Saaif Ahmed

saaifza2

CS:598 Foundations of Data Curation

Assignment 3: Part 1

1:

File set A Checksum: fa02ed07a59a9873d940b2ce6f8cfdf0

File set B Checksum: ffa1a34dae80e19fdfc917d92e2efc99

These were obtained at this link: https://emn178.github.io/online-tools/md5_checksum.html

For File set A it is as the description of the files says it is. This is an xml document containing customer complaints. Each complaint starts off with an id, it lists the product the issue is concerning, and may or may not have a customer narrative to accompany it. File A's xml is decently well formatted and easy to parse as one reads along it.

For File set B it is again as the description of the files says it is. This is a new schema but still an xml document covering customer complaints. Interestingly this document collects complaints in its own element much like File A but in addition to an id there is a required submission type attribute. This will come up in canonicalization. The format of File B xml is not as clean as File A and should be cleaned up if not for canonicalization but for other curatorial activities.

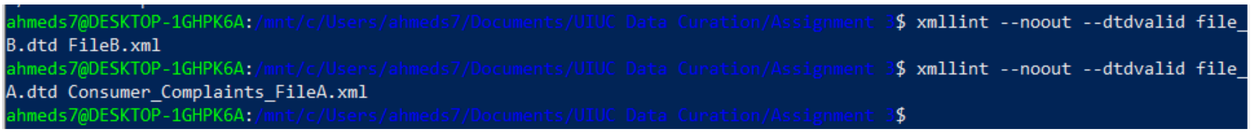
As for the DTD for each file set, they are mostly similar with the differences being that File B DTD has an additional ENTITY line and the submission type attribute. While File A DTD has the additional submitted via attribute. These attributes are very similar in their values in each xml vile which will come up in canonicalization. They should be able to be combined in some fashion without making a meaningful change.

2:

I used the tool xmllint to verify the xml against their dtd files. I used these specific commands.

xmllint --noout --dtdvalid file_B.dtd FileB.xml

xmllint --noout --dtdvalid file_A.dtd Consumer_Complaints_FileA.xml



```
ahmeds7@DESKTOP-1GHPK6A:/mnt/c/Users/ahmeds7/Documents/UIUC Data Curation/Assignment 3$ xmllint --noout --dtdvalid file_
B.dtd FileB.xml
ahmeds7@DESKTOP-1GHPK6A:/mnt/c/Users/ahmeds7/Documents/UIUC Data Curation/Assignment 3$ xmllint --noout --dtdvalid file_
A.dtd Consumer_Complaints_FileA.xml
ahmeds7@DESKTOP-1GHPK6A:/mnt/c/Users/ahmeds7/Documents/UIUC Data Curation/Assignment 3$
```

These were the outputs of the commands in bash shell. This command will throw an error is something goes wrong. Because there was no error message it is shown that the files are valid against their DTD documents.

3:

We start the canonicalization process by using xmllint to handle most of the heavy lifting. Below are the commands used to perform the canonicalization.

```
xmllint -c14n11 Consumer_Complaints_FileA.xml > fileA_c14n.xml
```

```
xmllint -c14n11 FileB.xml > fileB_c14n.xml
```

Starting with file set A we can check the difference between the original xml and the one outputted by xmllint. To begin, the xml definition at the top of the file is removed. From here two major changes are made within the file. The order of the attributes within an element are now ordered in alphabetical order. The "date" attribute now precedes the "type" attribute. In addition, the start and end tags for each element have now been standardized. Take for example the submitted element. Previously it would be closed as <submitted via = "foo"/> but in the canonicalization we standardize it to have start and end tags. Therefore, it becomes <submitted via="Phone"></submitted>.

Moving on to file set B we again check the difference between the original xml and the one outputted by xmllint. To begin, the beginning document and xml information at the top of the document is removed. This is in accordance with standard canonicalization practices. The attributes are then changed to be in alphabetical order within the elements much like what happened to file A. Empty elements have been standardized to start and end tags. Additionally, the embedded references were now replaced with "XXXX". The last thing that xmllint did for file B is adjust the white space within the tags of the doc itself. They have been normalized which has removed excess whitespace in the tags as per the standards of canonicalization.

