PART1

1) Write a short profile of each file we have given you.

FileA.xml description

MD5 checksum D30CBA6B00308A87FA3A384799C5FAF7

This file seems to represent the data for ConsumerComplaints. The parent tag is consumerComplaints which contains multiple complaints pertaining data to multiple complaints.

Each complaint has an id associated with it as an attribute that is unique to every element. Each complaint has events,product,issue, consumerNarrative(which is optional),company,submitted and response. Some of the id don't follow this exact order of tags.

Event has a type and date associated with it.

Product contains detail about the product and have two child tags productType and subproduct(optional)

Issue contains issueType and may contain subissue

Optionally the complaint may contain consumerNarrative in cases of some complaint.

This is followed by company details containing companyName , companyState and companyZip.

Then we have a submitted element that has attribute via having the details on how the complaint was submitted.

Lastly we have response which indicates timely and consumerDisputed as attributes. The response contains a publicResponse(optional) and responseType as well.

FileB.xml description

MD5 checksum 47677272E76E1F4332AFE859347C8695

This file also has data related to consumer complaints. The overall structure of this document is very much similar to FileA.xml but there are few differences. Unlike FileA this File has irregular line breaks.

Each complaint has an id as well as submissionType as attributes. Unlike FileA where submitted was an element we have a submissionType attribute. Each complaint has events,product,issue,consumerNarrative(in some cases),company and response. More or less these elements are similar to those present in FileA.xml.

Event has a type and date associated with it.

Product contains detail about the product and have two child tags productType and subproduct(optional)

Issue contains issueType and may contain subissue

Optionally the complaint may contain consumerNarrative in cases of some complaint.

This is followed by company details containing companyName , companyState and companyZip.

Lastly we have response which indicates timely and consumerDisputed or not as attributes. The attributes value are different from that of FileA.xmlThe response contains a publicResponse(optional) and responseType as well.

There is a tag <submitted/> present in the file which seems to be erroneous.

2) Create a DTD for each XML file

DTD for both the files have been created (internal DTD) and the files are

Consumer_Complaints_FileA_withDTD.xml

Consumer_Complaints_FileB_withDTD.xml

These have been validated for the provide files

Consumer_Complaints_FileA.xml and Consumer_Complaints_FileB.xml

From http://xmlvalidator.new-studio.org/

The DTD files for the provided files have been generated keeping in mind the structure of these two files as described in part1

3)

The files that we get after canonicalization of files Consumer_Complaints_FileA.xml and Consumer_Complaints_FileB.xml are FileA.xml and FileB.xml respectively.

Steps for canonicalization

Before we move forward to canonicalization process , we need to make following changes to the documents

1. FileA
   1.1. We will be removing element submitted that is present inside a complaint and we will instead add an attribute to complaint similar to FileB which is submissionType. The reason for doing is because the element submitted doesn't have any data and also it makes more sense to have it as an attribute because there will be limited number of values for it.
   1.2. Replace values Y as yes and N as no. The reason for this is because Yes and No are more readable and don't introduce any ambiguity.

2. FileB
   2.1. We will fill the missing values of submissionType attribute for some of the complaint ids by taking these values from FileA
   2.2. We will remove the tag </submitted> from our file as it has no matching tag and seems to be erroneous.
   2.3. Replace values Y as yes and N as no. The reason for this is because yes and no are more readable and don't introduce any ambiguity.
   2.4. Remove internal DTD and remove &redaction; and replace it with XXXX wherever required.
   2.5. Add missing values for timely attribute from FileA for two ids

Above changes can be seen with the github commit
https://github.com/elnin09/datacuration/commit/834775f034040fa0e47553ee97a0543397a94c79#diff-985959785319747668373cc6dee294b11db782b03cdd90a2851fbdc0637c6b7b

1. Convert to a single character encoding and normalize line ends.
   1.1. Both files have UTF-8 encoding.
   1.2. For normalizing line ends in VSCode I used https://marketplace.visualstudio.com/items?itemName=sohamkamani.code-eol extension which helped in normalizing line ends

2. Remove all comments, tabs, non-significant spaces, etc.
   2.1. Removed all the non-significant spaces from both the files and tabs. This has been done for both the files.
   2.2. For comments there were only one comment in FileB that was removed

<!-- Note: Sally modified this event on 2015-05-06 -->

3. **Propagate all attribute defaults indicated in the schema to the elements themselves**
   3.1. There were no attributes present in any of the files with default values so we don’t need to change anything in the files for this issue.

