Final Project

Part 1 Instructions

1. [5 points] Write a short profile of each file we have given you. Provide a narrative description of the file (50-100 words per file), including its format, properties, contents, and MD5 checksum. This website has a checksum calculator plus gives instructions for running a checksum in Unix and Linux systems: http://onlinemd5.com/; https://emn178.github.io/online-tools/md5_checksum.html

2. [7 points] Create a DTD for each XML file. (Note that File B currently contains a minimal internal DTD; you must expand this DTD to fully capture the elements, attributes, etc., in the document.) You may use a DTD generator to generate drafts of your DTDs, but make sure you manually inspect the resulting DTDs to ensure the results are accurate. DTD generators tend to generate attributes, constraints, and default values that do not accurately reflect the possibilities and requirements for your XML documents. Your choices of attributes, constraints, and default values must be justified. You will lose points if we see useless attributes, inaccurate constraints, or unjustified default values in your DTD. Ensure that your XML documents validate against your DTDs.

3. **[13 points] Canonicalize the two data files and run checksums again to check for equivalence. Please make sure to document your checksum results (which you will report in step 5). Optionally, you may write and employ a script to conduct the canonicalization process; if you do, please make sure to submit a well-documented, readable text file of the script. For an example of canonicalization, see videos for week 10.**

4. [10 points] Create and document the DTD of the final, canonlicalized data file, from step 3 above.

5. [5 points] Be sure that all of your files validate as they will be assessed for compliance to their DTDs. You must test whether your document will validate against your DTD here: http://xmlvalidator.new-studio.org/

Please note that you must select "DTD" as your validation type. This validation tool requires an internal DTD, which means that you must paste your DTD into your XML document, below the XML declaration and above the root element of your XML document. Note also that the syntax for internal vs external DTDs is slightly different.

6. [10 points] In a separate document, answer the following reflection prompts:

a) Describe your process for canonicalization (i.e., decisions, actions, representation selection, attribute issues, provenance decisions). Report the checksum values after canonicalization.

b) How does the way data is represented impact reproducibility?

c) How may your canonicalization support the overarching goals of data curation (revisit objectives and activities of Week 1)?

d) Which additional curation activities would you recommend to enhance the data set for future discovery and use?

Part 2 Instructions [37 points]

1. Write a convincing memo (650 word max) explaining why data curation services are important. Assume that the memo is written for your new director, who is not familiar with data curation, and not convinced whether to keep funding this work. You will want to make sure to introduce data curation within the broader context of data science. You will need to cover the key areas that you think are the most important for data curation at your company. We ask that you incorporate at least two of the following topics into your memo: Provenance, Policy, Metadata, and/or Preservation.

Submission Instructions

1. Submit your documents to Final Project Peer Review for peer grading. Your assignment submission should include:

a) A document with File A and B profiles from part 1, step 1.

b) Final data files from part 1, step 3.

c) Optional: canonicalization script from part 1, step 3.

d) All 3 DTDs from part 1, steps 2 and 4.

e) A document containing your answers to reflection questions in part 1, step 5.

f) A document containing your memo; see part 2.

2. Each student can grade the submissions of 5 of their peers (optional but highly recommended). Submissions will be graded based on the following criteria. Write a constructive and professional review and post to the course forum replying to the individual's submission. a) Is it clearly written? Is the data and canonicalization clearly presented? b) What are the pros and cons of this approach? c) How convincing is the argument for data curation in the memo? How could it be improved?

3. Revise and submit all documents as to Final Project Submission for instructor grading.

Instructor Grading Rubric

Criteria	Max Points
Evidence of in-depth examination of data files (accurate, validating DTDs, and file descriptions) [steps 2 & 5]	12 points
Evidence of understanding of canonicalization (successful canonicalization) [step 3]	13 points
Clarity, accuracy, and documentation of the canonicalization process [steps 1, 4, & 6]	25 points
Convincing and clear argument for data curation and related topics, written in an appropriate tone for the audience [part 2]	37 points
Overall quality analysis and discussion [missing parts?]	13 points
Total	**100 points**

1. Narrative (50 ~ 100 Words)

Consumer_Complaints_FileA:

This file contains complaints filed by Consumers expressing their dissatisfaction with the quality of service that are offered. Every complaints is uniquely identified with Complaint ID and event parameters event-type and data of complaint of the application. The is a place holder for the Product for which we have the complaint, the actual issue. Response field in this file explains the kind of response type, that we responded for the complaint, and in a timely fashion. We also have if customer was satisfied with our response. In summary, this files contains all the necessary fields required to understand the nature of complaint and the how it was responded.

