by search engines to relational structures such as used by DBpedia [9] and Yago [43]. Semantic Web applications in particular need to extract high quality facts and rules from often low quality webpages [42].

Extracting data is a difficult task and in the nineties there was a large number of proposals for query languages, see [15] for an extensive overview. A common thread in these languages is their declarative nature. They typically combine unstructured content-based keyword search queries with structure-based queries similar to those found in a database system. Later generation languages add the possibility to query the inner structure of documents and the link structure between documents, and to create new complex structures as the result of a query [15]. Thus these languages can both extract and transform information. Today, we see these two tasks back in the three most important XML technologies: XPath is used for extracting information that is further transformed either by XSLT or XQuery. All three languages are W3C recommendations.

Much of today’s Web is written in XML, usually in a well-designed language with its own schema. Prime examples are XHTML, RSS-ATOM, RDF(S) and various OWL dialects [20]. Below we will argue why extracting information from XML data should be done using XML technologies rather than general purpose solutions. We first illustrate this with an example.

The example use case is to extract infoboxes from Wikipedia articles, the core extraction task behind DBpedia. Listing 1 contains an XQuery that extracts attribute-value pairs from an infobox. The first line defines, using an XPath expression, the XML element that is the infobox. This is a table element. The second and third line output an XML element infobox with a source attribute containing the name of the Wikipedia page, extracted by means of an XPath expression. The for-statement then loops through the rows (the tr-elements) of the table. Using a filter expression we consider only rows which have both an attribute (stored in a th-element) and a value (stored in a td-element). Finally we extract the name and value of the attribute using XPath expressions and output them. Listing 2 contains a cleaned up version of the output of this XQuery when applied to the page of Alan Turing. The data cleaning can also be done elegantly in XPath 2.0. The elegant division of work between XPath, which extracts pieces of data, and XQuery, which assembles them again, makes for simple, easily maintainable and generalizable code.

```
let $infobox := //table[contains(@class,'infobox')]
return <infobox source='{$av-pair//@href}'>
  for $sav-pair in $infobox//tr/th[following-sibling::td]
    return <item attribute='{$sav-pair/th'} value='{$sav-pair/td}' />
</infobox>
```

Listing 1. XQuery that extracts attribute-value pairs from Wikipedia infoboxes.

1 This paper is an extension of a 6 page abstract which appeared in Proceedings of CIKM 2011 [22].
In this study we look at the prospects of using XML technology for information extraction and integration tasks on XML data found on the World Wide Web. Without becoming specific we refer to the multitude of these tasks as Extract-Transfer-Load (ETL) tasks [38]. Examples of ETL subtasks are data harvesting, text extraction, structure extraction, text mining [26], data de-duplication [14], data exchange (from one schema to another) and data publishing (from one format to another, e.g. XML to RDF or relational).

Apart from the actual harvesting of data from the Web, all of the above tasks can be expressed in the three XML query languages: XPath 2.0, XSLT 2.0 and XQuery 1.0. Not only can these tasks be expressed in these languages, when the input is XML it is desirable to do so for a number of reasons. XSLT and XQuery programs are largely declarative. The semantics of the languages is clear and well defined. The languages are vendor and software independent, developed and maintained by a committed community and W3C standards. The immense success of SQL [2] shows the great software engineering benefits of working with such programming languages. Maintainability of code is crucial for ETL tasks as they are typically applied in a changing environment not under control of the developers of the ETL code. The LiXTo [19] suite of Web-extraction tools is built on these principles. A recent addition to the LiXto tools is OXPath [17], an extension of XPath which allows declarative extraction of the deep Web.

Whether it is feasible to use XML technology for ETL tasks depends on many factors. This is out of the scope of this study. Here we only look whether it is possible. That is, is the quality of the XML documents found on the Web sufficient to apply XML technology?

Another reason to study the XML Web is the new XQIB, XQuery in the Browser, initiative [53]. XQIB aims at improving the programmability of Web browsers by enabling the execution of XQuery programs in the browser to navigate and manipulate XML. XQIB can be useful in the development of AJAX-style applications and is an alternative to JavaScript [16]. Obviously it needs XML of good quality.

Previous studies on HTML showed that the vast majority of HTML documents (around 95%) on the Web did not comply with the standards set by the World Wide Web Consortium [11][34][36]. For XML, studies that measure basic quality indicators (like being syntactically correct) on arbitrary XML data from the Web have not been performed yet. There are several empirical studies on XML but they either use data from repositories or have very small samples that only contain well-formed XML (Cf. Section 2).

We were unhappy with this omission and frustrated by our own efforts of using XML tools for a large data integration project which aims to create an XML repository of parliamentary proceedings from a number of European countries from data available on the Web [31]. We found that most XML data from these sources contain several types of errors resulting in an immense effort to clean up the XML data before XML tools can be used. The quality of XML from the same sources changes over time as well, which makes automation to keep the data collection up to date a huge challenge. To study the quality of XML and XML tools we set ourselves the following research goal:

Create a corpus of XML documents and accompanying schemas that is representative of the Web, evaluate which part is ready to be processed with XML tools, and evaluate the prospects of automatic error correction for the other part. In addition, process, document and store the corpus in such a manner that our study can easily (e.g., yearly) be repeated.

