How to use saopy (source: http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/saopy.html)

saopy depends on:

1. rdflib (http://rdflib.org/) v2.4.0 (easy_install)

saopy can be downloded from the following link: SAOPY (v1.1.4). Since it is an ongoing work, please make sure that you are using the latest version. In order to install SAOPY library, you have to use the following command in the directory that you have downloaded the SAOPY python wheels.

```
$ pip install ./saopy-1.1.4-py2.py3-none-any.whl
```

then start python using

```
$ python
```

Now let's open the python interpreter and give it a go...

```
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.56)] on
darwin
Type "help", "copyright", "credits" or "license" for more
information.
```

Start by importing the saopy package :

```
>>> import saopy
```

To view all "saopy" classes:

```
>>> dir(saopy)
['DUL', 'PropertySet', 'RDFInterface', 'SaoInfo',
'__builtins__', '__doc__', '__file__', '__name__',
'__package__', '__path__', 'ces', 'ct', 'exportRDFFile',
'exportRDFGraph', 'foaf', 'geo', 'importRDFFile',
'importRDFGraph', 'model', 'muo', 'owl', 'owlsg', 'owlss',
'owlssc', 'owlssp', 'owlssrp', 'prov', 'qoi', 'rdfs', 'sao',
'ssn', 'tl', 'tm', 'tzont']
```

To view all classes of sao ontology from saopy library:

```
>>> dir(saopy.sao)
['DiscreteCosineTransform', 'DiscreteFourierTransform',
'Mean', 'Median', 'PiecewiseAggregateApproximation', 'Point',
'Segment', 'StreamAnalysis', 'StreamData', 'StreamEvent',
'SymbolicAggregateApproximation', '__builtins__', '__doc__',
'__file__', '__name__', '__package__', '__path__', 'saopy']
```

Creating a sensor object:

```
>>> sensor124 = saopy.ssn.Sensor("http://example.org/x")
```

Why shall we use saopy for annotation of IoT data? Saopy enables a user to avoid various common problems including use of undefined properties and classes, poorly formed

namespaces, problematic prefixes, literal syntax and other optional heuristics.

```
>>> sensor124 = saopy.ssn.Senzor("http://example.org/x")
Traceback (most recent call last):
File "", line 1, in
AttributeError: 'module' object has no attribute 'Senzor'
>>> cityofaarhus = saopy.foaf.Organisation("http://
example.org/cityofaarhus")
Traceback (most recent call last):
File "", line 1, in
AttributeError: 'module' object has no attribute
'Organisation'
```

To mash up multiple sensory objects for serialisation, you can use the ``SaoInfo" class. The following example shows how to create and export a sensor observation object with a value:

```
>>> trafficData124 = saopy.sao.StreamData("http://example.org/
data1")
>>> trafficData124.value = "1234"
>>> saoOut = saopy.SaoInfo()
>>> saoOut.add(trafficData124)
>>> saopy.RDFInterface.exportRDFFile(saoOut,”example1.rdf”,
"n3")
```

Now that we know how to describe a simple observation, let's start annotating more detailed sensory observation. First create provenance information for sensory data:

```
>>> cityofaarhus = saopy.foaf.Organization("http://
example.org/cityofaarhus")
>>> cityofaarhus = saopy.prov.Agent("http://example.org/
cityofaarhus")


>>> trafficsensor158324 = saopy.ssn.Sensor("http://
example.org/data158324")
>>> trafficsensor158324.actedOnBehalfOf = cityofaarhus
```

creating properties of sensory data:

```
>>> measuredTime = saopy.ssn.Property("http://unis/ics/
property003")
>>> measuredTime.description = "Measured Time"
>>> estimatedTime = saopy.ssn.Property("http://unis/ics/
property004")
>>> estimatedTime.description = "Estimated Time"
>>> avgSpeed = saopy.ssn.Property("http://unis/ics/
property001")
>>> avgSpeed.description = "Average Speed"
>>> vcCount = saopy.ssn.Property("http://unis/ics/
property002")
>>> vcCount.description = "Vehicle Count"
>>> trafficsensor158324.observes.add(vcCount)
```

```
>>> trafficsensor158324.observes.add(avgSpeed)
>>> trafficsensor158324.observes.add(estimatedTime)
>>> trafficsensor158324.observes.add(measuredTime)
```

