pysemgen Documentation

Kyle Medley

Dec 12, 2018

Contents

CHAPTER 1

Semantic Annotations for Biological Models

This Python package is a Py4J wrapper for SemGen, a package for annotating biological models with semantic information precisely describing the chemistry and physics behind a particular model. Usage of this software allows the math behind a model (differential equations and rates of change) to be traced back to the biology (what a particular variable represents - mRNA, protein, metabolite). Chemical species can be traced back to their respective molecular structures via ChEBI ids, and proteins can be traced back to their respective entries in the Protein Ontology.

This type of auditing is quite common for curated model repositories such as BioModels. However, a crucial feature of SemGen is support new and more expressive types of annotations. Consider the following not unlikely scenario based on a model by Smith, Chase, Nokes, Shaw and Wake (2004). The model describes blood flow in the four chambers of the heart. This process can be described by annotating the model element for blood with FMA:9670. However, in order to describe the physical location of the element for bloody (say, the left ventricle), we need to create a new entity with a separate annotation pointing to FMA:9466 (left ventricle). SemGen accomplishes this with the following *composite annotation*:

```
@prefix bqb: <http://biomodels.net/biology-qualifiers/> .
@prefix dc: <http://purl.org/dc/terms/> .

<./smith_chase_nokes_shaw_wake_2004.cellml#left_ventricle_V_lv>
  bqb:isVersionOf <http://identifiers.org/opb/OPB_00154> ;
  bqb:isPropertyOf <http://njh.me/#entity_0> ;
  dc:description "Left ventricular blood volume" .

<.#entity_0>
  bqb:isPartOf <http://njh.me/#entity_1> ;
  bqb:is <http://identifiers.org/fma/FMA:9670> .

<.#entity_1> bqb:is <http://identifiers.org/fma/FMA:9466> .
```

CHAPTER 2

Installing

To install pysemgen:

```
pip install pysemgen
```

You will also need the SemGen jar containing the Py4J server. You can run the server with:

```
java -classpath SemSimAPI.jar semsim.Py4J
```

CHAPTER 3

Quickstart

The Python wrapper is versioned using the `SemSimLibrary.SEMSIM_VERSION` variable from the Java API.

```
Py4J server started.
>>> import semgen
>>> print(semgen.__version__) # SemSimLibrary.SEMSIM_VERSION in Java
0.2
```

PySemgen supports loading SBML models with *semgen.load_sbml_file()*, *semgen.load_sbml_str()*. CellML models can be loaded with *semgen.load_cellml_file()*, *semgen.load_cellml_str()*. Antimony is a human-readable abstraction of SBML. If it is installed as a Python package, the function *semgen.load_antimony_str()* can be used to load an Antimony string directly. Once a model is loaded, one can iterate through the physical entities present in the model.

```
>>> from semgen import load_antimony_str
>>> model = load_antimony_str('''
... model mymodel
...     const compartment cell_comp
...     var species ATP in cell_comp, ADP in cell_comp
...     J0: ATP -> ADP; k*ATP
...     k = 1
...     ATP = 1
... end
... ''')
>>> for e in model.physical_entities:
...     print(e.name, e.metaid, e.description)
```

To iterate through the entities in a model.

```
>>> for e in model.physical_entities:
...     print(e.name, e.metaid, e.description)
```

Model elements can also be accessed by name.

```
>>> s = model.ATP
>>> r = model.J0
```

The semantic annotations for a given element can be accessed via the *terms* property. New annotations can be added using += (using the bqb:is qualifier by default).

```
>>> # clear existing terms for ATP and ADP
>>> model.ATP.terms.clear()
>>> model.ADP.terms.clear()
>>> # now add new terms for the molecules
>>> from semgen import ChEBI, GO
>>> model.ATP += ChEBI(15422) # terms can be specified using the numeric ontology term
>>> model.ADP += ChEBI.ADP # can also use common name alias
>>> model.cell_comp += GO.cell
>>> # print out the updated definition uris for our model entities
>>> for e in model.physical_entities:
...     print(e.name, e.metaid, e.description)
...     for term in e:
...         print('  ', term)
```

To specify physical properties (a type of composite annotation), locate the smith et al. model inside its COMBINE archive and then use `setPhysicalProperty`.

```
>>> model = load_cellml_file('smith_chase_nokes_shaw_wake_2004.omex/smith_chase_nokes_
↪shaw_wake_2004.cellml')
>>> # set the physical property for the left ventricle
>>> model['left_ventricle.V_lv'].setPhysicalProperty(
...     term_uri=OPB(154),
...     entity_uris={FMA(9671): 'Portion of blood+1',
...                  FMA(9292): 'Cavity of right ventricle+1'},
...     desc='Fluid volume')
```

CHAPTER 4

API

semgen.**load_sbml_str**(*sbml*)
Loads an SBML model from a string.

