Gilda Documentation

Release 0.5.3

Benjamin M. Gyori

Feb 19, 2021

Contents

CHAPTER 1

Gilda modules reference

1.1 API

gilda.api.**ground**(*text*, *context=None*, *organisms=None*)

Return a list of scored matches for a text to ground.

Parameters

- **text** (*str*) – The entity text to be grounded.
- **context** (*Optional[str]*) – Any additional text that serves as context for disambiguating the given entity text, used if a model exists for disambiguating the given text.

Returns A list of ScoredMatch objects representing the groundings.

Return type list[*gilda.grounder.ScoredMatch*]

gilda.api.**get_models**()

Return a list of entity texts for which disambiguation models exist.

Returns The list of entity texts for which a disambiguation model is available.

Return type list[str]

gilda.api.**get_names**(*db*, *id*, *status=None*, *source=None*)

Return a list of entity texts corresponding to a given database ID.

Parameters

- **db** (*str*) – The database in which the ID is an entry, e.g., HGNC.
- **id** (*str*) – The ID of an entry in the database.
- **status** (*Optional[str]*) – If given, only entity texts with the given status e.g., "synonym" are returned.
- **source** (*Optional[str]*) – If given, only entity texts from the given source e.g., "uniprot" are returned.

1.2 Grounder

class gilda.grounder.**Grounder**(*terms=None*)
Bases: object

Class to look up and ground query texts in a terms file.

Parameters **terms** (*str or dict or None*) – Specifies the grounding terms that should be loaded in the Grounder. If None, the default grounding terms are loaded from the versioned resource folder. If str, it is interpreted as a path to a grounding terms TSV file which is then loaded. If dict, it is assumed to be a grounding terms dict with normalized entity strings as keys and Term objects as values. Default: None

get_models()
Return a list of entity texts for which disambiguation models exist.

Returns The list of entity texts for which a disambiguation model is available.

Return type list[str]

get_names(*db*, *id*, *status=None*, *source=None*)
Return a list of entity texts corresponding to a given database ID.

Parameters

- **db** (*str*) – The database in which the ID is an entry, e.g., HGNC.
- **id** (*str*) – The ID of an entry in the database.
- **status** (*Optional[str]*) – If given, only entity texts with the given status e.g., "synonym" are returned.
- **source** (*Optional[str]*) – If given, only entity texts from the given source e.g., "uniprot" are returned.

Returns **names** – A list of entity texts corresponding to the given database/ID

Return type list[str]

ground(*raw_str*, *context=None*, *organisms=None*)
Return scored groundings for a given raw string.

Parameters

- **raw_str** (*str*) – A string to be grounded with respect to the set of Terms that the Grounder contains.
- **context** (*Optional[str]*) – Any additional text that serves as context for disambiguating the given entity text, used if a model exists for disambiguating the given text.

Returns A list of ScoredMatch objects representing the groundings sorted by decreasing score.

Return type list[*gilda.grounder.ScoredMatch*]

lookup(*raw_str*)
Return matching Terms for a given raw string.

Parameters **raw_str** (*str*) – A string to be looked up in the set of Terms that the Grounder contains.

Returns A list of Terms that are potential matches for the given string.

Return type list of Term

class gilda.grounder.**ScoredMatch**(*term*, *score*, *match*, *disambiguation=None*)
Bases: object

Class representing a scored match to a grounding term.

term
The Term that the scored match is for.

Type gilda.grounder.Term

score
The score associated with the match.

Type float

match
The Match object characterizing the match to the Term.

Type *gilda.scorer.Match*

disambiguation
Meta-information about disambiguation, when available.

Type Optional[dict]

gilda.grounder.**load_terms_file**(*terms_file*)
Load a TSV file containing terms into a lookup dictionary.

Parameters **terms_file** (*str*) – Path to a TSV terms file with columns corresponding to the serialized elements of a Term.

Returns A lookup dictionary whose keys are normalized entity texts, and values are lists of Terms with that normalized entity text.

Return type dict

1.3 Scorer

class gilda.scorer.**Match**(*query*, *ref*, *exact=None*, *space_mismatch=None*, *dash_mismatches=None*, *cap_combos=None*)
Bases: object

Class representing a match between a query and a reference string

gilda.scorer.**generate_match**(*query*, *ref*, *beginning_of_sentence=False*)
Return a match data structure based on comparing a query to a ref str.

