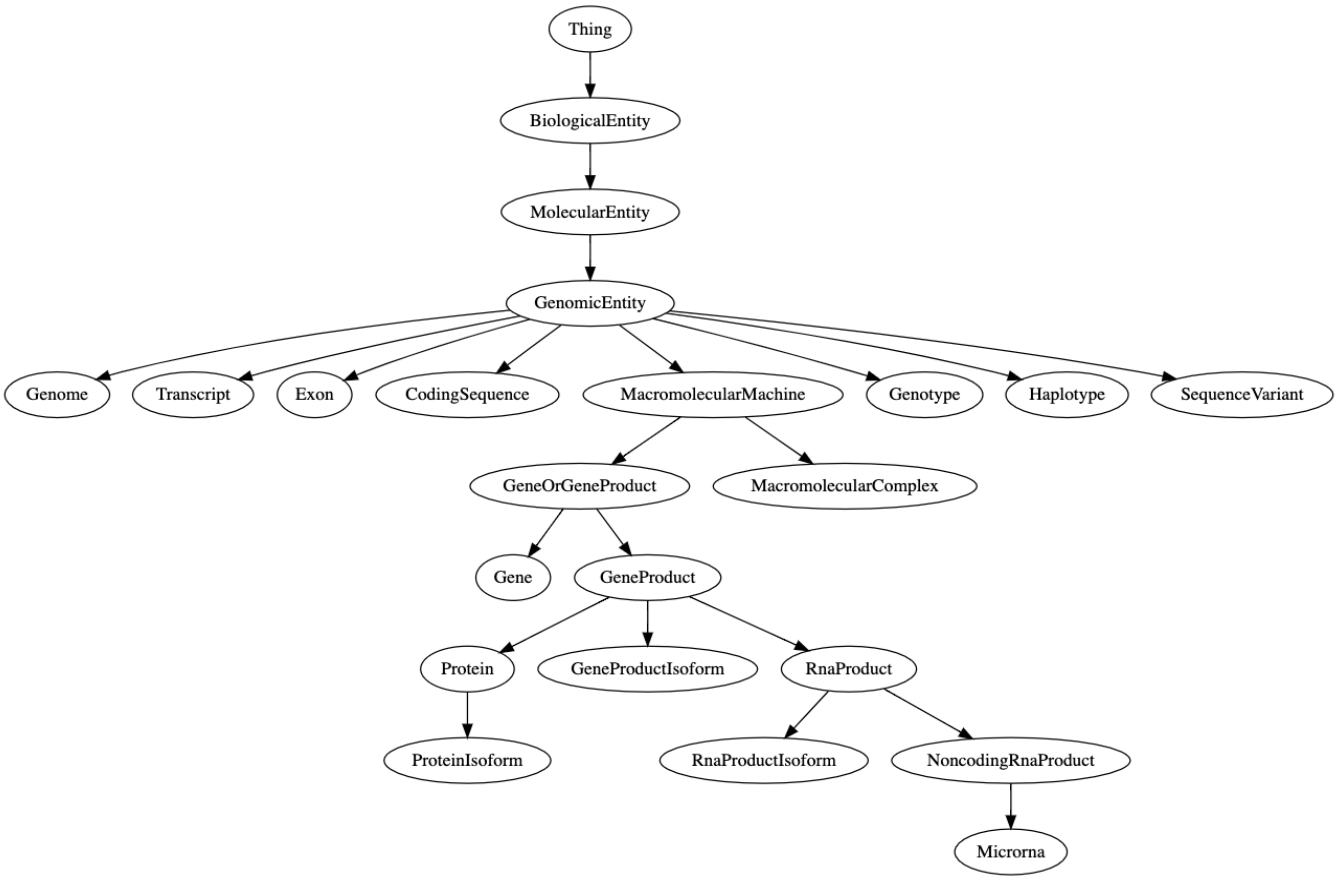
biothings$_{s}chema$

Release 0.0.1

Mar 18, 2023

Contents:

biothings_schema is a Python package for the creation, extension and exploration of the schemas defined using the Schema.org standard.



CHAPTER 1

Our Beloved Features

- Visulize schema structure
- Find parents/children of a schema class
- Find properties associated with a schema class
- Validate your schema against JSON schema
- Edit your schema

CHAPTER 2

Audience

The audience for biothings_schema python package includes developers interested in **consuming and extending** the schema defined by Schema.org.

CHAPTER 3

Schema.org and biothings_schema

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

In addition, schema.org offers the ability to extend the existing vocabulary, so that third-parties could specify additional properties or sub-types to existing types.

biothings_schema python package allows users to easily explore and consume the existing vocabularies defined in the schema.org. It also helps users to extend the schema.org vocabularies.

For more information related to schema.org, please refer to https://www.schema.org/

CHAPTER 4

Schema.org Basics

4.1 Introduction of Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

4.1.1 Why use schema.org

Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities and actions, and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences.

4.1.2 Who founded Schema.org

Founded by Google, Microsoft, Yahoo and Yandex, Schema.org vocabularies are developed by an open community process, using the public-schemaorg@w3.org mailing list and through GitHub.

4.2 Schema.org Basics

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

4.2.1 Classes and Properties

Everything in Schema.org schema is either an item type/class or a property. For example, Person is a schema.org Class. Every Class has its own properties. For example, Address is a schema.org Property.

