Big Data For Engineers – Exercises

Spring 2020 – Week 6 – ETH Zurich

XML validation

1. XML Data Models – Information Sets

XML "Information Set" provides an abstract representation of an XML document—it can be thought of as a set of rules on how one would draw an XML document on a whiteboard.

Draw the Information Set trees for the following XML documents. You can confine your trees to only have the following types of information items: document information item, elements, character information items, and attributes.

Document 1

```
<Burger>
  <Bun>
    <Pickles/>
    <Cheese origin="Switzerland" />
    <Patty/>
  </Bun>
</Burger>
```

Document 2

```
<Band name="Metallica">
  <Member>James Hetfield</Member>
  <Member>Lars Ulrich</Member>
  <Member>Kirk Hammett</Member>
  <Member>Robert Trujillo</Member>
</Band>
```

Document 3

```
<catalog>
  <!-- A list of books -->
  <book id='bk101'>
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date version='hard' version2='soft'>2000-10-01</publish_date>
  </book>
</catalog>
```

Document 4

```
<eth:eth xmlns:eth="http://www.ethz.ch"
        xmlns:ethdb="http://www.dbis.ethz.ch"
        date="11.11.2006"
        ethdb:date="12.11.2006">
  <eth:date>16.11.2017</eth:date>
  <eth:president since="2015">Prof. Dr. Lino Guzzella</eth:president>
  <ethdb:Rektor>Prof. Dr. Sarah M. Springman</ethdb:Rektor>
</eth:eth>
```

2. Validate JSON

In this task we will use the validate function from python's jsonschema to validate Json objects.

First, import Json and the validate function.

In [2]:

```
import json
from jsonschema import validate
```

Define the schema.

In [1]:

```
schema={'type': 'object', 'properties': {'target': {'type': 'string'}, 'choices': {'type': 'array', 'items': {'type': 'string'}},\
'guess': {'type': 'string'}, 'date': {'type': 'string'}, 'country': {'type': 'string'}, 'sample': {'type': 'string'}} ,\
'required': ['target', 'choices', 'date', 'country', 'sample']}
```

Define an array of JSON objects to validate.

In [3]:

```
Records=[{'guess': 'Norwegian', 'target': 'Norwegian', 'country': 'AU', 'choices': ['Maori', 'Mandarin', 'Norwegian', 'Tongan'], 'sample': '4
8f9c924e0d98c959d8a6f1862b3ce9a', 'date': '2013-08-19'},
{'guess': 'Dinka', 'target': 'Dinka', 'country': 'AU', 'choices': ['Danish', 'Dinka', 'Khmer', 'Lao'], 'sample': 'af5e8f27cef9e689a070b8814dcc
02c3', 'date': '2013-08-19'},
{'guess': 'Turkish', 'target': 'Samoan', 'country': 'AU', 'choices': ['German', 'Hungarian', 'Samoan', 'Turkish'], 'sample': '509c36eb58dbce
009ccf93f375358d53', 'date': '2013-08-19'},
{'guess': 'Latvian', 'target': 'Somali', 'country': 'AU', 'choices': ['Danish', 'Korean', 'Latvian', 'Somali'], 'sample': 'a505ab771ae7c32744ad
31b3051b8ee9', 'date': '2013-08-19'},
{'guess': 'Japanese', 'target': 'Japanese', 'country': 'AU', 'choices': ['Bangla', 'Dinka', 'Italian', 'Japanese'], 'sample': '3569611136ea04b
ab18a0cd605ced358', 'date': '2013-08-19'},
{'guess': 'Maltese', 'target': 'Turkish', 'country': 'AU', 'choices': ['Hindi', 'Lao', 'Maltese', 'Turkish'], 'sample': 'af0e25c7637fb0dcdc56fac6
d49aa55e', 'date': '2013-08-19'},
{'guess': 'French', 'target': 'French', 'country': 'AU', 'choices': ['Burmese', 'Danish', 'French', 'Swedish'], 'sample': '92f9e1c17e6df98878
0527341fdb471d', 'date': '2013-08-19'},
{'guess': 'German', 'target': 'German', 'country': 'AU', 'choices': ['German', 'Serbian', 'Swedish', 'Vietnamese'], 'sample': 'e77d97b712a
dffc39e531e20237a5589', 'date': '2013-08-19'},
{'guess': 'Spanish', 'target': 'Spanish', 'country': 'AU', 'choices': ['Amharic', 'Czech', 'Sinhalese', 'Spanish'], 'sample': 'dc3ace49393de5
18e87d4f8d3ae8d9db', 'date': '2013-08-19'},
{'guess': 'Romanian', 'target': 'Romanian', 'country': 'AU', 'choices': ['Estonian', 'Japanese', 'Lao', 'Romanian'], 'sample': '901fc45cba52
45f21deaaca3966f825b', 'date': '2013-08-19'}]
```

Validate the Json objects with the "validate" function (if no exceptions are raised, the objects are valid).

In [4]:

```
for line in Records:
    validate(instance=line,schema=schema)
```

Define and validate a new object. What is the problem with this object? Can you change the object to become valid?