The paper describes the created collection (Section 3), and the evaluation of its quality (Section 4). We also created a corpus of schemas in the three most used schema languages for XML and evaluated their inter-convertibility (Section 4.4). The remainder of this introduction consists of our operationalization of XML-quality and an overview of the main results. Related work is presented in Section 2.

1.1 Basic quality requirements

One can only apply XML tools to XML files if they satisfy a few basic, but important, properties. As XML is a self-describing format, these properties all state that files should not lie about themselves concerning some aspect X. We looked at three aspects: a file should not lie about its encoding, it should not state that it is XML when it is not, and it should not lie about its validity with respect to a schema. More precisely,

1. The document should be encoded using a single encoding that is stated in the document.
2. The document should be well-formed XML.
3. If the file refers to a schema, it should be useful and truthful. This means that:
   a. The URI identifying the schema should be resolvable. Also all included schema files should be resolvable recursively;
   b. All these schemas are syntactically correct, and
   c. The file is valid with respect to the schema(s).

These quality criterions have been used in previous studies. Related studies are described in Section 2.

1.2 Overview of the study

This study concentrates on the three most used schema languages: DTD [13], XML Schema [50] and Relax NG [39]. To avoid confusion we use the common abbreviation XSD to refer to XML Schema.

Crawling Yahoo and Google for XML documents, we were able to collect almost 180K unique XML files from the Web from almost 100K websites with a total size of 40GB. We now summarize the main results. Our first result states that encodings do not pose a real problem as 99.5% had a correctly specified encoding. The other results are neatly summarized in Figure 1.
Valid documents are rare on the Web. Just over 10% of the well-formed documents are also valid. Figure 2 and Figure 3 present the causes for non-validation of those files that reference downloadable schemas but could not be validated.

Figure 1. Summary of the Quality of the XML Web.
If we pick a random XML document from our collection there is a 14.6% chance that it is not well-formed (the complement of the green circle in Figure 1). This is much better than could be expected from earlier studies on HTML. Almost a quarter of the documents reference a DTD or XSD. Ill-formed XML is more common in files referencing a schema than in general (32% vs 14.6%).

Figure 2. Distribution of causes for non-validation: DTD.

Figure 3. Distribution of causes for non-validation: XSD.

The differences between DTD and XSD are remarkable. 73.5% of files referencing a DTD are not valid just because the XML files are not even well-formed. For XSDs this cause of error is negligible. In contrast, 31% of all XSD validity errors are due to a schema that is not syntactically correct. This happens in only 4% of the DTDs.

1.3 Main contributions
Our main contribution is an up-to-date and reliable estimate of the quality of the state of the XML Web in 2010. Our second contribution consists of a catalogue of the type of errors that compromise the quality of the XML Web. As they are Pareto distributed we believe this can be used to guide research in automatic error repairing. Our third contribution is the collected data itself. All data is made publicly available in a uniform format [48]. All referenced schema files are locally available, and all references in XML and schema files point to these local copies. Information about headers, encodings, and errors of each XML file is stored in a relational database that is also available. All scripts and settings for crawling and analyzing the collection are available and well-documented. This makes our study easily repeatable. We hope that this is a start of a longitudinal XML collection. Of course, we also hope to see a steady improvement of the quality of the XML Web.

2 BACKGROUND
Data quality is a research field established by Madnick and Wang and matured into a field with its own ACM journal [27]. Within the categorization of data quality research described in [27], our study is concerned with assessment and uses an empirical method. Ultimately, we study whether XML data from the Web is ‘fit for use’ by XML technologies. The idea to measure data quality as data being ‘fit for use’ by data consumers goes back to [51]. The latter paper describes a hierarchy of 20 quality dimensions grouped into 4 categories. All of these dimensions are semantic in nature, and thus our study does not fit in very well. This may be attributed to the different setting of our research. Traditional data quality research considers integration of a handful of typically tightly controlled relational databases. We look at coupling thousands of loosely controlled sources consisting of semi-structured data. Ownership and responsibility of data is not easily traced back to individuals or organizations. Often these are group efforts. Semi-structured data and data not residing at clearly defined entities are mentioned as future data research areas in [27].

A large number of descriptive studies on XML have been conducted. There are three main themes identifiable in the literature, which will be discussed accordingly: studies on XML collections; studies on the quality of the HTML Web; and studies on schema languages for XML.

2.1 Studies on XML collections
Studies on XML document collections mainly differ in sample data [45]. A study has been done on 200,000 publicly available XML documents from the Xyleme repository [3]. This collection contains only well-formed XML documents. A second study used a total of 16,534 XML documents, accounting for a total size of 20 Gigabytes, from a number of XML collections including DocBook samples, XML bibles, RDF samples and IMDb. In Table 1 we provide a comparison of our collection to these two collections. A third study [25] used 601 XHTML Web pages, 3 DocBook XML documents, and documents from the XML Data repository project [52].