When we run checksum on this file we get the following MD-5 Checksum result :
fed0eda489626463876541c56e93c099

Consumer_Complaints_FileB:

This file contains complaints filed by Consumers expressing their dissatisfaction with the quality of service that are offered. This files is similar with FileA above with subtle difference. Each complaints are uniquely identified with a Complaint ID and also have field submission-type in the complaint element that refers to more details on the filed complaint. Submission-type looks similar with “submitted via” field in FileA. Along the file attributes, most of the attributes are common with File A attributes. In summary, this files contains all the necessary fields required to understand the nature of complaint and the how it was responded.

When we run checksum on this file we get the following MD-5 Checksum result :
2550c686aafbdf320bd74836a511cd54

Folder Explantaion from the report:

Below explains the folder structures in the submitted report folder

canonicalized_xml – Contains the canonicalized xml files for fileA and fileB
dtd – Contains generated dtd files for fileA, fileB and canonicalized fileA and fileB names as final.dtd
dtd_xml – This folder contains xml files with embedded dtd files. This way, when opened in a html browser the xml files are automatically validated.
Scripts – This folder contains the scripts I created for md5 checksum and dtd validations.
Xml – This folder contains the all the x ml files for the project namely the fileA, fileB and canonicalized fileA and fileB.

2. DTD for XML files

DTD for consumer_complaints_fileA

```
<!ELEMENT consumerComplaints (complaint+)>
<!ATTLIST consumerComplaints xmlns  CDATA #FIXED ''>

<!ELEMENT complaint ((event+|product|issue|consumerNarrative*|company|submitted)+, response)>
<!ATTLIST complaint xmlns CDATA #FIXED ''
   id CDATA #REQUIRED
>

<!ELEMENT event EMPTY>
<!ATTLIST event xmlns CDATA #FIXED ''
   date CDATA #REQUIRED
   type CDATA #REQUIRED
>

<!ELEMENT product (productType, subproduct?)>
<!ATTLIST product xmlns CDATA #FIXED ''>
<!ELEMENT productType (#PCDATA)>
<!ATTLIST productType xmlns CDATA #FIXED ''>
<!ELEMENT subproduct (#PCDATA)>
<!ATTLIST subproduct xmlns CDATA #FIXED ''>

<!ELEMENT issue (issueType, subissue?)>
<!ATTLIST issue xmlns CDATA #FIXED ''>
<!ELEMENT issueType (#PCDATA)>
<!ATTLIST issueType xmlns CDATA #FIXED ''>
<!ELEMENT subissue (#PCDATA)>
<!ATTLIST subissue xmlns CDATA #FIXED ''>

<!ELEMENT consumerNarrative (#PCDATA)>

<!ELEMENT company (companyName,companyState,companyZip)>
<!ATTLIST company xmlns CDATA #FIXED ''>
<!ELEMENT companyName (#PCDATA)>
<!ATTLIST companyName xmlns CDATA #FIXED ''>
<!ELEMENT companyState (#PCDATA)>
<!ATTLIST companyState xmlns CDATA #FIXED ''>
<!ELEMENT companyZip (#PCDATA)>
<!ATTLIST companyZip xmlns CDATA #FIXED ''>


<!ELEMENT submitted EMPTY>
<!ATTLIST submitted xmlns CDATA #FIXED ''
   via CDATA #REQUIRED
>

<!ELEMENT response (publicResponse?, responseType)>
<!ATTLIST response xmlns CDATA #FIXED ''
   consumerDisputed CDATA #REQUIRED
   timely CDATA #REQUIRED
>
<!ELEMENT publicResponse (#PCDATA)>
<!ATTLIST publicResponse xmlns CDATA #FIXED ''>
<!ELEMENT responseType (#PCDATA)>
<!ATTLIST responseType xmlns CDATA #FIXED ''>
]>
```