4.2.2 Hierarchical Structure

All classes in Schema.org schema is defined in a hierarchical tree structure. Thing is the root of this structure. And all classes besides Thing will have its own parent class(es) and may also have its own child class(es). For example, Patient is a subclass(child) of Person and Person is a subclass(child) of Thing.

Moreover, properties could also have parent properties. For example, isbn is a subproperty(child) of identifier.

4.2.3 Inheritance

All classes defined in Schema.org schema will have its own specific properties. For example, Address is a property of Person. In addition, all classes will inherit the properties of its ancestors. For example, description is a property of Thing. And since Person is a subclass(child) of Thing, Person inherit the property description from its parent Thing.

4.3 Extend Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

4.3.1 Extending Mechanism

Over the years, we have experimented with a couple of different extension mechanisms (see 2011-2014 and 2014-2018 docs for details).

The primary motivation behind these models was to enable decentralized extension of the vocabulary. However, we have come to realize that for adoption, we need a simpler model, where publishers can be sure that a piece of vocabulary is indeed part of Schema.org. Until April 2019, we relied primarily on the 'hosted extensions' mechanism that used hosted subdomains of schema.org (such as bib.schema.org for the bibliography extension and autos.schema.org for the autos extension). Going forward, the content of these hosted extensions are being folded into schema.org, although each will retain an "entry point" page as before.

Terms from these hosted extensions are now considered fully part of schema.org, although they remain tagged with a label corresponding to the extension it originated from. Over a period of time, as usage of these terms gets intermingled with other terms, these labels may be dropped or simplified.

One label — 'pending' — is kept reserved for enabling us to more rapidly introduce terms into schema.org, on an experimental basis. After a period of time of being in a pending status, depending on usage, adoption, discussions etc. a term will be incorporated into the core or get dropped.

External extensions, where a third party might want to host a broadly applicable extension themselves (e.g., http://gs1.org/voc) will continue as before.

We continue to welcome improvements and additions to Schema.org, and to encourage public discussion towards this in various groups both at W3C and elsewhere. Additions to schema.org will now come in primarily via Pending. As always we place high emphasis on vocabulary that is likely to be consumed, rather than merely published.

4.3.2 How to Extend

You could extend Schema.org Schema by creating a subclass of an existing Schema.org Class or creating a subproperty of an existing Schema.org Property.

CHAPTER 5

Tutorials

5.1 Load External Schema

Currently, biothings_schema pyton package can handle 3 types of input:

1. A JSON document represented as a python dict
2. A URL to the JSON/YAML document
3. A file path to the JSON/YAML document

5.1.1 Use Python Dictionary as Input

biothings_schema python package accepts a JSON document as its input

```
In [1]: from biothings_schema import Schema

In [2]: schema = {
            "@context": {
                "bts": "http://schema.biothings.io/",
                "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
                "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
                "schema": "http://schema.org/",
                "xsd": "http://www.w3.org/2001/XMLSchema#"
            },
            "@graph": [
                {
                    "@id": "bts:BiologicalEntity",
                    "@type": "rdfs:Class",
                    "rdfs:comment": "biological entity",
                    "rdfs:label": "BiologicalEntity",
                    "rdfs:subClassOf": {
                        "@id": "schema:Thing"
                    },
```

(continues on next page)

(continued from previous page)

```
                    "schema:isPartOf": {
                        "@id": "http://schema.biothings.io"
                    }
                },
                {
                    "@id": "bts:affectsAbundanceOf",
                    "@type": "rdf:Property",
                    "rdfs:comment": "affects abundance of",
                    "rdfs:label": "affectsAbundanceOf",
                    "schema:domainIncludes": {
                        "@id": "bts:BiologicalEntity"
                    },
                    "schema:isPartOf": {
                        "@id": "http://schema.biothings.io"
                    },
                    "schema:rangeIncludes": {
                        "@id": "bts:BiologicalEntity"
                    }
                }
            ],
            "@id": "http://schema.biothings.io/#0.1"
        }
In [3]: se = Schema(schema=schema)
```

5.1.2 Use URL as Input

biothings_schema python package also accepts URL as its input. The data loaded from the URL must be either a JSON document or a YAML document.

```
In [1]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [2]: se = Schema(schema=schema_url)
```

5.1.3 Use Local File Path as Input

biothings_schema python package also accepts local file path as its input. The data loaded from the local file path must be either a JSON document or a YAML document.

```
In [1]: schema_path = '../data/schema.jsonld'

In [2]: se = Schema(schema=schema_path)
```

5.2 Visualize Schema

biothings_schema pyton package allows you to visualize your own schema.