No thorough studies on XML well-formedness, our second quality measure, are available. Previous studies concentrate on XML collections with only well-formed documents, as described in Table 1.
Macro-level analysis shows that XML documents are found in all geographic regions and across all major internet domains. 53% of all documents, accounting for 76% of the total file size, can be found at ‘.com’ and ‘.net’ internet domains [3].

In 2003, only 48% of the XML documents from the Xyleme repository refers to a DTD, and 0.09% to an XSD [3].

Most XML documents are small: around 4 Kilobytes. Also, the volume of markup in relation to the actual content of the documents is surprisingly high. Lastly, 99% of the documents had less than 8 levels of element nesting, and 15% appears to have recursive content. This all seems to indicate that most XML documents are not complex [3][25].

2.2 Studies on HTML Web quality

Several surveys on the quality of HTML documents on the Web exist [34][11][4][36]. Although XML’s predecessor HTML differs greatly in applicability, these studies are relevant to this paper because of the similarity in experimental design, data collection and quality measurements.

The sample data across the studies differ between 226 websites from environmental issues [36], 13,312 websites under the ‘co.uk’ domain [4], samples that combined websites from search engines and Alexa.com’s top websites [10], and homepages of the Alexa.com’s top 100,000 websites [34].

The studies use different methods to assess the quality of HTML documents: WebXACT [36], NSGMLS parser [4] and the W3C HTML Validator [11][34]. The differences in sample collections and quality measures do not seem to make a big difference on the results, as all results indicate a poor quality of the HTML documents: a mere 6.5% [4], 5% [10], 4% [36] and 3% [34] of the HTML documents complied with W3C’s HTML standards. The different methods of assessing quality of HTML documents are also interesting because they contain assessment of the encoding through the outcome of the parsers (e.g. [34]).

2.3 Studies on schema languages for XML

Schema languages for XML describe the structure of XML data. They allow automation and optimization of search, integration and processing of XML data [8].

There are three main schema languages in use and one language for specifying dependencies. These are DTD [13], XSD [50], Relax NG [39] and Schematron [41]. Schematron is the language used for expressing dependencies in the form of implications between XPath expressions. As it is rarely used we will not discuss it further. The three other languages are all W3C recommendations. XSD and Relax NG documents are themselves written in XML. Relax NG also has a human-friendly compact syntax. DTDs have their own syntax. No research yet exists on the actual use of Relax NG schemas in documents on the Web.

Schemas for XML have been studied in a number of ways. Firstly, they have been studied in relation to XML collections. As we have seen above, in our sample only a quarter of documents reference a schema. In the Xyleme sample 48% of the documents refer to a DTD, and 0.09% to an XSD [3]. However, in the semi-automatic collection by Mlýnková et al. [33] only 7.4% do not reference a schema; we suspect this might be due to the collection process. Sahuguet [40] found that schemas are often too permissive. For instance, (a|b)* is used to indicate an optional a,b and c, in any order. As is the case with HTML files, the syntax of most DTD files is incorrect [40][12]. This is generally also the case for XSD [7].

Secondly, the actual use of syntactic and semantic features of schema languages are studied. Most of this work has focused on DTDs. DTDs differ greatly in size and form. However, DTDs are generally simple [12][24]. Many features of DTDs are not used or misused; this indicates that the features have not been properly understood. Also, there are many ways to do things in DTDs, and people use hacks to cope with DTD shortcomings [40].

Thirdly, work has been done in developing metrics to measure the quality and complexity of DTDs [24] and XSDs [32]. These metrics might be interesting to use in future quality analyses of our dataset.

Lastly we discuss research on the expressive power of XML schema languages. The three languages are incomparable in expressive power and their effect when validating. For instance, validating a document with a DTD changes the document: default values are added. DTDs have no means to restrict data values to data types like string or integer while this is possible with Relax NG and XSD. Theoretical work on the expressive power of schema languages abstracts many features of the concrete languages and compares their core logical part. DTD is less expressive than XSD, which is less expressive than Relax NG [6][29]. Relax NG has the unlimited form of subtyping of the extended DTD’s from [35] which can specify exactly the regular languages of finite unranked trees. This makes Relax NG expressively complete. While XSDs allow expressions that cannot be expressed in DTD syntax, these extras are rarely used in practice [7][28]. In our study we look at these differences in expressive power from a pragmatic point of view: how often can schemas found on the Web be inter-converted using the schema conversion tool Trang (Cf. Section 4.4)?

3 DATA

We briefly describe our collections of XML and schema files, and how they were obtained.

3.1 Desired Data

The population of the data in this study is the XML Web. The definition of the XML Web used here will be: the subset of the Web made of XML documents only [3]. The population data consists of all kinds of XML documents: RSS, Atom, XSL style sheets, XSD data and XHTML are all written in XML, and are therefore part of the population of the XML Web.

The actual amount of files in the XML Web is unknown. Obtaining an estimate of its size is intrinsically difficult because of its global nature and the loosely structured information it holds, and its sheer size makes exhaustive exploration of the Web prohibitively expensive [1]. The size of the population is, however, irrelevant in calculating the required sample size that will yield confident statistical results, as it is the absolute sample size that matters [47]. Unfortunately, the sample does need to be collected randomly, and randomly collecting XML documents from the Web is difficult. The objective of our study is to assess the quality of the XML Web, and a large collection will maximize the probability that errors are included in the collection. Therefore, we decided to harvest as many XML documents from the Web as we could retrieve from Google and Yahoo’s indexes.