SAOPY allows to describe time **instants** and **intervals**. Time instants should only be used with **tl:at** to describe the time instance that data has been collected, whereas time interval should specify both the beginning time of the interval and duration. Here we provide both of the examples.

```
>>> universaltimeline = saopy.tl.PhysicalTimeLine("http://
purl.org/NET/c4dm/timeline.owl#universaltimeline")
>>> instant = saopy.tl.Instant("http://unis/ics/timeinstant")
>>> interval = saopy.tl.Interval("http://unis/ics/
timeinterval")
>>> instant.at = "2014-09-30T06:00:00"
>>> instant.onTimeLine = universaltimeline
>>> interval.beginsAtDateTime = "2014-09-30T06:00:00"
>>> interval.durationXSD = "PT5M"
```

SAO ontology subsumes the measurement unit descriptions from Measurement Unit Ontology (muo). Therefore, it enables to describe measurement unit of an observation as follows:

```
>>> unitseconds = saopy.muo.UnitOfMeasurement("http://unis/
ics/unit1:seconds")
>>> unitkilometer = saopy.muo.UnitOfMeasurement("http://unis/
ics/unit2:km-per-hour")
```

Now, we can annotate sensor observations for two sensor features, namely average speed and measure time:

```
>>> trafficData001 = saopy.sao.StreamData("http://unis/ics/
trafficdataavgspeed001")
>>> trafficData001.value = "60"
>>> trafficData001.hasUnitOfMeasurement=unitkilometer
>>> trafficData001.observedProperty = avgSpeed
>>> trafficData001.observedBy = trafficsensor158324
>>> trafficData001.time = instant

>>> trafficData003 = saopy.sao.StreamData("http://unis/ics/
trafficdataMeasuredTime001")
>>> trafficData003.value = "30"
>>> trafficData003.hasUnitOfMeasurement=unitseconds
>>> trafficData003.observedProperty = measuredTime
>>> trafficData003.observedBy = trafficsensor158324
>>> trafficData003.time = interval
```

exporting sensor data in N3 format:

```
>>> saoOut.add(trafficData001)
>>> saoOut.add(trafficData003)
>>> saoOut.add(cityofaarhus)
```

```
>>> saoOut.add(trafficsensor158324)
>>> saoOut.add(estimatedTime)
>>> saoOut.add(measuredTime)
>>> saoOut.add(avgSpeed)
>>> saoOut.add(vcCount)
>>> saoOut.add(unitkilometer)
>>> saoOut.add(unitseconds)
>>> saoOut.add(universaltimeline)
>>> saoOut.add(instant)
>>> saoOut.add(interval)
>>> saopy.RDFInterface.exportRDFFile(saoOut, "example2.rdf",
"n3")
```

saopy also allows to annotate quality values, such as correctness, frequency, age and completeness. The following example illustrates how to annotate quality features for a data segment of the traffic data stream. The correctness has been obtained based on an algorithm by examining the difference between two successful submitted traffic streams to validate the correctness quality value.

```
>>> segmentSample1 = saopy.sao.Segment("segment-sample-1")
>>> age = saopy.qoi.Age("age-sample-1")
>>> age.hasAge = "10"
>>> completeness = saopy.qoi.Completeness("completeness-
sample-1")
>>> completeness.hasCompleteness = "1"
>>> correctness = saopy.qoi.Correctness("correctness-
sample-1")
>>> correctness.hasCorrectness = "1"
>>> frequency = saopy.qoi.Frequency("frequency-sample-1")
>>> frequency.hasFrequency = "10"
>>> segmentSample1.hasQuality.add(frequency)
>>> segmentSample1.hasQuality.add(correctness)
>>> segmentSample1.hasQuality.add(completeness)
>>> segmentSample1.hasQuality.add(age)
>>> owner = saopy.prov.Person("MisterX")
>>> segmentSample1.hasProvenance = owner
>>> saoOut = saopy.SaoInfo()
>>> saoOut.add(segmentSample1)
>>> saoOut.add(age)
>>> saoOut.add(completeness)
>>> saoOut.add(correctness)
>>> saoOut.add(frequency)
>>> saoOut.add(owner)
>>> saopy.RDFInterface.exportRDFFile(saoOut, "example3.rdf",
"n3")
```