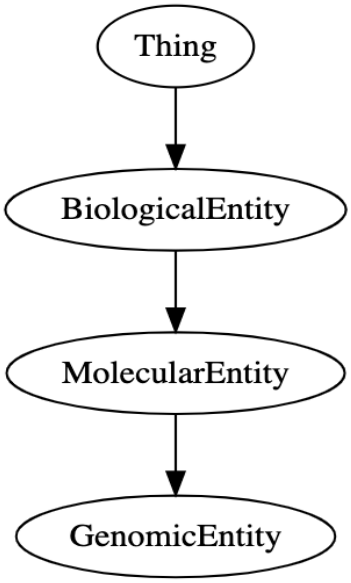
5.2.1 Visualize the Full Schema as a Tree

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: se.full_schema_graph()

# A tree structured graph of your schema would be displayed
```

5.2.2 Visulize the ancestors of a specific class

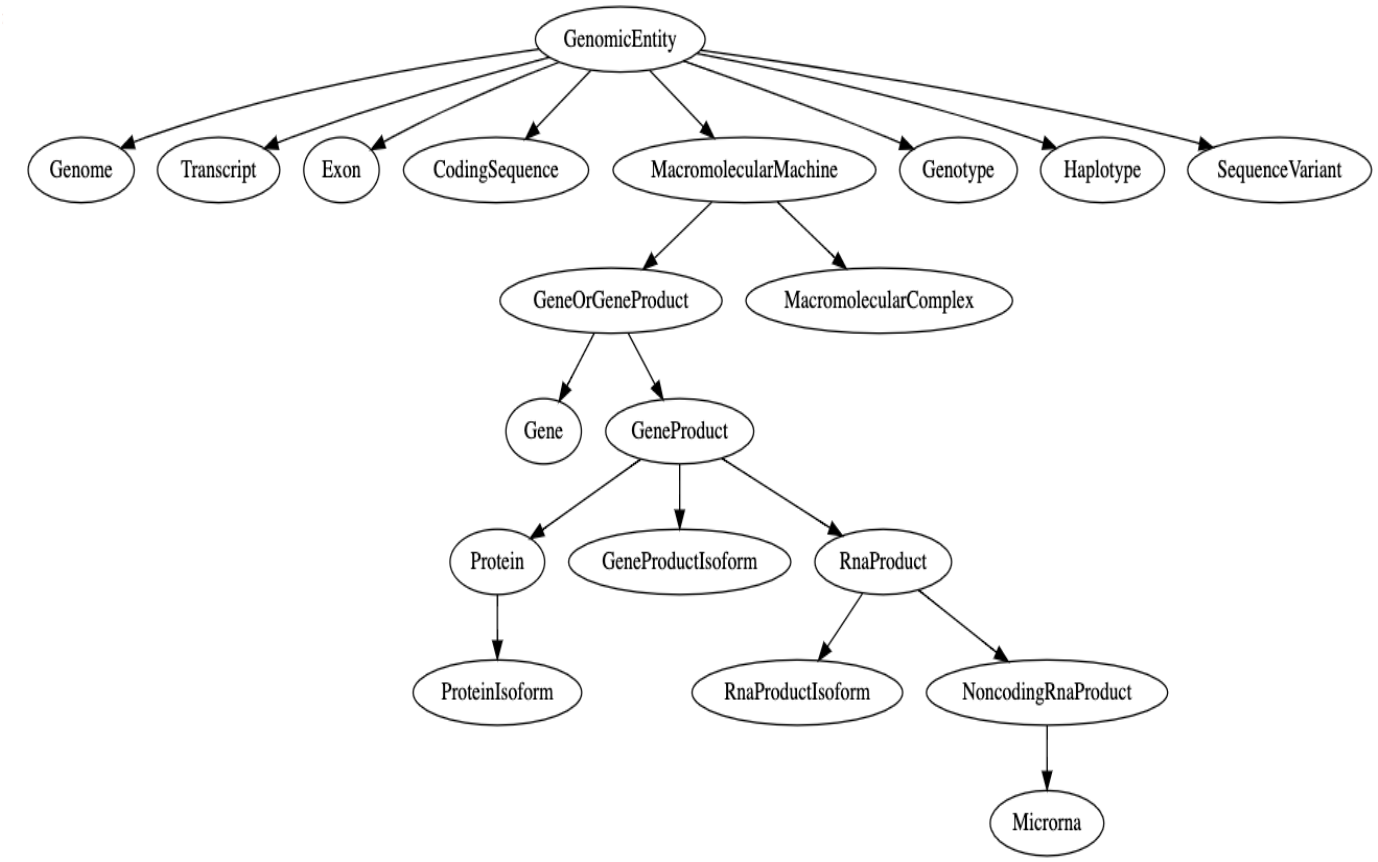
biothings_schema python package also accepts URL as its input. The data loaded from the URL must be either a JSON document or a YAML document.