In [5]:

```
obj={'guess': 'Dinka', 'target': 'Dinka', 'country': 'AU', 'choices': 'Dinka', 'sample': 'af5e8f27cef9e689a070b8814dcc02c3', 'date': '2013-0
8-19'}
validate(instance=obj,schema=schema)
```

```
---------------------------------------------------------------------------
ValidationError                           Traceback (most recent call last)
<ipython-input-5-3d87b42f16d9> in <module>()
      1 obj={'guess': 'Dinka', 'target': 'Dinka', 'country': 'AU', 'choices': 'Dinka', 'sample': 'af5e8f27cef9e689a070b8814dcc02c3', 'date': '2
013-08-19'}
----> 2 validate(instance=obj,schema=schema)
```

```
~/anaconda3_420/lib/python3.5/site-packages/jsonschema/validators.py in validate(instance, schema, cls, *args, **kwargs)
    476         cls = validator_for(schema)
    477     cls.check_schema(schema)
--> 478     cls(schema, *args, **kwargs).validate(instance)

~/anaconda3_420/lib/python3.5/site-packages/jsonschema/validators.py in validate(self, *args, **kwargs)
    121         def validate(self, *args, **kwargs):
    122             for error in self.iter_errors(*args, **kwargs):
--> 123                 raise error
    124
    125         def is_type(self, instance, type):

ValidationError: 'Dinka' is not of type 'array'

Failed validating 'type' in schema['properties']['choices']:
    {'items': {'type': 'string'}, 'type': 'array'}

On instance['choices']:
    'Dinka'
```

Can you change the schema so that the object becomes valid?

In [ ]:

```
schema={'type': 'object', 'properties': {'target': {'type': 'string'}, 'choices': {'type': 'array', 'items': {'type': 'string'}},\
'guess': {'type': 'string'}, 'date': {'type': 'string'}, 'country': {'type': 'string'}, 'sample': {'type': 'string'}} ,\
'required': ['target', 'choices', 'date', 'country', 'sample']}

obj={'guess': 'Dinka', 'target': 'Dinka', 'country': 'AU', 'choices': 'Dinka', 'sample': 'af5e8f27cef9e689a070b8814dcc02c3', 'date': '2013-08-19'}

validate(instance=obj,schema=schema)
```

3. XML Schema

XML Schema is a way to provide schemas for XML documents, i.e., to describe certain restrictions on the structure and content of XML documents such as, for example, " `<birthday>` elements should only contain valid dates".

In this task we will explore XML Schemas in detail.

To test XML validation, you can either use an online validator like this one or use *oXygen* again. When you open an XML Schema in oXygen, you can switch to its graphical representation, by choosing the "Design" mode at the bottom of the document pane; "Text" mode shows the XML Schema as an XML document.

3.0 Episode 0

Match the following XML documents to XML Schemas that will validate them:

Document 1

```
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```

Document 2

```
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <health/>
  <friends/>
  <family/>
</happiness>
```

Document 3

```
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  3.141562
</happiness>
```

Document 4

```
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <health value="100"/>
  <friends/>
  <family/>
</happiness>
```

Document 5

```
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <health/>
  <friends/>
  <family/>
  But perhaps everybody defines it differently...
</happiness>
```

Schema 1

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="happiness">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="health"/>
        <xs:element name="friends"/>
        <xs:element name="family"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema 2

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="happiness">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element name="health"/>
        <xs:element name="friends"/>
        <xs:element name="family"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema 3

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="happiness" type="xs:decimal"/>
</xs:schema>
```

Schema 4

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="happiness">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema 5

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="happiness">
```

```
      <xs:complexType>
        <xs:sequence>
          <xs:element name="health">
            <xs:complexType>
              <xs:attribute name="value" type="xs:integer" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="friends"/>
          <xs:element name="family"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

3.1 Episode 1

The Great Language Game was a game in which you are given a voice clip to listen, and you are asked to identify the language in which the person was speaking. It was a multiple-choice question–you make your choice out of several alternatives. The game is closed now, unfortunately ;(But, we can still use their datasets for our exercises!

The following XML document presents a user's attempt at answering a single question in the game: it contains the identifier of the voice clip, the choices presented to the player, and the player's response. Provide an XML Schema which will validate this document (use oXygen to help you):

```
<?xml version="1.0" encoding="UTF-8"?>
<attempt
  country="AU" date="2013-08-19">
  <voiceClip>48f9c924e0d98c959d8a6f1862b3ce9a</voiceClip>
  <choices>
    <choice>Maori</choice>
    <choice>Mandarin</choice>
    <choice>Norwegian</choice>
    <choice>Tongan</choice>
  </choices>
  <target>Norwegian</target>
  <guess>Norwegian</guess>
</attempt>
```

3.2 Episode 2

Continuing the topic of the Great Language Game, provide an XML Schema which will validate the following document:

```
<?xml version="1.0" encoding="UTF-8"?>
<attempts>
  <attempt country="AU" date="2013-08-19">
    <voiceClip>48f9c924e0d98c959d8a6f1862b3ce9a</voiceClip>
    <choices>
      <choice>Maori</choice>
      <choice>Mandarin</choice>
      <choice>Norwegian</choice>
      <choice>Tongan</choice>
    </choices>
    <target>Norwegian</target>
    <guess>Norwegian</guess>
  </attempt>
  <attempt country="US" date="2014-03-01">
    <voiceClip>5000be64c8cc8f61dda50fca8d77d307</voiceClip>
    <choices>
      <choice>Finnish</choice>
      <choice>Mandarin</choice>
      <choice>Scottish Gaelic</choice>
      <choice>Slovak</choice>
```

```
            <choice>Swedish</choice>
            <choice>Thai</choice>
        </choices>
        <target>Slovak</target>
        <guess>Slovak</guess>
    </attempt>
    <attempt country="US" date="2014-03-01">
        <voiceClip>923c0d6c9e593966e1b6354cc0d794de</voiceClip>
        <choices>
            <choice>Hungarian</choice>
            <choice>Sinhalese</choice>
            <choice>Swahili</choice>
        </choices>
        <target>Hungarian</target>
        <guess>Sinhalese</guess>
    </attempt>
</attempts>
```

This concludes our exercise sheet on XML validation. Thank you for taking your time to go through it!