Our method of data collection does not access the Hidden Web [37]. As a consequence, our collections will not contain any data from the Hidden Web.

3.2 Description of Data

We describe some general statistics about the XML collection and compare it to those in two earlier studies, report on the
geographic origins of the data [3], and describe the collection of schema files.

### 3.2.1 XML documents

The XML collection contains 180,640 files having filetype .xml. We will call them XML files even though not all of them are well-formed XML. Table 1 shows that this is 5.1% smaller than the collection used by Barbosa et al. [3], but 992.3% larger than the collection used by Mlynkova et al. [33]. The total file size of the collection is 40 Gigabytes. The largest file in the collection is 683.7 Megabytes, and the smallest is 1 byte. The average file size is approximately 223 Kilobytes.

Duplicates in the collection can bias the statistics of our analysis. We therefore filter out duplicates. Determining duplicates is an intricate process described in Section 3.3. The number of documents that has a duplicate according to our metric described in Section 3.3 is 1296. This means there is a 0.007% chance that a document has a duplicate. The highest number of duplicates of one file is 119.

Because we kept and stored the URL of each document in the collection we can describe the distribution of XML documents on the Web. The regions from which the XML Web is hosted and served can be explored and, up to a point, the underlying institutional goals of the XML can be described (e.g., for XML from educational or commercial domains). The URL of a document contains the site from which it was retrieved. We define a site as the combination of a base domain and the top-level domain. Typically this looks like w3.org. There are files from 96,650 sites in our collection. To gather meaningful data, we have clustered the results of the websites by zones, consisting of generic internet domains and geographical regions. We used the zones defined by Barbosa et al. [3], with the only difference that we define the European Union as of June 2010. Figure 4 shows that 38,197 sites (39.5%) in the collection are in the ‘.com’ domain.

![Figure 4. Distribution of websites by zone.](image)

The EU follows with 25,870 websites (26.8%), and the Rest of the World category accounts for 18,753 websites (19.4%). These results are in line with results by Barbosa et al. [3] with the exception that in geographical terms North America (composed of North America, .edu, .gov and .mil) is under-represented: it accounts for only 3% in our collection, while it accounts for at least 16% in Barbosa’s collection. This might be due to the fact that the harvesting process of our new sample was located in The Netherlands as opposed to North America.

<table>
<thead>
<tr>
<th>Table 1. Comparison of XML collections.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paper</td>
</tr>
<tr>
<td>Amount of files</td>
</tr>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Type of website</td>
</tr>
<tr>
<td>Total size uncompressed</td>
</tr>
<tr>
<td>Total size compressed (.tar.gz)</td>
</tr>
<tr>
<td>Amount of websites</td>
</tr>
<tr>
<td>Amount of duplicates</td>
</tr>
<tr>
<td>Preprocessing</td>
</tr>
</tbody>
</table>

With 180,640 documents and 96,650 sites in our collection, there is an average of 1.87 documents per site. The site ‘gentoo.org’ has the most documents: 451, followed by ‘themann.de’ with 207 documents. The distributions of documents per zone and of document size per zone largely mirrors the distribution of websites in Figure 4.

### 3.2.2 Schemas

Our collection contains 24,426 (13.5%) files with a reference to a DTD. 21,033 (86.1%) of all references use a public identifier, and 24,420 (99.9%) use a system identifier. A public identifier identifies documents of a certain structure across multiple applications, and a system identifier exclusively in one application. Unlike the public identifier, the system identifier is a relative or absolute URL. We therefore used the system identifier to download the DTDs. Of these, 3059 (12.5%) failed to download via the system identifier. The DTD schemas contained a total of 5410 includes of other DTDs or entity documents. These have been downloaded recursively, and the original schemas have been modified to include the locally downloaded schemas. 1786 (33.0%) of them failed to download. In total we downloaded 1375 unique DTD files.

XSD schemas have been extracted from references in the attribute labels “SchemaLocation” and “noNameSpaceSchemaLocation”. Includes have been downloaded recursively, and the original schemas have been modified to include the locally downloaded versions. The collection contains 24,087 files with a reference to an XSD (13.3%). There are files that contain multiple references to XSDs. The maximum amount of references in one
file is 2399, and 90 documents have more than one reference to an XSD. Of the unique URLs with XSD schemas, 217 failed to download. A total of 2110 XSD includes were found. Of these, only 23 (1%) failed to download. The final collection consists of 437 XSDs. The most popular XSD\(^2\) was referenced in 82.5% of all files that reference an XSD. In contrast to DTD references, the list is not dominated by W3C schemas, but rather by sitemaps.org, indicating that XSDs are widely used for sitemaps.

Apart from collecting schema files that were referenced in an XML file, we also collected schema files directly using the same method of restricting searches at Google and Yahoo to specific file extensions. This way we also harvested Relax NG files. In total we collected almost 8000 unique schema files, in particular 3078 DTDs, 4,141 XSDs, 338 Relax NGs in XML and 337 Relax NGs in the compact syntax.