```
In [1]: se.sub_schema_graph(source='GenomicEntity', include_children=False, include_
↪parents=True)
```



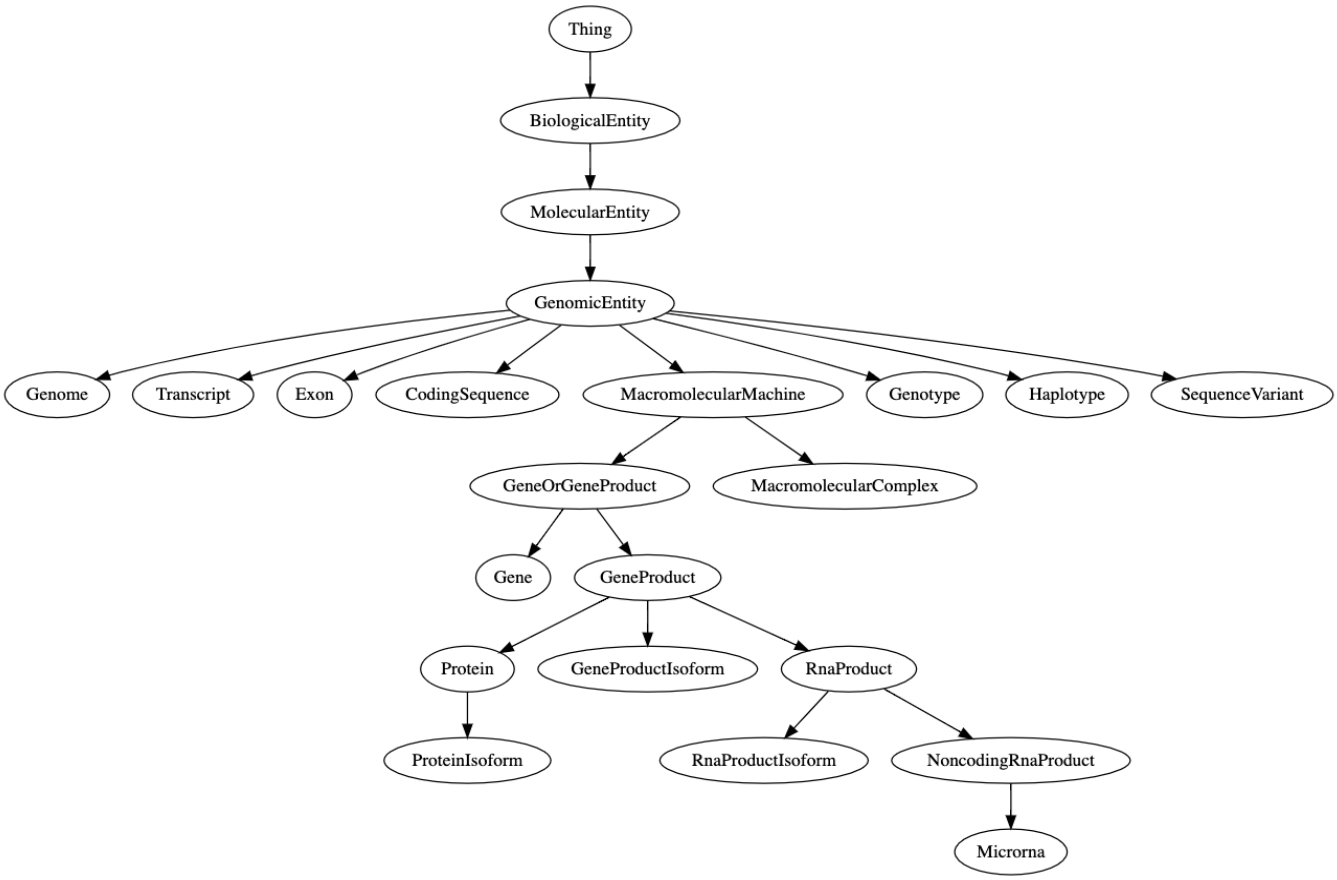
5.2.3 Visulize the descendants of a specific class

```
In [1]: se.sub_schema_graph(source='GenomicEntity', include_children=True, include_
↪parents=False)
```



5.2.4 Visulize the full ancestry of a specific class

```
In [1]: se.sub_schema_graph(source='GenomicEntity', size="9,6")
```



5.3 Explore Classes and Properties Defined in Schema

biothings_schema pyton package allows you to explore the classes (e.g. ancestors, descendants, associated properties, etc.) as well as the properties defined within the schema.

5.3.1 List all classes defined in schema

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: se.list_all_classes()

Out [4]: [SchemaClass(name=BiologicalEntity),
          SchemaClass(name=Thing),
          SchemaClass(name=OntologyClass),
          SchemaClass(name=RelationshipType),
          SchemaClass(name=GeneOntologyClass),
          ......
```

(continues on next page)

(continued from previous page)

```
          SchemaClass(name=CellLine),
          SchemaClass(name=GrossAnatomicalStructure),
          SchemaClass(name=ProteinStructure)]
```

5.3.2 Find all parents of a specific class

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("Gene")

In [5]: scls.parent_classes

Out [5]: [[SchemaClass(name=Thing),
           SchemaClass(name=BiologicalEntity),
           SchemaClass(name=MolecularEntity),
           SchemaClass(name=GenomicEntity),
           SchemaClass(name=MacromolecularMachine),
           SchemaClass(name=GeneOrGeneProduct)]]
```

5.3.3 Find all children of a specific class

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("MolecularEntity")

In [5]: scls.child_classes

Out [5]: [SchemaClass(name=ChemicalSubstance),
          SchemaClass(name=GenomicEntity),
          SchemaClass(name=GeneFamily)]
```

5.3.4 Find all descendants of a specific class

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)
```

(continues on next page)

(continued from previous page)

```
In [4]: scls = se.get_class("MolecularEntity")

