GOparser Documentation

Release 1.1.4

Florian Wagner

February 02, 2016

Contents

GOparser is a Python framework for working with Gene Ontology (GO) terms and annotations. GOparser is free and open-source software (see License).

CHAPTER 1

Main Features

- Efficient parsing of GO term information contained in `go-basic.obo` from the Gene Ontology Consortium.
- Efficient parsing of GO annotations contained in GAF files from EBI's GO Annotation Database (UniProt-GOA).
- Filtering for annotations of protein-coding genes (using `genometools`).
- Easy and fast retrieval of all genes annotated with a particular GO term, or all GO terms a particular gene is annotated with.
- Annotations are fully propagated based on `is_a` and `part_of` relations between GO terms.
- Cross-species support.
- Support for filtering annotations based on evidence code.

CHAPTER 2

Demo Notebooks

- Demo.ipynb (`download`): GOparser demo notebook

CHAPTER 3

Installation

GOparser can be installed from PyPI using pip:

```
$ pip install goparser
```

CHAPTER 4

Missing Features

- Visualizations (e.g., to show relationships between GO terms).
- Support for other GO relations, e.g., `regulates` and `has_part`.

4.1 GOparser Modules

4.1.1 Overview

The *GOParser* class provides the API for parsing and accessing GO terms and annotations. Each GO term is represented by a *GOTerm* object, and each GO anntoation is represented by a *GOAnnotation* object.

4.1.2 Module List

goparser.annotation module

class goparser.annotation.**GOAnnotation**(*gene*, *term*, *evidence*, *db_id=None*, *db_ref=None*, *with_=None*)

Bases: object

Class representing an annotation of a gene with a GO term.

For a list of annotation properties, see the GAF 2.0 file format specification.

Parameters

- **gene** (*str*) – See *gene* attribute.
- **term** (*GOTerm* object) – See *term* attribute.
- **evidence** (*str*) – See *evidence* attribute.
- **db_id** (*str, optional*) – See *db_id* attribute.
- **db_ref** (*list of str, optional*) – See *db_ref* attribute.
- **with** (*list of str, optional*) – See *with_* attribute.

gene

str

The gene that is annotated (e.g., "MYOD1").

term
GOTerm object

The GO term that the gene is annotated with.

evidence
str

The three-letter evidence code of the annotation (e.g., "IDA").

db_id
str, optional

Database Object ID of the annotation.

db_ref
list of str, optional

DB:Reference of the annotation.

with_
list of str, optional

"With" information of the annotation.

get_gaf_format()
Return the annotation as a tab-delimited string acccording to the GAF 2.0 file format.

get_pretty_format()
Return a nicely formatted string representation of the GO annotation.

get_gaf_format()
Return a GAF 2.0-compatible string representation of the annotation.

Parameters **None** –

Returns The formatted string.

Return type str

goparser.parser module

Module containing the GOParser class.

class goparser.parser.**GOParser**(*quiet=False*, *verbose=False*)
Bases: object

A class for accessing Gene Ontology (GO) term and annotation data.

This class provides functions for parsing text files describing the Gene Ontology and GO annotations, for accessing information about specific GO terms, as well as for querying the data for associations between genes and GO terms.

Parameters

- **quiet** (*bool, optional*) – If True, only warnings and errors will be reported.
- **verbose** (*bool, optional*) – If True, enable verbose logging (i.e., including debug messages). If quiet is set to True, the value of this parameter is ignored.

terms
dict [str:GOTerm]

A mapping of GO term IDs to *GOTerm* objects, each representing a single GO term. Populated by the member function *parse_ontology*.

genes
set of str

A set of all "valid" gene names. Populated by the member function *parse_annotations*. Typically, this is the set of all protein-coding genes of a particular species. GOparser ignores all annotations for genes that are not in this set.

annotations
list of GOAnnotation objects

A list of *GOAnnotation* objects, each representing a single GO annotation. Populated by the member function *parse_annotations*.

term_annotations
dict [str:list of GOAnnotation objects]

A mapping of GO term IDs to lists of *GOAnnotation* objects, with each list representing all annotations that use a particular GO term.

gene_annotations
dict [str:list of GOAnnotation objects]

A mapping of gene symbols to lists of *GOAnnotation* objects, with each list representing all annotations of a particular gene.

get_term_by_id(*id_*)
Return the term with the given term ID as a *GOTerm* object.

get_term_by_name(*name*)
Return the term with the given name as a *GOTerm* object.

get_gene_goterms(*gene*, *ancestors=False*)
Return all GO terms that the given gene is annotated with. If `ancestors` is set to True, also return all ancestor GO terms of those terms.

get_goterm_genes(*id_*, *descendants=True*)
Return all genes annotated with the GO term corresponding to the given GO term ID. If `descendants` is set to True, also return genes annotated with any descendant GO term of this term. Since annotations should be propagated down to descendant terms, this is the default behavior.

save(*ofn*, *compress=False*)
Stores the GOParser object as a `pickle` file. If `compress` is set to True, the object is stored as a gzip'ed pickle file.

load(*fn*)
Loads the GOParser object from a `pickle` file. Gzip compression is detected automatically.