### 3.3 Data Collection

The data was collected using the following 4 steps:

1. Crawl a list of URLs of XML documents from Yahoo and Google.
2. Download the content of each URL.
3. Organize the collection.
4. Determine duplicates.

The list of URLs was created using a modified version of the crawler by Bex et al. [7]. The crawler executes several keyword queries with the filetype restricted to .xml. Only Yahoo and Google were used because they provide a function to limit the search to a specific file type. Bing does not have this function.

Furthermore, Yahoo and Google may return different search results based on the geographical location of the searcher. To minimize bias in our collection based on the geographical location of the crawler, we used the .com domains of both search engines. In addition, we executed queries for all Yahoo’s ‘region’ and Google’s ‘countryCode’ parameters to collect XML files from as many geographical locations as possible.

The results of the first two steps are described in Table 2. Note that the number of downloaded files contains duplicates.

<table>
<thead>
<tr>
<th>Filetype</th>
<th>Unique URLs in List</th>
<th>Files Downloaded</th>
<th>Loss Percentage</th>
<th>Last File Downloaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>188,332</td>
<td>180,640</td>
<td>4.08%</td>
<td>2010-07-17</td>
</tr>
<tr>
<td>XSD</td>
<td>8,416</td>
<td>8,100</td>
<td>3.75%</td>
<td>2010-07-30</td>
</tr>
<tr>
<td>DTD</td>
<td>8,765</td>
<td>8,229</td>
<td>6.12%</td>
<td>2010-07-31</td>
</tr>
<tr>
<td>RNG</td>
<td>8,751</td>
<td>8,447</td>
<td>3.47%</td>
<td>2010-07-30</td>
</tr>
<tr>
<td>RNC</td>
<td>816</td>
<td>753</td>
<td>7.72%</td>
<td>2010-07-30</td>
</tr>
</tbody>
</table>

Files Downloaded contains the number of files that could be downloaded. Loss Percentage is the percentage of URLs that could not be downloaded successfully. Last File Downloaded is provided to give an indication of outliers.

The resulting collection was organized in a MySQL database. The relational schema consists of nine main relations. For each downloaded file, the database stores its URL, its HTTP header, a list of its duplicates in the collection, information on the encoding, a list of all recursively referenced schemas, and all well-formedness and validity errors. The actual XML and schema files are saved on disk with the appropriate id as their filename. Duplicates were not removed from the dataset.

Determining duplicates of XML files is an intricate process because one tree can be serialized in a large number of syntactically different ways. We use the following definition of duplicates: two files are the same if they are identical after removal of all whitespace (as defined in the W3C XML 1.0 specification). We decided to use this definition because of the following three reasons. Firstly, the definition is used in similar research [3][33]. Secondly, the calculations of equality using the W3C definitions are computationally expensive. Thirdly, the W3C definitions are only usable for well-formed XML.

### 3.4 Data Repository

The collected data, the databases with the results of all analyses and programs used in this study are available on the Web for further research [48].

### 4 QUALITY OF XML ON THE WEB

In the following three subsections we look at the basic quality requirements outlined in Section 1.1: character encoding, well-formedness and validity. We are not only interested in the amount of errors but also whether a small amount of error categories (like “opening and ending tag mismatch”) is responsible for a large amount of errors. We also report correlations between errors and other variables. The last subsection contains our results on converting DTD, XSD and Relax NG schemata into each other.

#### 4.1 Encoding

For every document in the collection, we checked whether the encodings as specified either in the HTTP header, in the encoding attribute of the XML declaration or in the Content-Type meta tag (often used in XHTML documents) was correct (meaning that every character in the document is encoded according to the specification). We checked correctness of the encoding using the mb detect_encoding function in PHP. Our main result is that 99.47% of all specified encodings is correct. The vast majority of documents (94%) is encoded in UTF-8 or ISO-8859-1 and these are all correctly specified. The 0.5% (n=900) documents with incorrectly specified encodings are encoded in Windows-1251, Windows-1252 and KOI8-R.

#### 4.2 XML Well-formedness

We created a modified version of the XML parser libxml2 to check whether a document is well-formed. The main change was to make the output uniform for all errors, so that it can be parsed with a few regular expressions for insertion into our database. Every error encountered by libxml2 consists of three parts: error level, error domain and error category. Errors in several error categories are optionally accompanied by other information such as the line number and element name where the error occurred.

The modified version of libxml2 distinguishes four different error levels: no error, warning, recoverable error and fatal error. Before modifications were made, recoverable and fatal errors could not be distinguished. This distinction is, however, crucial for our intended use-case because when a fatal error is found, the XML cannot be parsed at all. Files in which all errors are recoverable can still be parsed. The modified version of libxml2 also categorizes the error that occurred. Our collection contains 74 distinct error categories.

We note that libxml2 is not designed to collect statistics on the number of errors in an ill-formed XML file. It often happens that libxml2 outputs a large amount of fatal errors while fixing just one makes the document well-formed. We, however, believe that the output is still useful for giving directions to research on automatic error repair.