In [5]: scls.descendant_classes

Out [5]: [SchemaClass(name=Metabolite),
          SchemaClass(name=ProteinIsoform),
          SchemaClass(name=GeneProduct),
          SchemaClass(name=GeneProductIsoform),
          SchemaClass(name=Genome),
          SchemaClass(name=Haplotype),
          SchemaClass(name=Transcript),
          SchemaClass(name=GeneOrGeneProduct),
          SchemaClass(name=RnaProductIsoform),
          SchemaClass(name=GeneFamily),
          SchemaClass(name=Drug),
          SchemaClass(name=RnaProduct),
          SchemaClass(name=Protein),
          SchemaClass(name=Gene),
          SchemaClass(name=GenomicEntity),
          SchemaClass(name=Microrna),
          SchemaClass(name=CodingSequence),
          SchemaClass(name=MacromolecularMachine),
          SchemaClass(name=Exon),
          SchemaClass(name=SequenceVariant),
          SchemaClass(name=MacromolecularComplex),
          SchemaClass(name=Genotype),
          SchemaClass(name=NoncodingRnaProduct),
          SchemaClass(name=ChemicalSubstance)]
```

5.3.5 Find properties associated to a specific class

Only fetch properties specifically defined for this class

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("Gene")

In [5]: scls.list_properties()

Out [5]: [{'class': 'Gene',
           'properties': [SchemaProperty(name=hgnc),
                          SchemaProperty(name=mgi),
                          SchemaProperty(name=rgd),
                          SchemaProperty(name=zfin),
                          SchemaProperty(name=flybase),
                          SchemaProperty(name=sgd),
                          SchemaProperty(name=pombase),
                          SchemaProperty(name=dictybase),
                          SchemaProperty(name=tair),
                          SchemaProperty(name=inTaxon),
```

(continues on next page)

(continued from previous page)

```
                            SchemaProperty(name=entrez),
                            SchemaProperty(name=pharos),
                            SchemaProperty(name=pharmgkb),
                            SchemaProperty(name=symbol),
                            SchemaProperty(name=omim),
                            SchemaProperty(name=umls),
                            SchemaProperty(name=unigene),
                            SchemaProperty(name=geneticallyInteractsWith),
                            SchemaProperty(name=hasGeneProduct),
                            SchemaProperty(name=hasTranscript),
                            SchemaProperty(name=geneAssociatedWithCondition)]}]
```

List all properties associated with a class (include properties for its ancestors)

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("Gene")

In [5]: scls.list_properties(class_specific=False)

Out [5]: [{'class': 'Gene',
           'properties': SchemaProperty(name=hgnc),
                         SchemaProperty(name=mgi),
                         SchemaProperty(name=rgd),
                         SchemaProperty(name=zfin),
                         SchemaProperty(name=flybase),
                         SchemaProperty(name=sgd),
                         SchemaProperty(name=pombase),
                         SchemaProperty(name=dictybase),
                         SchemaProperty(name=tair),
                         SchemaProperty(name=inTaxon),
                         SchemaProperty(name=entrez),
                         SchemaProperty(name=pharos),
                         SchemaProperty(name=pharmgkb),
                         SchemaProperty(name=symbol),
                         SchemaProperty(name=omim),
                         SchemaProperty(name=umls),
                         SchemaProperty(name=unigene),
                         SchemaProperty(name=geneticallyInteractsWith),
                         SchemaProperty(name=hasGeneProduct),
                         SchemaProperty(name=hasTranscript),
                         SchemaProperty(name=geneAssociatedWithCondition)]}]},
          {'class': 'GeneOrGeneProduct',
           'properties': [SchemaProperty(name=ensembl),
                          SchemaProperty(name=refseq),
                          SchemaProperty(name=metabolize),
                          SchemaProperty(name=targetedBy),
                          SchemaProperty(name=enablesMF),
                          SchemaProperty(name=involvedInBP),
                          SchemaProperty(name=involvedInPathway),
                          SchemaProperty(name=involvedInWikipathway),
                          SchemaProperty(name=involvedInReactomePathway),
```

(continues on next page)

(continued from previous page)

```
                        SchemaProperty(name=hasHomolog),
                        SchemaProperty(name=orthologousTo),
                        SchemaProperty(name=hasProteinStructure),
                        SchemaProperty(name=inPathwayWith),
                        SchemaProperty(name=inComplexWith),
                        SchemaProperty(name=inCellPopulationWith),
                        SchemaProperty(name=expressedIn)]},
          {'class': 'MacromolecularMachine', 'properties': []},
          {'class': 'GenomicEntity', 'properties': []},
          {'class': 'MolecularEntity',
           'properties': [SchemaProperty(name=molecularlyInteractsWith),
                        SchemaProperty(name=affectsAbundanceOf),
                        SchemaProperty(name=increasesAbundanceOf),
                        SchemaProperty(name=decreasesAbundanceOf),
                        SchemaProperty(name=affectsActivityOf),
                        .....
                        SchemaProperty(name=decreasesUptakeOf),
                        SchemaProperty(name=regulates,EntityToEntity),
                        SchemaProperty(name=biomarkerFor)]},
          {'class': 'BiologicalEntity',
           'properties': [SchemaProperty(name=hasPhenotype)]},
          {'class': 'Thing',
           'properties': [SchemaProperty(name=sameAs),
                        SchemaProperty(name=alternateName),
                        SchemaProperty(name=image),
                        SchemaProperty(name=additionalType),
                        SchemaProperty(name=name),
                        SchemaProperty(name=identifier),
                        SchemaProperty(name=subjectOf),
                        SchemaProperty(name=mainEntityOfPage),
                        SchemaProperty(name=url),
                        SchemaProperty(name=potentialAction),
                        SchemaProperty(name=description),
                        SchemaProperty(name=disambiguatingDescription)]}]
```

5.3.6 Find class usage

Find where this class has been used in the schema

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("GenomicEntity")

In [5]: scls.used_by()

Out [5]: [{'property': SchemaProperty(name=affectsExpressionOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between two molecular entities where the action or␣
↪effect of one changes the level of expression of the other within a system of␣
↪interest'},
```