4. **Put attribute/value pairs on elements in alpha order**
   4.1. For element event in both the files, put date attribute before type attribute for both files
   4.2. For element response put consumerDisputed attribute before timely for both files
5. **Expand all character references**
6. **Remove any internal schema or declarations**
   This has been removed from FileB which had internal declaration and internal DTD

   Now above changes made can be checked using this commit in FileA.xml and FileB.xml
   https://github.com/elnin09/datacuration/commit/9a3552b4d1917684f73d8b58f28fa89fe2a7fe9f

7. **Now test to see if character sequences are identical.**

   The files are FileA.xml and FileB.xml after above changes and the MD5 checksum from the link
   http://onlinemd5.com/
   is A26D2C1C1962570D68539F5CD932C78F for FileA.xml
   is A26D2C1C1962570D68539F5CD932C78F for FileB.xml

   the checksum is same for both the files after steps 1-6 and hence we can see that the data

4) Create and document the DTD of the final, canonicalized data file, from step 3 above.

The final files in previous step were FileA.xml and FileB.xml which are equivalent. The file with internal DTD and the canonicalized data file is FileC.xml the file have been validated using http://xmlvalidator.new-studio.org/

Explanation/documentation of the DTD

DTD	Comments
<!DOCTYPE consumerComplaints [<!ELEMENT consumerComplaints (complaint*)>	The root of the document ConsumerComplaints contains multiple complaints
<!ELEMENT complaint ((event*,product,issue,consumerNarrative?,company,response) |(company,event,issue,product,event,response))> <!ATTLIST complaint id CDATA #REQUIRED> <!ATTLIST complaint submissionType CDATA #REQUIRED>	Complaint in itself has multiple elements. There are two set of possibilities that’s why we have two sequence of elements for complaint. We will have attributes id and submissionType which is required instead of implied because we have data for this attribute for all the complaint.
<!ELEMENT event (#PCDATA)> <!ATTLIST event type (received|sentToCompany) #REQUIRED> <!ATTLIST event date CDATA #REQUIRED>	Then we have event element with attributes type and data.
<!ELEMENT consumerNarrative (#PCDATA)>	We have consumerNarrative which has normal char data.
<!ELEMENT product (productType,subproduct?)> <!ELEMENT productType (#PCDATA)> <!ELEMENT subproduct (#PCDATA)>	Product contains producttype or may contain subproduct hence we have ? for subproduct productType and subproduct are elements with char data

<!ELEMENT issue (issueType,subissue?)> <!ELEMENT subissue (#PCDATA)> <!ELEMENT issueType (#PCDATA)>	Issue contains issueType or may contain subissue hence we have ? for subissue issueType and subissue are elements with char data
<!ELEMENT company (companyName,companyState,companyZip)> <!ELEMENT companyName (#PCDATA)> <!ELEMENT companyState (#PCDATA)> <!ELEMENT companyZip (#PCDATA)>	Element company contains subelements companyName, companyState and companyZip. companyName, companyState and companyZip are elements with char data
<!ELEMENT response (publicResponse?,responseType)> <!ELEMENT publicResponse (#PCDATA)> <!ATTLIST response timely (yes|no) #REQUIRED> <!ATTLIST response consumerDisputed (yes|no) #REQUIRED> <!ELEMENT responseType (#PCDATA)>]>	Element response may contain publicResponse followed by responseType that’s why we have ? for publicResponse Response also have attributes timely and customerDisputed with values either yes or no. Elements responseType and publicResponse contain char data

5) 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/

The 3 files with internal DTD have been validated FileC.xml, Consumer_Complaints_FileA_withDTD.xml

And Consumer_Complaints_FileB_withDTD.xml

6)

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?