---

\(^2\) http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd
4.2.1 Fatal Errors
We found that 26,377 different files (14.6% of the collection) had at least one fatal error. Figure 5 lists the 10 most common error categories. Note that one document can have multiple errors, so the total sum is higher than the number of bad XML files. ‘Opening and ending tag mismatch’ is encountered in most documents (16,996 docs) followed by ‘Premature end of data in tag’ (14,250 docs). Third is an unknown encoding (11,615 docs). This last error does not necessarily have to be a fatal error, as libxml2 allows specifying the encoding of a document as an external parameter.

Errors from only a small amount of error categories are responsible for the majority of errors and hence follow a Pareto distribution. The Pareto Chart in Figure 6 shows the first nine error categories. Errors from 12% of the error categories (9 from a total of 74) account for 97.3% of the fatal errors.

The amount of fatal errors found per document differs. We found that in non well-formed documents, it is most common that only one fatal error occurs. This is the case in 5708 documents (21.6%). Also in these documents, ‘Opening and ending tag mismatch’ is most often (25.1%) responsible for the document being not well-formed.

Where do the documents with fatal errors come from? Figure 7 shows the distribution of files with fatal errors across the zones described in Section 3.2. In comparison with the original distribution of websites in Figure 4 there is not much difference: the categories '.com', EU and Rest of the World rank highest on the chart. We can conclude that fatal errors in documents occur in every region, and have approximately the same distribution.

![Figure 5. Top Ten Fatal Error Categories.](image)

![Figure 6. Pareto Chart of Fatal Errors per Error Category.](image)

4.2.2 Recoverable Errors
Recoverable errors are those errors that do not make a document ill-formed. We found 883,231 recoverable errors, which is 27.76% of all errors. Over 99% of recoverable errors concern undefined entities. The top undefined entities are &nbsp; (555,571 times, 63.5% of total), &eacute; (65,107 times, 7.4% of total) and &oacute; (24,488 times, 2.8% of total). Indeed, these are not part of XML which only supports 5 entities by default: &amp; (&) &lt; (<), &gt; (>) and &apos; (’). However, they are part of XHTML specifications. It is possible that the errors occur in XHTML documents that would normally support them. The analysis of validation in section 4.3 indeed shows that undefined entities account for only 1.3% of recoverable errors during validation with DTDs.

4.2.3 Warnings
A total of 5,808 warnings were encountered during XML parsing, accounting for only 0.18% of all warnings and errors. The error ‘xmlns: URI is not absolute’ accounts for 96.9% of total warnings.

4.3 XML Validity
This section discusses the results of validity tests of the XML files with respect to DTDs and XSDs. Distributions of the reasons why files that reference a downloadable DTD or XSD are not valid are provided in Figure 2 and Figure 3 and discussed in Section 1.2.

4.3.1 DTD vs XSD
There are 44.7K XML files that reference DTDs or XSDs. Table 3 and Table 4 describe these files.
3. Overview of XML documents referencing a DTD.

<table>
<thead>
<tr>
<th>DTDs</th>
<th>Compiles</th>
<th>Fails to Compile</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Well-formed</td>
<td>6305 (29.8%)</td>
<td>815 (3.9%)</td>
</tr>
<tr>
<td>Not Well-formed</td>
<td>8934 (42.2%)</td>
<td>5107 (24.1%)</td>
</tr>
<tr>
<td></td>
<td>15239 (72.0%)</td>
<td>5922 (28.0%)</td>
</tr>
</tbody>
</table>

4. Overview of XML documents referencing an XSD.

<table>
<thead>
<tr>
<th>XSDs</th>
<th>Compiles</th>
<th>Fails to Compile</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Well-formed</td>
<td>20365 (86.3%)</td>
<td>3010 (12.8%)</td>
</tr>
<tr>
<td>Not Well-formed</td>
<td>166 (0.7%)</td>
<td>52 (0.2%)</td>
</tr>
<tr>
<td></td>
<td>20531 (87.0%)</td>
<td>3062 (13.0%)</td>
</tr>
</tbody>
</table>

The only files for which we can actually check validity are those in the top-left cell of the tables. These are well-formed XML files which reference a schema file that can be compiled (meaning that all referenced schema files can be retrieved and are syntactically correct). Our corpus contains 2046 valid XML files referencing a DTD and 13,950 valid files referencing an XSD. We can conclude that files referencing an XSD are more reliable than those referencing a DTD. They have 59% chance of being valid in general and 68% when they are well-formed and reference a schema which compiles. For files referencing a DTD, these numbers are 9.7% and 32%, respectively.

4.3.2 Geographic Distribution

We checked whether validity errors were over- or under-represented in certain domains and found one significant deviation. Respectively 2% and 4% of all files come from the .edu and .gov domains, but of all files that refer to a well-formed DTD they contribute 11% and 0.9%, respectively.

4.3.3 Most Common Errors

We first look at errors when validating using DTDs. A total of 28 different errors have been found, of which the top ten is shown in Figure 8.