(continues on next page)

(continued from previous page)

```
          {'property': SchemaProperty(name=increasesExpressionOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between two molecular entities where the action or␣
↪effect of one increases the level of expression of the other within a system of␣
↪interest'},
          {'property': SchemaProperty(name=decreasesExpressionOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between two molecular entities where the action or␣
↪effect of one decreases the level of expression of the other within a system of␣
↪interest'},
          {'property': SchemaProperty(name=affectsMutationRateOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between a molecular entity and a genomic entity␣
↪where the action or effect of the molecular entity impacts the rate of mutation of␣
↪the genomic entity within a system of interest'},
          {'property': SchemaProperty(name=increasesMutationRateOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between a molecular entity and a genomic entity␣
↪where the action or effect of the molecular entity increases the rate of mutation␣
↪of the genomic entity within a system of interest'},
          {'property': SchemaProperty(name=decreasesMutationRateOf),
           'property_used_on_class': SchemaClass(name=MolecularEntity),
           'description': 'holds between a molecular entity and a genomic entity␣
↪where the action or effect of the molecular entity decreases the rate of mutation␣
↪of the genomic entity within a system of interest'}]
```

5.3.7 Explore everything related to a class

Find all information related to a specific class, including its ancestors, descendants, associated properties as well as its usage

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: scls = se.get_class("GenomicEntity")

In [5]: scls.describe()

Out [5]:{'properties': [{'class': 'GenomicEntity', 'properties': []},
                        {'class': 'MolecularEntity',
                         'properties': [SchemaProperty(␣
↪name=molecularlyInteractsWith),
                                        SchemaProperty(name=affectsAbundanceOf),
                                        SchemaProperty(name=increasesAbundanceOf),
                                        SchemaProperty(name=decreasesAbundanceOf),
                                        ......
                                        SchemaProperty(name=decreasesUptakeOf),
                                        SchemaProperty(name=regulates,EntityToEntity),
                                        SchemaProperty(name=biomarkerFor)]},
                        {'class': 'BiologicalEntity',
                         'properties': [SchemaProperty(name=hasPhenotype)]},
```

(continues on next page)

(continued from previous page)

```
                        {'class': 'Thing',
                         'properties': [SchemaProperty(name=sameAs),
                                        SchemaProperty(name=alternateName),
                                        SchemaProperty(name=image),
                                        SchemaProperty(name=additionalType),
                                        SchemaProperty(name=name),
                                        SchemaProperty(name=identifier),
                                        SchemaProperty(name=subjectOf),
                                        SchemaProperty(name=mainEntityOfPage),
                                        SchemaProperty(name=url),
                                        SchemaProperty(name=potentialAction),
                                        SchemaProperty(name=description),
                                        SchemaProperty(name=disambiguatingDescription)]}],
         'description': 'an entity that can either be directly located on a genome (gene, transcript, exon, regulatory region) or is encoded in a genome (protein)',
         'uri': 'http://schema.biothings.io/GenomicEntity',
         'usage': [{'property': SchemaProperty(name=affectsExpressionOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between two molecular entities where the action or effect of one changes the level of expression of the other within a system of interest'},
                   {'property': SchemaProperty(name=increasesExpressionOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between two molecular entities where the action or effect of one increases the level of expression of the other within a system of interest'},
                   {'property': SchemaProperty(name=decreasesExpressionOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between two molecular entities where the action or effect of one decreases the level of expression of the other within a system of interest'},
                   {'property': SchemaProperty(name=affectsMutationRateOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between a molecular entity and a genomic entity where the action or effect of the molecular entity impacts the rate of mutation of the genomic entity within a system of interest'},
                   {'property': SchemaProperty(name=increasesMutationRateOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between a molecular entity and a genomic entity where the action or effect of the molecular entity increases the rate of mutation of the genomic entity within a system of interest'},
                   {'property': SchemaProperty(name=decreasesMutationRateOf),
                    'property_used_on_class': SchemaClass(name=MolecularEntity),
                    'description': 'holds between a molecular entity and a genomic entity where the action or effect of the molecular entity decreases the rate of mutation of the genomic entity within a system of interest'}],
         'child_classes': [SchemaClass(name=Genome),
                           SchemaClass(name=Transcript),
                           SchemaClass(name=Exon),
                           SchemaClass(name=CodingSequence),
                           SchemaClass(name=MacromolecularMachine),
                           SchemaClass(name=Genotype),
                           SchemaClass(name=Haplotype),
                           SchemaClass(name=SequenceVariant)],
         'parent_classes': [[SchemaClass(name=Thing),
                             SchemaClass(name=BiologicalEntity),
```

(continues on next page)

(continued from previous page)