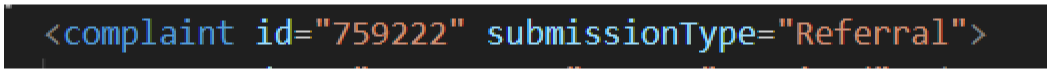
This has covered what xmllint has done for the canonicalization process. What proceeds will be an account of the manual changes made to the documents for canonicalization. Changes will not be meaningful as the documents are meant to be logically equivalent.

To begin we first must beautify file B xml as the format is difficult to analyze and compare to file A otherwise. This was done with an online tool found at the below link.

https://codebeautify.org/xmlviewer#

Beautification does not change relevant whitespace in the files. Within each tag, the whitespace will be preserved, thus the data will be preserved. The movement of the tags as a whole outside of the data is a not a meaningful change in the lens of canonicalization.

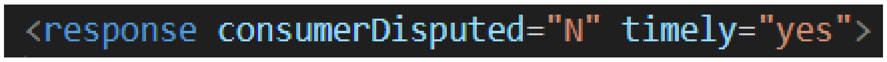
Afterwards we notice the information stored within the submission type object in each file. In file A the "submissionType" is an element, whereas in file B it is an attribute within the elements. While they have similar yet slightly different names in each DTD, one can intuit that these parts of the xml document represent the exact same information. The values for each instance of the submissionType are equivalent between the files further showing that they represent the exact same information. Therefore, for the purposes of canonicalization we can represent the information in the manner of the new file. Which means we remove the element from file A xml and instead have it as attributes in the same manner as file B xml. We chose file B as the template as it represents the new system for the scenario of the of files. Below shows the final form.



```
<complaint id="759222" submissionType="Referral">
```

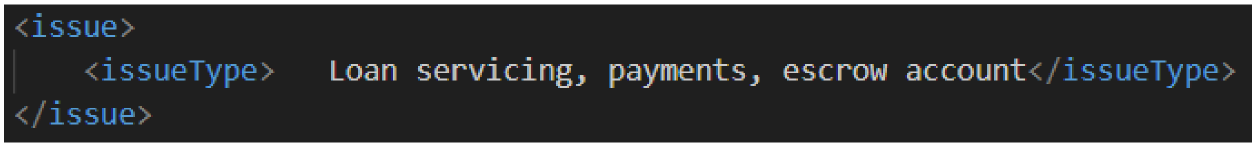
The next change for the canonicalization that xmllit missed was removing all the comments. File B has comments and those need to be removed. Removing those are not a meaningful change thus we can just remove them.

The next step of canonicalization that can be taken is to add attributes to the elements that don't have them. In file B certain attributes are required where they are not required in file A. These attributes are specifically “consumerDisputed” and “timely”. For these we will add the default values and put them on the canonicalized files. For comsumerDispute the value is “N” and for timely the default is “Y”. Additional note to this is that for “timely”, the expected values for the attribute are different between the files. This however is not a meaningful change as it is “Y” in one and “yes” in another, and for the consequent it is “N” and “no”. It is very easy to intuit what they each mean against each other and thus we can make the change to standardize the values for “timely”. We do not try to standardize between consumerDispute and timely because it is not needed for canonicalization. Below shows the final form.

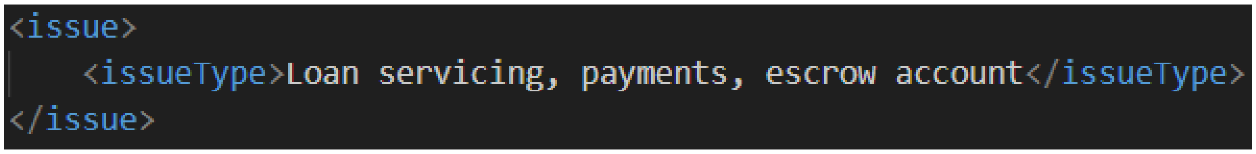


```
<response consumerDisputed="N" timely="yes">
```

The last part of the canonicalization is whitespace. Non-significant white space, including leading and trailing white space can be removed. Below indicates an example of the change that can be made for the purposes of canonicalization.



```
<issue>
    <issueType>   Loan servicing, payments, escrow account</issueType>
</issue>
```



```
<issue>
    <issueType>Loan servicing, payments, escrow account</issueType>
</issue>
```

With that the checksums can now be verified.

Using the website below we compute the MD5 checksum.

https://emn178.github.io/online-tools/md5_checksum.html

Afterwards we report the findings.