![Figure 8. Top Ten Errors in DTD validation based on occurrence in files.](image)

We take a closer look at the top three validation errors. Analyzing the category ‘No declaration for attribute’ does not give us much information because the additional information is extremely specific. An example of such extremely specific information is: “the attribute ‘available’ on element ‘offer’ is most often not declared” (82,158 times in 26 different files). The next validation category ‘Element content does not follow the DTD’ has a lot of errors concerning CDATA: a text node is encountered where only element nodes are allowed. The elements that miss a declaration most often are ‘DEFINITION’ (60,691 times in 26 different files), ‘memorial’ (60,536 times in 31 different files) and ‘mrow’ (57,260 times in 152 different files).

The ‘no declaration’ errors are an indication that the data is in fact richer than the schema describes. If an application is built on the schema it can thus simply ignore the extra information. The second error (element content does not follow the DTD) is problematic for parsers, and potentially difficult to repair automatically. Still, simple heuristics may perform well in specific cases. For instance, to repair the CDATA errors in XHTML files, all forbidden text nodes could be wrapped in span elements.

Table 5 shows the top five DTDs referenced most often in invalid XML files. The list is dominated by W3C DTDs. The mobile and math DTDs are over-represented, indicating that they might be hard to comply with (or people just don’t care).

<table>
<thead>
<tr>
<th>DTD (Reconstructed URL)</th>
<th>References</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd</a></td>
<td>1405</td>
<td>38.3%</td>
</tr>
<tr>
<td><a href="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd</a></td>
<td>606</td>
<td>16.5%</td>
</tr>
<tr>
<td><a href="http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd">http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd</a></td>
<td>181</td>
<td>4.9%</td>
</tr>
<tr>
<td><a href="http://www.w3.org/TR/MathXHTML2/xhtml-math11-f.dtd">http://www.w3.org/TR/MathXHTML2/xhtml-math11-f.dtd</a></td>
<td>89</td>
<td>2.4%</td>
</tr>
<tr>
<td><a href="http://www.wapforum.org/DTD/xhtml-mobile10.dtd">http://www.wapforum.org/DTD/xhtml-mobile10.dtd</a></td>
<td>64</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

XSDs show roughly the same picture: 93% of all validation errors are of the type ‘this element is not expected’. The top five error types with XSDs are shown in Figure 9.

![Figure 9. Top Five Errors in XSD validation based on occurrence in files.](image)

Table 6 shows the top five XSDs referenced most often in invalid XML files. Sitemaps.org XSDs are referenced by the largest number of documents with validation errors. This might be
due to the fact that these XSDs are simply referenced most often in the collection, as well as due to the fact that the XSDs are complex to work with and are highly susceptible to mistakes.

Table 6. Top Five XSDs responsible for most errors and warnings in distinct number of documents.

<table>
<thead>
<tr>
<th>DTD (Reconstructed URL)</th>
<th>References*</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sitemaps.org/schemas/sitemap/0.9/site.map.xsd">http://www.sitemaps.org/schemas/sitemap/0.9/site.map.xsd</a></td>
<td>5974</td>
<td>93.1%</td>
</tr>
<tr>
<td><a href="http://www.sitemaps.org/schemas/sitemap/09/sitemap.ap.xml">http://www.sitemaps.org/schemas/sitemap/09/sitemap.ap.xml</a></td>
<td>85</td>
<td>1.3%</td>
</tr>
<tr>
<td><a href="http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd</a></td>
<td>47</td>
<td>0.7%</td>
</tr>
<tr>
<td><a href="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xml">http://graphml.graphdrawing.org/xmlns/1.0/graphml.xml</a></td>
<td>2</td>
<td>0.0%</td>
</tr>
<tr>
<td><a href="http://assault.cubers.net/docs/schemas/cuberef.xsd">http://assault.cubers.net/docs/schemas/cuberef.xsd</a></td>
<td>1</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

* DTD referenced in distinct number of documents containing validation errors.

4.3.4 The Good Guys and the Bad Guys

Though not of direct practical value, it is interesting to see which background variables correlate strongly with (in)validity. We have made a selection of a document’s properties that can be useful in predicting its validity. Several properties that one would expect to be a good predictor are not attractive because they are computationally expensive, or can only be used on well-formed XML data. We looked at file size, domain extension, encoding and the type of webserver used for the site.

We used Spearman’s rho [47] to determine if there is a relationship between file size and validation of the document. There indeed is a statistically significant, but weak relationship $r(131,831) = .214, p < .01$.

Regarding the base domain, we did a binary logistic regression analysis. The produced model does indicate that the domain name extension explains variations in validity of the documents ($\chi^2=6087.791, df=334, p < 0.01$). We found 33 domain name extensions with a statistically significant effect ($p < 0.05$). The 5 bad guys are ‘.jp’, ‘.org.au’, ‘.cat’, ‘.gov.uk’, ‘.gov.br’. They are respectively 3.2, 5.1, 3.6, 3.6 and 9.4 times more likely to be invalid than to be valid. The rest are good guys, ranging from 2.226 (.gov) to 24.750 (.im) more likely to be valid than invalid.