The files that we get after canonicalization of files Consumer_Complaints_FileA.xml and Consumer_Complaints_FileB.xml are FileA.xml and FileB.xml respectively.

The files that we get after canonicalization of files Consumer_Complaints_FileA.xml and Consumer_Complaints_FileB.xml are FileA.xml and FileB.xml respectively.

Steps for canonicalization

Before we move forward to canonicalization process , we need to make following changes to the documents

1. FileA
   1.1. We will be removing element submitted that is present inside a complaint and we will instead add an attribute to complaint similar to FileB which is submissionType. The reason for doing is because the element submitted doesn't have any data and also it makes more sense to have it as an attribute because there will be limited number of values for it.
   1.2. Replace values Y as yes and N as no. The reason for this is because Yes and No are more readable and don't introduce any ambiguity.
2. FileB
   2.1. We will fill the missing values of submissionType attribute for some of the complaint ids by taking these values from FileA
   2.2. We will remove the tag </submitted> from our file as it has no matching tag and seems to be erroneous.
   2.3. Replace values Y as yes and N as no. The reason for this is because yes and no are more readable and don't introduce any ambiguity.
   2.4. Remove internal DTD and remove &redaction; and replace it with XXXX wherever required.
   2.5. Add missing values for timely attribute from FileA for two ids

Above changes can be seen with the github commit https://github.com/elnin09/datacuration/commit/834775f034040fa0e47553ee97a0543397a94c79#diff-985959785319747668373cc6dee294b11db782b03cdd90a2851fbdc0637c6b7b

1. **Convert to a single character encoding and normalize line ends.**
   1.1. Both files have UTF-8 encoding.
   1.2. For normalizing line ends in VSCode I used https://marketplace.visualstudio.com/items?itemName=sohamkamani.code-eol extension which helped in normalizing line ends
2. **Remove all comments, tabs, non-significant spaces, etc.**
   2.1. Removed all the non-significant spaces from both the files and tabs. This has been done for both the files.
   2.2. For comments there were only one comment in FileB that was removed
   <!-- Note: Sally modified this event on 2015-05-06 -->
3. **Propagate all attribute defaults indicated in the schema to the elements themselves**
   3.1. There were no attributes present in any of the files with default values so we don't need to change anything in the files for this issue.

b) The way data is represented in the new file FileC helps to make analysis about customer complaints. The data the way it was represented in previous files didn't have any proper structure and it would have been very difficult to process data or transfer data from one system to another because of the structure of the schema. For eg there are some missing attributes or there is difference in the attribute order for different elements. If we want to transfer data from one source to another it is important to have a common structure. Also the validity of the data can be determined only if we have schema somewhat like FIleC

c)

- *Collection* : The data is collected from two sources to populate the final xml document.
- *Organization* : We are using tree data/DTD model to organize the data
- Storage: This XML document can be stored on any OS and validation of the data can be done as well.
- *Preservation:* The attributes and elements will help in understanding the data. We have a DTD schema and that will help our application in identifying data present inside xml elements
- *Discoverability:* The data can be searched using the elements. There are various frameworks with which we can access our data using keys in our application.
- *Access* The access can be taken care by the File System where the files are stored.
- *Workflow* Support the ability to systematize data workflows. This will also be take care by publishing software. A utility can be written on the top of the apis to automate various workflows and that can be documented. Eg we can document how data will be populated from text document to XML schema and vice versa.
- *Identification* We have various constraints and attributes and keys which will help in authenticating and validating the data. The sample text has also been validated using the online validator
- *Integration* The data can be integrated from various complaint sources i.e web mobile using the XML schema to represent data.
- *Security*: This is also managed by File System but we need to ensure that the permissions and ACLs are managed accordingly
- *Compliance* This will be ensured by the legal team of the organization. Any data that we need to populate in the xml document should be first approved by the legal team. Eg some of the information is redacted in the texts

d) I would recommend following activities to enhance data set for future discovery

- Develop metadata to support searching relevant data from the files
- Maintain a preservation strategy so that data is usable and understandable
- Create well maintained scripts for executing data transformations
- Any modifications or corrections in data must be managed so that we avoid any errors and changes can be tracked.