Parameters

- **query** (*str*) – The string to be compared against a reference string.
- **ref** (*str*) – The reference string against which the incoming query string is compared.
- **beginning_of_sentence** (*bool*) – True if the query_str appears at the beginning of a sentence, relevant for how capitalization is evaluated.

Returns A Match object characterizing the match between the two strings.

Return type *Match*

gilda.scorer.**score_namespace**(*term*)
Note: this is currently not included as an explicit score term. It is just used to rank identically scored entries.

gilda.scorer.**score_string_match**(*match*)
Return a score between 0 and 1 for the goodness of a match.

This score is purely based on the relationship of the two strings and does not take the status of the reference into account.

Parameters **match** (gilda.scorer.Match) – The Match object characterizing the relationship of the query and reference strings.

Returns A match score between 0 and 1.

Return type float

1.4 Term

class gilda.term.**Term**(*norm_text*, *text*, *db*, *id*, *entry_name*, *status*, *source*, *organism=None*)
Bases: object

Represents a text entry corresponding to a grounded term.

norm_text
The normalized text corresponding to the text entry, used for lookups.

Type str

text
The text entry itself.

Type str

db
The database / name space corresponding to the grounded term.

Type str

id
The identifier of the grounded term within the database / name space.

Type str

entry_name
The standardized name corresponding to the grounded term.

Type str

status
The relationship of the text entry to the grounded term, e.g., synonym.

Type str

source
The source from which the term was obtained.

Type str

to_json()
Return the term serialized into a JSON dict.

to_list()
Return the term serialized into a list of strings.

1.5 Process

Module containing various string processing functions used for grounding.

gilda.process.**depluralize**(*word*)
Return the depluralized version of the word, along with a status flag.

Parameters **word** (*str*) – The word which is to be depluralized.

Returns

- *str* – The original word, if it is detected to be non-plural, or the depluralized version of the word.
- *str* – A status flag represeting the detected pluralization status of the word, with non_plural (e.g., BRAF), plural_oes (e.g., mosquitoes), plural_ies (e.g., antibodies), plural_es (e.g., switches), plural_cap_s (e.g., MAPKs), and plural_s (e.g., receptors).

gilda.process.**get_capitalization_pattern**(*word*, *beginning_of_sentence=False*)
Return the type of capitalization for the string.

Parameters

- **word** (*str*) – The word whose capitalization is determined.
- **beginning_of_sentence** (*Optional[bool]*) – True if the word appears at the beginning of a sentence. Default: False

Returns The capitalization pattern of the given word. Returns one of the following: sentence_initial_cap, single_cap_letter, all_caps, all_lower, initial_cap, mixed.

Return type str

gilda.process.**normalize**(*s*)
Normalize white spaces, dashes and case of a given string.

Parameters **s** (*str*) – The string to be normalized.

Returns The normalized string.

Return type str

gilda.process.**remove_dashes**(*s*)
Remove all types of dashes in the given string.

Parameters **s** (*str*) – The string in which all types of dashes should be replaced.

Returns The string from which dashes have been removed.

Return type str

gilda.process.**replace_dashes**(*s*, *rep='-'*)
Replace all types of dashes in a given string with a given replacement.

Parameters

- **s** (*str*) – The string in which all types of dashes should be replaced.
- **rep** (*Optional[str]*) – The string with which dashes should be replaced. By default, the plain ASCII dash (-) is used.

Returns The string in which dashes have been replaced.

Return type str

gilda.process.**replace_greek_latin**(*s*)
Replace Greek spelled out letters with their latin character.

gilda.process.**replace_greek_spelled_out**(*s*)
Replace Greek unicode character with latin spelled out.

gilda.process.**replace_greek_uni**(*s*)
Replace Greek spelled out letters with their unicode character.

gilda.process.**replace_whitespace**(*s*, *rep=' '*)
Replace any length white spaces in the given string with a replacement.

Parameters

- **s** (*str*) – The string in which any length whitespaces should be replaced.
- **rep** (*Optional[str]*) – The string with which all whitespace should be replaced. By default, the plain ASCII space ( ) is used.

Returns The string in which whitespaces have been replaced.

Return type str

gilda.process.**split_preserve_tokens**(*s*)
Return split words of a string including the non-word tokens.

Parameters **s** (*str*) – The string to be split.

Returns The list of words in the string including the separator tokens, typically spaces and dashes..

Return type list of str

- genindex
- modindex
- search

Python Module Index

g

Index