Also, from these 33 statistically significant domain name extensions there are seven domain name extensions in the educational and academic domains (containing .edu or .ac), which are more likely to be valid than invalid. Documents from governmental domains in the USA are more likely to be valid than invalid (.gov), while documents from two other governmental domains are less likely to be valid (.gov.br and .gov.uk) than invalid. Another interesting fact is that documents from the .uk domain are generally almost 2.5 times more likely to be valid than invalid, while documents from the governmental domain in the UK (.gov.uk) are 3.6 times more likely to be invalid than valid. It might indicate that documents from the British government are of poorer quality than other documents originating in the UK.

Does it matter for validity whether a website uses a commercial (Microsoft’s IIS) or an open source (Apache) server? The effect is significant but extremely small: documents served by an Apache server are 1.07 times more likely to be valid than documents that are served by Microsoft IIS.

The encoding of a file has a minor effect on validity. The only statistically significant effects we found are for Windows-1251(12) which is twice more likely to be invalid than valid and ISO-8859-1 which is 2.5 times more likely to be valid than invalid.

4.4 Conversions Between Schema Languages

Understanding a schema created by others is difficult. Studying the schema in a human-friendly format can help. Below we argue that Relax NG provides such a format. An indication of quality of a schema is thus whether it can automatically be converted into an equivalent Relax NG schema. We study that question in this section.

We found not a single XML file that refers to a Relax NG schema. This is probably due to the fact that, unlike with DTD and XSD, there is no standard way to refer to a Relax NG schema from within an XML file. This may change in the future, when referencing schemas becomes standardized through the XML model processing instruction, currently being proposed by a W3C working group [23].

Relax NG, however, has several advantages over DTD and XSD, and it has been used to develop large and complex schemas such as the Text Encoding Initiative [44]. We list some of its advantages [49]:

- It is the most expressive language of the three schema languages. In fact it is expressively complete with respect to unranked tree automata and thus to Monadic Second Order Logic.
- It is exponentially more succinct than DTD, because of the interleave operator.
- It is side effect free (the XML document is never altered during validation as happens e.g. with DTD’s default values for attributes).
- It has a user-friendly syntax based on regular expressions, and a typing mechanism that encourages modular design of schemas.

Because DTD and XSD are more widely known and supported by software, it might be that developers are bound to these languages, while they would prefer to develop their schemas in Relax NG [49]. For this reason, James Clarke developed a schema conversion tool, Trang [46] that can convert all three schema languages except from XSD. Trang converts each input schema into the same internal object model and converts that into the required output schema. Because of this design, Trang cannot always convert schemas. Van der Vlist [49] advocates to develop each schema in Relax NG, even if a DTD or XSD is required and then convert it with Trang to the required format. It is claimed that in the vast majority of real world schemas Trang is able to convert them.

We decided to test whether this claim holds for schemas crawled from the Web and also whether Trang can be used for converting in the other direction (from DTD to Relax NG). Because our collection does not contain any Relax NG schemas we also crawled schema files directly. Combined with the schemas that were successfully extracted from references in XML files, we created a collection of 3,087 DTDs, 4,141 XSDs, 337 Relax NGs in compact syntax and 338 Relax NGs in XML. All schemas are syntactically correct.

While XSDs (and thus also Relax NG) allow expressions that cannot be expressed in DTD syntax, these extras are rarely used in practice [6][7], and thus we expect that a high percentage of Relax NG schemas can be converted into DTD. Our experimental results show that 30% of the Relax NG schemas can be converted to DTD and 96% of them to XSD. The surprisingly low 30% seems
due to the use of functions in Relax NG that are not (yet) supported by Trang. We also used Trang to convert DTDs to Relax NG, Relax NG compact and XSD. For all output languages, the conversion succeeded in 2,702 cases (88%). This rather low percentage seems due to Trang, not to the use of DTD features the other languages cannot handle (like default values for attributes).

The database group of the University of Dortmund is working on an improved convertor based on [18], which can also convert from XSD [30].

5 CONCLUSIONS

Our results show that it is possible to do ETL tasks on XML files using only XML query and transformation languages. Only 14.6% is not truly XML. Of course ETL development would not handle XML.

The quality of the XML Web needs drastic improvement as less than 10% is valid. Although it is hard to compare our data with previous studies, it seems to be a positive development that there is a growing number of files referencing an XSD because files that reference an XSD tend to be valid twice as often as those that reference a DTD. We set up our study in such a way that it can be easily replicated in the future. Hopefully, we can measure an upward trend in validity. We did not consider the content of the XML files, whether they were intended for machines or humans, and whether they were data- or document-centric. Also, files from the Hidden Web are not part of our collection and were therefore not considered. These are interesting extra dimensions for future work.

XML syntax errors are Pareto distributed, with just 3 error categories responsible for 80% of all errors. Validation errors occur because referenced schemas are syntactically ill-formed or the schema and the XML file do not match. This shows that work on (semi-) automatically learning DTDs or XSDs from XML documents is useful [5][8]. Most validation errors occur because there is an element or attribute used that is not defined in the schema. This could mean that either the schema is not correct or a wrong name is used in the XML file. Schema learning techniques may be expanded to schema repairing techniques. In this case, techniques used in data-deduplication and learning schema mappings to repair XML documents seem an interesting direction to investigate.

6 REFERENCES