```
                                SchemaClass(name=MolecularEntity)]]}
```

5.3.8 Find all parents of a specific property

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: sp = se.get_property("ensembl")

In [5]: sp.parent_properties

Out [5]: [SchemaClass(name=identifier)]
```

5.3.9 Find all children of a specific property

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: sp = se.get_property("ensembl")

In [5]: sp.child_properties

Out [5]: [SchemaClass(name=ensembl),
          SchemaClass(name=hgnc),
          SchemaClass(name=mgi),
          SchemaClass(name=rgd),
          SchemaClass(name=zfin),
          SchemaClass(name=flybase),
          ......,
          SchemaClass(name=unigene),
          SchemaClass(name=inchi),
          SchemaClass(name=inchikey),
          SchemaClass(name=rxcui),
          SchemaClass(name=smiles),
          SchemaClass(name=pubchem),
          SchemaClass(name=chembl),
          SchemaClass(name=drugbank),
          SchemaClass(name=unii)]
```

5.3.10 Explore everything related to a property

Find all information related to a specific property, including its ancestors, descendants, etc.

```
In [1]: from biothings_schema import Schema

In [2]: schema_url = 'https://raw.githubusercontent.com/data2health/schemas/biothings/
↪biothings/biothings_curie_kevin.jsonld'

In [3]: se = Schema(schema=schema_url)

In [4]: sp = se.get_property("ensembl")

In [5]: sp.describe()

Out [5]: {'child_properties': [],
          'descendant_properties': [],
          'parent_properties': [SchemaClass(name=identifier)],
          'id': 'ensembl',
          'description': 'Ensembl ID for gene, protein or transcript',
          'domain': [SchemaClass(name=GeneOrGeneProduct),
                     SchemaClass(name=Transcript)],
         'range': SchemaClass(name=Text)}
```

5.4 Validate Schema

biothings_schema pyton package allows you to validate your JSON document against the JSON schema defined in schema file.

5.4.1 Validate schema against JSON schema

```
In [1]: from biothings_schema import Schema

# load schema
In [2]: schema = {
            "@context": {
                "schema": "http://schema.org/",
                "bibo": "http://purl.org/ontology/bibo/",
                "dc": "http://purl.org/dc/elements/1.1/",
                "dcat": "http://www.w3.org/ns/dcat#",
                "dct": "http://purl.org/dc/terms/",
                "dcterms": "http://purl.org/dc/terms/",
                "dctype": "http://purl.org/dc/dcmitype/",
                "eli": "http://data.europa.eu/eli/ontology#",
                "foaf": "http://xmlns.com/foaf/0.1/",
                "owl": "http://www.w3.org/2002/07/owl#",
                "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
                "rdfa": "http://www.w3.org/ns/rdfa#",
                "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
                "skos": "http://www.w3.org/2004/02/skos/core#",
                "snomed": "http://purl.bioontology.org/ontology/SNOMEDCT/",
                "void": "http://rdfs.org/ns/void#",
                "xsd": "http://www.w3.org/2001/XMLSchema#",
                "xsd1": "hhttp://www.w3.org/2001/XMLSchema#",
                "cvisb": "https://data.cvisb.org/schema/"
            },
            "@graph": [
```

(continues on next page)

(continued from previous page)

```
                {
                    "@id": "cvisb:CvisbDataset",
                    "@type": "rdfs:Class",
                    "rdfs:comment": "A schema describing Dataset in the Center for␣
↪Viral Systems Biology",
                    "rdfs:label": "CvisbDataset",
                    "rdfs:subClassOf": {
                        "@id": "schema:Dataset"
                    },
                    "$validation": {
                        "$schema": "http://json-schema.org/draft-07/schema#",
                        "title": "Dataset",
                        "description": "A schema describing Dataset in the Center for␣
↪Viral Systems Biology",
                        "type": "object",
                        "properties": {
                            "name": {
                                "//": "the starting of inherented fields from␣
↪schema:Dataset",
                                "description": "The name of the Cvisb Dataset",
                                "type": "string"
                            },
                            "description": {
                                "description": "A description of the Cvisb Dataset",
                                "type": "string"
                            },
                            "url": {
                                "description": "URL of the item.",
                                "type": "string"
                            },
                            "sameAs": {
                                "description": "URL of a reference Web page that␣
↪unambiguously indicates the item's identity. E.g. the URL of the item's Wikipedia␣
↪page, Wikidata entry, or official website.",
                                "type": "string"
                            },
                            "keywords": {
                                "description": "Keywords or tags used to describe␣
↪this content. Multiple entries in a keywords list are typically delimited by commas.
↪",
                                "type": "array",
                                "items": {
                                    "type": "string"
                                }
                            },
                            "datePublished": {
                                "description": "Date of first broadcast/publication.",
                                "oneOf": [
                                    {
                                        "type": "string",
                                        "format": "date-time"
                                    },
                                    {
                                        "type": "string",
                                        "format": "date"
                                    }
                                ]
```

(continues on next page)

(continued from previous page)