<Narrative>

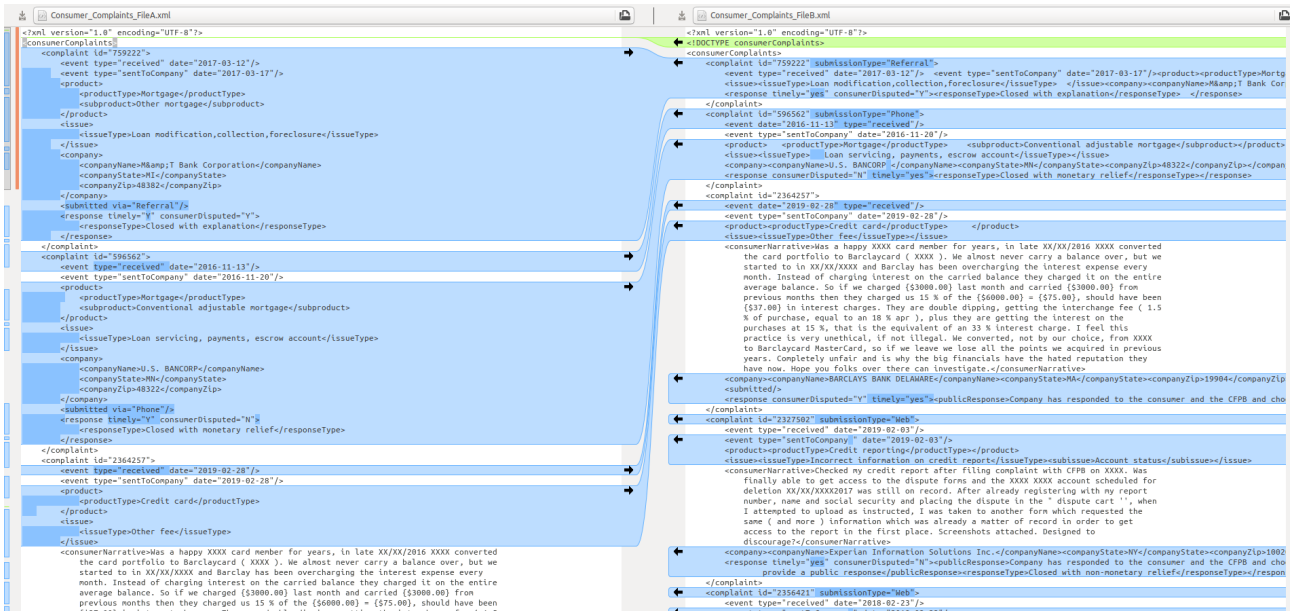
The DTD file for FileA and FileB looks almost similar with subtle difference. Subtle differences are highlighted with RED color. Also please not that these DTD files are validated against the xml files. Please refer to the folder DTD_XML on the report submissions.