Parameters **sbml** (*str*) – The raw SBML/XML content.

Returns A SemSim model constructed from the SBML model.

Return type *ModelWrapper*

class semgen.**DataStructureWrapper**(*datastructure*)
Bases: object

description

metaid

name

setPhysicalProperty(*term_uri*, *entity_uris*, *desc=''*)

semgen.**bqb_isDescribedBy**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**CHEBI**
alias of semgen.chebi.ChEBI

semgen.**load_sbml_file**(*filename*)
Loads an SBML model from a file.

Parameters **filename** (*str*) – The name of the file to load.

Returns A SemSim model constructed from the SBML model.

Return type *ModelWrapper*

semgen.**bqb_isVersionOf**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**load_cellml_file**(*filename*)
Loads a CellML model from a file.

Parameters **filename** (*str*) – The name of the file to load.

Returns A SemSim model constructed from the CellML model.

Return type *ModelWrapper*

semgen.**bqb_hasVersion**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**searchbp**(*term*, *ontology*, *n_results*)

semgen.**chebi**
alias of semgen.chebi.ChEBI

class semgen.**OPB**
Bases: semgen.ontology_base.OntologyHelper

Ontology helper for the Ontology of Physics for Biology (OPB).

base_uri = 'http://identifiers.org/opb/OPB_'

semgen.**bqb_isPropertyOf**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_isHomologTo**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_hasTaxon**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_hasPart**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_isPartOf**
alias of semgen.bqb.make_relation.<locals>.newrelation

class semgen.**FMA**
Bases: semgen.ontology_base.OntologyHelper

Ontology helper for the Functional Model of Anatomy (FMA) ontology.

base_uri = 'http://identifiers.org/fma/FMA:'

class semgen.**GO**
Bases: semgen.ontology_base.OntologyHelper

Ontology helper for the Gene Ontology (GO).

base_uri = 'http://identifiers.org/obo.go/GO:'

cell = 'http://identifiers.org/obo.go/GO:0005623'

cytosol = 'http://identifiers.org/obo.go/GO:0005829'

semgen.**load_antimony_str**(*sb_string*)
Loads an SBML model from an Antimony string. Antimony must be installed.

Parameters **sbml** (*str*) – An Antimony string.

Returns A SemSim model constructed from the SBML model.

Return type *ModelWrapper*

semgen.**bqb_encodes**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_hasProperty**
alias of semgen.bqb.make_relation.<locals>.newrelation

class semgen.**ChEBI**
Bases: semgen.ontology_base.OntologyHelper

Ontology helper for Chemical Entities of Biological Interest (ChEBI).

base_uri = 'http://identifiers.org/chebi/CHEBI:'

classmethod make_alias(*id*)

semgen.**bqb_occursIn**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**bqb_is**
alias of semgen.bqb.make_relation.<locals>.newrelation

class semgen.**ModelWrapper**(*semsimmodel*)
Bases: object

getCellML(*base='model.xml'*)
Write out the model as CellML.

getRDF(*base='model.xml'*)
Get the model annotations as RDF.

getSBML(*base='model.xml'*)
Write out the model as SBML.

get_turtle()
Get the model annotations as Turtle RDF.

physical_entities
Get a list of all DataStructures which have a physical property or component.

semgen.**humanize**(*t*)
Substitutes an alias for the URI if one exists.

semgen.**bqb_isEncodedBy**
alias of semgen.bqb.make_relation.<locals>.newrelation

semgen.**load_cellml_str**(*cellml*)
Loads a CellML model from a file.

Parameters **filename** (*str*) – The name of the file to load.

Returns A SemSim model constructed from the CellML model.

Return type *ModelWrapper*

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index