```
                            },
                            "dateModified": {
                                "description": "The date on which the CreativeWork␣
↪was most recently modified or when the item's entry was modified within a DataFeed.
↪",
                                "oneOf": [
                                    {
                                        "type": "string",
                                        "format": "date-time"
                                    },
                                    {
                                        "type": "string",
                                        "format": "date"
                                    }
                                ]
                            },
                            "author": {
                                "description": "The author of this content or rating.␣
↪Please note that author is special in that HTML 5 provides a special mechanism for␣
↪indicating authorship via the rel tag. That is equivalent to this and may be used␣
↪interchangeably.",
                                "type": "string"
                            },
                            "publisher": {
                                "description": "The publisher of the creative work.",
                                "type": "string"
                            },
                            "version": {
                                "description": "The version of the CreativeWork␣
↪embodied by a specified resource.",
                                "type": "string"
                            },
                            "identifier": {
                                "description": "The identifier property represents␣
↪any kind of identifier for any kind of <a class=\"localLink\" href=\"http://schema.
↪org/Thing\">Thing</a>, such as ISBNs, GTIN codes, UUIDs etc. Schema.org provides␣
↪dedicated properties for representing many of these, either as textual strings or␣
↪as URL (URI) links. See <a href=\"/docs/datamodel.html#identifierBg\">background␣
↪notes</a> for more details.",
                                "type": "string"
                            },
                            "temporalCoverage": {
                                "description": "The temporalCoverage of a␣
↪CreativeWork indicates the period that the content applies to, i.e. that it␣
↪describes, either as a DateTime or as a textual string indicating a time period in
↪<a href=\"https://en.wikipedia.org/wiki/ISO_8601#Time_intervals\">ISO 8601 time␣
↪interval format</a>. In\n      the case of a Dataset it will typically indicate the␣
↪relevant time period in a precise notation (e.g. for a 2011 census dataset, the␣
↪year 2011 would be written \"2011/2012\"). Other forms of content e.g.␣
↪ScholarlyArticle, Book, TVSeries or TVEpisode may indicate their temporalCoverage␣
↪in broader terms - textually or via well-known URL.\n      Written works such as␣
↪books may sometimes have precise temporal coverage too, e.g. a work set in 1939 -␣
↪1945 can be indicated in ISO 8601 interval format format via \"1939/1945\".<br/><br/
↪>\n\nOpen-ended date ranges can be written with \"..\" in place of the end date.␣
↪For example, \"2015-11/..\" indicates a range beginning in November 2015 and with␣
↪no specified final date. This is tentative and might be updated in future when ISO␣
↪8601 is officially updated.",
```

(continues on next page)

(continued from previous page)

```
                                "type": "string",
                                "oneOf": [
                                    {
                                        "type": "string",
                                        "format": "date-time"
                                    },
                                    {
                                        "type": "string",
                                        "format": "uri"
                                    },
                                    {
                                        "type": "string"
                                    }
                                ]
                            },
                            "spatialCoverage": {
                                "description": "The spatialCoverage of a CreativeWork␣
↪indicates the place(s) which are the focus of the content. It is a subproperty of\n␣
↪     contentLocation intended primarily for more technical and detailed materials.␣
↪For example with a Dataset, it indicates\n      areas that the dataset describes: a␣
↪dataset of New York weather would have spatialCoverage which was the place: the␣
↪state of New York.",
                                "type": "string"
                            },
                            "schemaVersion": {
                                "description": "Indicates (by URL or string) a␣
↪particular version of a schema used in some CreativeWork. For example, a document␣
↪could declare a schemaVersion using an URL such as http://schema.org/version/2.0/␣
↪if precise indication of schema version was required by some application.",
                                "type": "string"
                            },
                            "sourceCode": {
                                "//": "the starting of new fields added to␣
↪schema:Dataset",
                                "type": "object",
                                "properties": {
                                    "codeRepository": {
                                        "type": "string",
                                        "format": "uri"
                                    }
                                }
                            }
                        },
                        "required": [
                            "distribution",
                            "measurementTechnique",
                            "version",
                            "author",
                            "description",
                            "name",
                            "identifier",
                        ]
                    }
                },
                {
                    "@id": "cvisb:sourceCode",
                    "@type": "rdf:Property",
```

(continues on next page)

(continued from previous page)

```
                    "rdfs:comment": "Computer programming source code. Example: Full␣
↪(compile ready) solutions, code snippet samples, scripts, templates.",
                    "rdfs:label": "sourceCode",
                    "schema:domainIncludes": [
                        {
                            "@id": "cvisb:CvisbDataset"
                        }
                    ],
                    "schema:rangeIncludes": [
                        {
                            "@id": "schema:SoftwareSourceCode"
                        }
                    ]
                }
            ]
        }

In [3]: se = Schema(schema=schema)

In [4]: json_doc = {'name': "aa",
                    'description': "bb",
                    'url': "http://ddd.com",
                    'author': "kevin",
                    'publisher': "kevin",
                    'version': "1",
                     "distribution": "33",
                     "measurementTechnique":  "11",
                    'identifier': "kk"}

In [5]: se.validate_against_schema(sample2, "https://data.cvisb.org/schema/
↪CvisbDataset")

Out [5]: The JSON document is valid
```

CHAPTER 6

Installation

To install biothings_schema, simply use pip:

Option 1 install the latest code directly from the repository:

```
pip install git+https://github.com/biothings/biothings_schema.py
↪#egg=biothings_schema.py
```