Notes

The typical workflow for reading the GO annotations for a specific species looks as follows:

- Step 1) Extract a list of protein-coding genes using the script `extract_protein_coding_genes.py` from the `genometools` package. (See the GenomeTools documentation.)

- Step 2) Use the *parse_ontology* member function to parse the `go-basic.obo` file, containing the Gene Ontology. (This file can be downloaded from the website of the Gene Ontology Consortium.)

- Step 3) Use the *parse_annotations* member function to parse a gene association file (GAF), containing annotations of genes with GO terms for a specific species. The list of protein-coding genes generated

in Step 1) is used to only parse annotations for protein-coding genes. A species-specific file can be downloaded from the ftp server of the UniProt-GOA database.

Afterwards, the member functions and *get_term_by_id* and *get_term_by_name* can be used to obtain GOTerm objects containing information about individual GO terms. The member function *get_gene_goterms* can be used to obtain a list of all GO terms a particular gene is annoatated with, and the member function *get_goterm_genes* can be used to obtain a list of all genes annotated with a particular GO term.

Examples

The following example assumes that the Gene Ontology OBO file and the UniProt-GOA gene association files have been downloaded, and that a list of protein-coding genes named "protein_coding_genes_human.tsv" has been generated using the genometools Python package.

```
>>> from goparser import GOParser
>>> G = GOParser()
>>> GOParser.parse_ontology('go-basic.obo')
>>> GOParser.parse_annotations('gene_association.goa_human.gz','protein_coding_genes_human.tsv')
>>> print GOParser.get_gene_goterms('MYC')
```

clear_annotation_data()
Clear annotation data.

Parameters **None** –

Returns

Return type None

clear_data()
Clear both ontology and annotation data.

Parameters **None** –

Returns

Return type None

get_gene_goterms(*gene*, *ancestors=False*)
Return all GO terms a particular gene is annotated with.

Parameters

- **gene** (*str*) – The gene symbol of the gene.
- **ancestors** (*bool, optional*) – If set to True, also return all ancestor GO terms.

Returns The set of GO terms the gene is annotated with.

Return type set of GOTerm objects

Notes

If a gene is annotated with a particular GO term, it can also be considered annotated with all ancestors of that GO term.

get_gene_sets(*min_genes=None*, *max_genes=None*)
Return the set of annotated genes for each GO term.

Parameters

- **min_genes** (*int, optional*) – Exclude GO terms with fewer than this number of genes.
- **max_genes** (*int, optional*) – Exclude GO terms with more than this number of genes.

Returns A gene set "database" with one gene set for each GO term.

Return type GeneSetDB

get_goterm_genes(*id_*, *descendants=True*)
Return all genes that are annotated with a particular GO term.

Parameters

- **id** (*str*) – GO term ID of the GO term.
- **descendants** (*bool, optional*) – If set to False, only return genes that are directly annotated with the specified GO term. By default, also genes annotated with any descendant term are returned.

Returns

Return type Notes

get_term_by_acc(*acc*)
Get the GO term corresponding to the given GO term accession number.

Parameters **acc** (*int*) – The GO term accession number.

Returns The GO term corresponding to the given accession number.

Return type *GOTerm*

get_term_by_id(*id_*)
Get the GO term corresponding to the given GO term ID.

Parameters **id** (*str*) – A GO term ID.

Returns The GO term corresponding to the given ID.

Return type *GOTerm*

get_term_by_name(*name*)
Get the GO term with the given GO term name.

If the given name is not associated with any GO term, the function will search for it among synonyms.

Parameters **name** (*str*) – The name of the GO term.

Returns The GO term with the given name.

Return type *GOTerm*

Raises `ValueError` – If the given name is found neither among the GO term names, nor among synonyms.

parse_annotations(*annotation_file*, *genes*, *db_sel='UniProtKB'*, *select_evidence=None*, *exclude_evidence=None*, *exclude_ref=None*, *strip_species=False*, *ignore_case=False*)
Parse a GO annotation file (in GAF 2.0 format).

GO annotation files can be downloaded from the UniProt-GOA download site or from their FTP server.

Parameters

- **annotation_file** (*str or unicode*) – Path of the annotation file (in GAF 2.0 format).