DTD for consumer_complaints_fileB

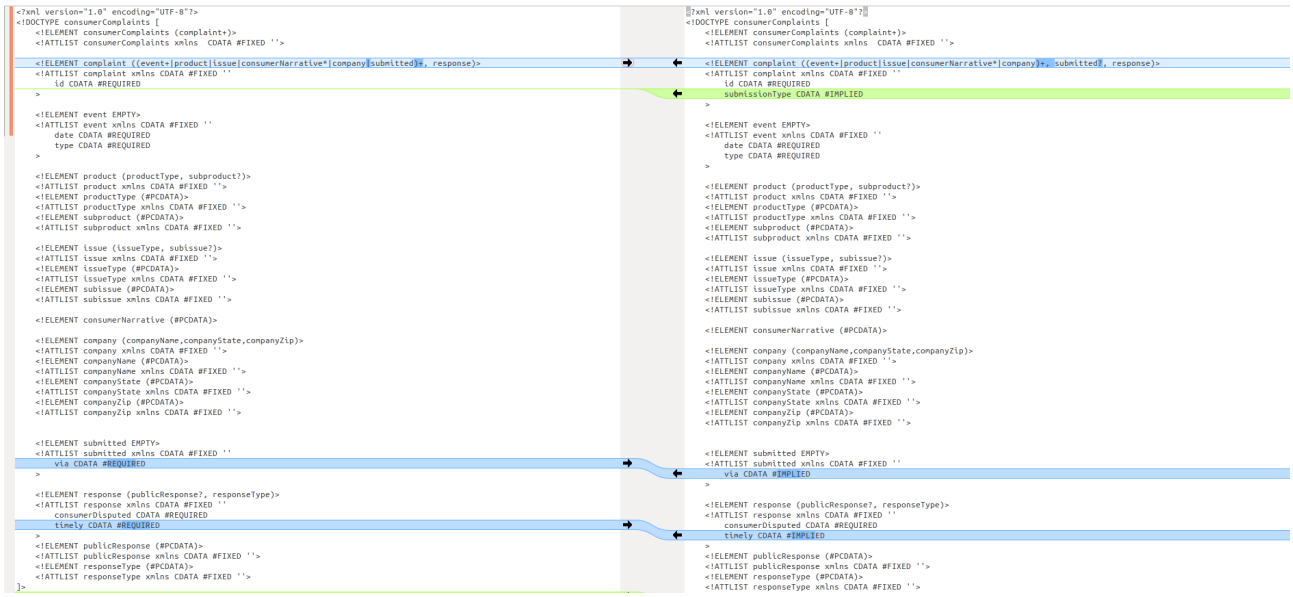
```
<!DOCTYPE consumerComplaints [
<!ELEMENT consumerComplaints (complaint+)>
<!ATTLIST consumerComplaints xmlns CDATA #FIXED ''>

<!ELEMENT complaint ((event+|product|issue|consumerNarrative*|company)+, submitted?, response)>
<!ATTLIST complaint xmlns CDATA #FIXED ''
id CDATA #REQUIRED
submissionType CDATA #IMPLIED
>

<!ELEMENT event EMPTY>
<!ATTLIST event xmlns CDATA #FIXED ''
date CDATA #REQUIRED
type CDATA #REQUIRED
>

<!ELEMENT product (productType, subproduct?)>
<!ATTLIST product xmlns CDATA #FIXED ''>
<!ELEMENT productType (#PCDATA)>
<!ATTLIST productType xmlns CDATA #FIXED ''>
<!ELEMENT subproduct (#PCDATA)>
<!ATTLIST subproduct xmlns CDATA #FIXED ''>

<!ELEMENT issue (issueType, subissue?)>
<!ATTLIST issue xmlns CDATA #FIXED ''>
<!ELEMENT issueType (#PCDATA)>
<!ATTLIST issueType xmlns CDATA #FIXED ''>
<!ELEMENT subissue (#PCDATA)>
<!ATTLIST subissue xmlns CDATA #FIXED ''>

<!ELEMENT consumerNarrative (#PCDATA)>

<!ELEMENT company (companyName,companyState,companyZip)>
<!ATTLIST company xmlns CDATA #FIXED ''>
<!ELEMENT companyName (#PCDATA)>
<!ATTLIST companyName xmlns CDATA #FIXED ''>
<!ELEMENT companyState (#PCDATA)>
<!ATTLIST companyState xmlns CDATA #FIXED ''>
<!ELEMENT companyZip (#PCDATA)>
<!ATTLIST companyZip xmlns CDATA #FIXED ''>


<!ELEMENT submitted EMPTY>
<!ATTLIST submitted xmlns CDATA #FIXED ''
via CDATA #IMPLIED
>

<!ELEMENT response (publicResponse?, responseType)>
<!ATTLIST response xmlns CDATA #FIXED ''
consumerDisputed CDATA #REQUIRED
timely CDATA #IMPLIED
>
<!ELEMENT publicResponse (#PCDATA)>
<!ATTLIST publicResponse xmlns CDATA #FIXED ''>
<!ELEMENT responseType (#PCDATA)>
<!ATTLIST responseType xmlns CDATA #FIXED ''>
]>
```

3. Canonicalize the data files

Please note the difference in the xml file **externally** as displayed by a diff-tool.



Please note the difference in the xml (DTD) file internally as displayed by a diff-tool.



<Narrative>

In this section, we follow the following steps for canonicalization as listed in the lectures

I. Convert to a single character encoding and normalize line ends.
II. Remove all comments, tabs, non-significant spaces, etc.
III. Propagate all attribute defaults indicated in the schema to the elements themselves
IV. Put attribute/value pairs on elements in alpha order
V. Expand all character references
VI. Remove any internal schema or declarations.
VII. Now test to see if character sequences are identical.

Exact steps followed for canonicalization is explained in the ReflectionPoints.