File A: d41d8cd98f00b204e9800998ecf8427e

File B: d41d8cd98f00b204e9800998ecf8427e

Thus, the files are logically equivalent.

4:

The XSD schema was created for the canonicalized files. Below will be a brief description of the elements of the document.

consumerComplaints: the main element of the document that encompasses all pieces of data.

complaint: this is one instance of the complaints found in each document. They can be described as a single customer's single complaint against a single financial institution. A complaint has 2 attributes that are required. The id attribute is a unique identifier for each complaint not only on the document but whole data landscape. The submissionType attribute is an indicator of how the complaint was given to the government agency for their records, and it only accepts Referrals, Web, Phone, InPerson, and Mail.

event: Events are timestamps indicating the status of the complaint, whether it was received or sent to a company. They are unbounded for a given complaint as this element can be made numerous times to track the status of the complaint.

product: This is an encompassing element meant to store information about the product.

productType: This is the product offered by the company that the consumer has a complaint towards.

subproduct: The subproduct is an optional element that can apply additional scrutiny to the product with complaints.

issue: This is an encompassing element meant to store information on the issue the consumer has with the product.

issueType: This element holds information about the issue that the consumer has. These are standardized issues.

subissue: The subissue is an optional element that can apply additional scrutiny to the issue the consumer has.

consumerNarrative: This element is an optional element that the consumer can use to write an additional narrative for information outside of the form submission.

company: This is an encompassing element meant to store information about the company who the consumer has complaints towards.

companyName: The name of the company whom the complaint concerns.

companyState: The state in which the company operates in.

companyZip: The zip code of the company office.

submitted: This is an optional empty element. It exists for canonicalization purposes.

response: This is an encompassing element meant to store information about the response the company had for the customer. The response element has 2 required attributes that need mentioning. The consumerDisputed indicates whether the complaint was fought by the customer and only accepts "Y" or "N". The timely attribute indicates whether the complaint was handled in an efficient manner and only accepts "yes" or "no"

publicResponse: This is an optional element where the notes about the company's action can be marked down.

responseType: This element indicates, from a standardized response, the way the company resolved the complaint.

The XSD was made to encompass all the information within the documents and previous DTD files such that as little information was lost as possible.

5:

Using the link below the documents the canonicalized documents were validated. This is the Oxygen XML online validator for XSD schema against a given XML. The results of the validation were a success.

https://www.liquid-technologies.com/online-xsd-validator

6:

Canonicalization plays an important role in data curation. Data curation has a large problem with identification and the goal of canonicalization is to work towards solving those problems. The objectives of data curation the identity problem deals with are for example, storage, preservation, security, reproducibility, and provenance. Canonicalization is a means by which curators may combat the problems of identity.

Canonicalization is in essence, ensuring that two or more data forms are logically equivalent. And this form of equivalence shows how it supports the objectives of data curation. Storage deals with the reliable and effective use and retrieval of data. With canonicalization, we can store the canonical files and determine exactly if the logical representation of a given piece of data exists within some database. Preservation is yet another objective of data curation that canonicalization deals with. As standards and formats improve, the need to preserve the data in these new formats arises. This scenario dealt exactly with ensuring that the data was preserved from one format to a newer format. By creating canonical documents, we can compute a checksum from the documents and verify exactly if the data was preserved in the transition.

Canonicalization's ability to show logical equivalence is also useful for the security of the data. Security begs the question of tampering with the data. If a given dataset has been potentially tampered with, but the canonical form exists, we can canonicalize the dataset to that canonical form and determine how much tampering has occurred if at all. Reproducibility is handled much the same. If we wanted to determine if an XML had the same information as a JSON, we could convert the JSON to an XML and then perform canonicalization on those documents to determine the answer to that question.

Lastly provenance is handled by canonicalization as well for the objectives of data curation. Provenance deals with the inputs, processes, and calculations that are necessary for data values. In essence, we are asking by which means are these data values derived from, or the dataset they are derived from. Canonicalization is a perfect means to complete this objective of data curation. All that needs to occur is the dataset must be made into an equivalent form, and then we can determine the logical equivalence of the data. Imagine we have 2 different datasets and 1 dataset we imagine is the progenitor. We just need to perform canonicalization on each of them to determine logical equivalence and determine where the data was derived from.