- **genes** (*List (tuple, set) of str*) – List of valid gene names.
- **db_sel** (*str, optional*) – Select only annotations with this `DB` (column 1) value. If empty, disable filtering based on the `DB` value.
- **select_evidence** (*list of str, optional*) – Only include annotations with the given evidence codes. It not specified, allow all evidence codes, except for those listed in `exclude_evidence`.
- **exclude_evidence** (*list of str, optional*) – Exclude all annotations with any of the given evidence codes. If `select_evidence` is specified, this parameter is ignored. If not specified, allow all evidence codes.
- **exclude_ref** (*list of str, optional*) – Exclude all annotations with the given DB:reference (column 6). Example: `["PMID:2676709"]`. Note: This filter is currently ignored if an annotation has more than one reference.
- **strip_species** (*bool, optional*) – Undocumented.
- **ignore_case** (*bool, optional*) – Undocumented.

Returns

Return type None

parse_ontology(*fn*, *flatten=True*, *part_of_cc_only=False*)

Parse an OBO file and store GO term information.

This function needs to be called before *parse_annotations*, in order to read in the Gene Ontology terms and structure.

Parameters

- **fn** (*str*) – Path of the OBO file.
- **flatten** (*bool, optional*) – If set to False, do not generate a list of all ancestors and descendants for each GO term. Warning: Without flattining, GOparser cannot propagate GO annotations properly.
- **part_of_cc_only** (*bool, optional*) – Legacy parameter for backwards compatibility. If set to True, ignore `part_of` relations outside the `celluclar_component` domain.

Notes

The function erases all previously parsed data. The function requires the OBO file to end with a line break.

static **read_pickle**(*fn*)

Load a GOParser object from a pickle file.

The function automatically detects whether the file is compressed with gzip.

Parameters **fn** (*str*) – Path of the pickle file.

Returns The GOParser object stored in the pickle file.

Return type *GOParser*

write_pickle(*ofn*, *compress=False*)

Serialize the current GOParser object and store it in a pickle file.

Parameters

- **ofn** (*str*) – Path of the output file.

- **compress** (*bool, optional*) – Whether to compress the file using gzip.

Returns

Return type None

Notes

Compression with gzip is significantly slower than storing the file in uncompressed form.

goparser.term module

class goparser.term.**GOTerm**(*id_*, *name*, *domain*, *definition*, *is_a*, *part_of*)

Bases: object

Class representing a GO term.

This class is used by `GOParser.parse_ontology()` to store all parsed GO term data.

Parameters

- **id** (*str*) – See `id` attribute.
- **name** (*str*) – See `name` attribute.
- **definition** (*str*) – See `definition` attribute.
- **domain** (*str*) – See `domain` attribute.
- **is_a** (*List of str*) – See `is_a` attribute.
- **part_of** (*List of str*) – See `part_of` attribute.
- **children** –

id

str

The ID of the GO term.

name

str

The name of the GO term.

domain

str

The domain of the GO term (e.g., "biological_process").

definition

str

The definition (description) of the GO term.

is_a

set of str

Set of GO term IDs that this GO term is a "subtype" of.

part_of

set of str

Set of GO term IDs that this GO term is a "part" of.

ancestors
set of str

Set of GO term IDs that are "ancestors" of this GO term.

children
set of str

Set of GO term IDs that are "children" of this GO term.

parts
set of str

Set of GO term IDs that are "parts" of this GO term.

descendants
set of str

Set of GO terms IDs that are "descendants" of this GO term.

get_pretty_format(*omit_acc=False*, *max_name_length=0*, *abbreviate=True*)
Returns a formatted version of the GO term name and ID.

acc
Returns the GO term accession number (part of the ID).

static **acc2id**(*acc*)
Converts a GO term accession number to an ID.

Parameters **acc** (*int*) – A GO term accession number.

Returns The ID corresponding to the GO term accession number.

Return type str

domain_short

get_pretty_format(*include_id=True*, *max_name_length=0*, *abbreviate=True*)
Returns a nicely formatted string with the GO term information.

Parameters

- **include_id** (*bool, optional*) – Include the GO term ID.
- **max_name_length** (*int, optional*) – Truncate the formatted string so that its total length does not exceed this value.
- **abbreviate** (*bool, optional*) – Do not use abberviations (see _abbrev) to shorten the GO term name.

Returns The formatted string.

Return type str

static **id2acc**(*id_*)
Converts a GO term ID to an accession number.

Parameters **id** (*str*) – A GO term ID.

Returns The accession number corresponding to the GO term ID.

Return type int

4.2 License

4.2.1 GOparser Documentation

4.2.2 GOparser

g

W