4. Final DTD and Narrative

```
<!DOCTYPE consumerComplaints [
<!ELEMENT consumerComplaints (complaint)+>
<!ATTLIST consumerComplaints xmlns CDATA #FIXED ''>
<!ELEMENT complaint ((company|consumerNarrative|event|issue|product|submitted)+, response)>
<!ATTLIST complaint xmlns CDATA #FIXED ''
id CDATA #REQUIRED
submissionType CDATA #IMPLIED
>
<!ELEMENT event EMPTY>
<!ATTLIST event xmlns CDATA #FIXED ''
date CDATA #REQUIRED
type CDATA #REQUIRED
>
<!ELEMENT product (productType,subproduct?)>
<!ATTLIST product xmlns CDATA #FIXED ''>
<!ELEMENT productType (#PCDATA)>
<!ATTLIST productType xmlns CDATA #FIXED ''>
<!ELEMENT subproduct (#PCDATA)>
<!ATTLIST subproduct xmlns CDATA #FIXED ''>
<!ELEMENT company (companyName,companyState,companyZip)>
<!ATTLIST company xmlns CDATA #FIXED ''>

<!ELEMENT companyName (#PCDATA)>
<!ATTLIST companyName xmlns CDATA #FIXED ''>

<!ELEMENT companyState (#PCDATA)>
<!ATTLIST companyState xmlns CDATA #FIXED ''>

<!ELEMENT companyZip (#PCDATA)>
<!ATTLIST companyZip xmlns CDATA #FIXED ''>

<!ELEMENT consumerNarrative (#PCDATA)>
<!ATTLIST consumerNarrative xmlns CDATA #FIXED ''>

<!ELEMENT issue (issueType,subissue?)>
<!ATTLIST issue xmlns CDATA #FIXED ''>

<!ELEMENT issueType (#PCDATA)>
<!ATTLIST issueType xmlns CDATA #FIXED ''>

<!ELEMENT subissue (#PCDATA)>
<!ATTLIST subissue xmlns CDATA #FIXED ''>

<!ELEMENT submitted EMPTY>
<!ATTLIST submitted xmlns CDATA #FIXED ''
via CDATA #IMPLIED
>

<!ELEMENT response (publicResponse?,responseType)>
<!ATTLIST response xmlns CDATA #FIXED ''
consumerDisputed CDATA #REQUIRED
timely CDATA #IMPLIED
>

<!ELEMENT publicResponse (#PCDATA)>
<!ATTLIST publicResponse xmlns CDATA #FIXED ''>
```

```
<!ELEMENT responseType (#PCDATA)>
<!ATTLIST responseType xmlns CDATA #FIXED ''>
]>
```

<Narrative>

Above is the DTD file that can be used for both File A and File B together. As we can notice from Section 3, even though the files are externally different, when represented by a common DTD file, File A and File B are internally same. The steps that are used to Canonicalize fileA and fileB are explain in detail in Reflection Points document.

5. Files Validation

For validation purpose, I wrote two scripts. Scripts can be found in the scripts folder.

dtd_validator.py

This python script file uses lxml open-source library to validate xml files against their dtd files as per the directions below.

1. Read fileA, fileB, canonicalized_fileA, canonicalized_fileB from xml and canonicalized_xml folders.
2. Read dtd files for fileA, fileB and final from dtd folder.
3. Read file contents in a xml tree format via “etree.XML”
4. validate dtd files against respective xml files using “validate()” api from lxml, etree package.
5. If validate returns TRUE then it is safe to assume that xml files are validated against their dtd files.
6. Below is the out after executing the script

DTD Validations
validate fileA with dtd_fileA True
validate fileB with dtd_fileB True
validate canonicalized_fileA with dtd_final True
validate canonicalized_fileB with dtd_final True

md5_validator.py

This python script file uses hashlib open-source library to perform md5 checksum validations as per the directions below.

1. Read fileA, fileB, canonicalized_fileA, canonicalized_fileB from xml and canonicalized_xml folders.
2. Calculate md5 checksum values on fileA, fileB, canonicalized_fileA and canonicalized_fileB using the hashlib library api ***hashlib.md5(canon_fileA.encode(encoding='utf-8')).hexdigest()***
3. We notice that checksum values for fileA and fileB are different, where as canonicalized_fileA and canonicalized_fileB has same values.
4. Below is the output after executing the script.

MD5-Checksums
md5-checksum for FileA: 93a1afb527c92d9bdeb8d56454c5b7f8
md5-checksum for FileB: 2550c686aafbdf320bd74836a511cd54
md5-checksum for Canonicalized_FileB: 76cb27603b94e4845f7c18f323d4e5a9
md5-checksum for Canonicalized_FileB: 76cb27603b94e4845f7c18f323d4e5a9