Practicum Documentation

Release 1

Víctor Fernández Rico

Sep 15, 2016

CONTENTS

Contents:

class `dataset.`**`Dataset`**(*new_endpoint=None*, *thread_limiter=100*)

`add_element`(*element*, *complete_list*, *complete_list_dict*, *only_uri=False*)

Add element to a list of the dataset. Avoids duplicate elements.

Parameters

- **`element`** (*`str`*) – The element that will be added to list
- **`complete_list`** (*`list`*) – The list in which will be added
- **`complete_list_dict`** (*`dict`*) – The dict which represents the list.
- **`only_uri`** (*`bool`*) – Allow load objects distincts than URI's

Returns The id on the list of the added element

Return type int

`build_levels`(*n_levels*)

Generates a simple *chain* of triplets for the desired levels

Parameters **`n_levels`** (*`int`*) – Deep of the search on wikidata graph

Returns A list of chained triplets

Return type list

`build_n_levels_query`(*n_levels=3*)

Builds a CONSTRUCT SPARQL query of the desired deep

Parameters **`n_levels`** (*`int`*) – Deep of the search on wikidata graph

Returns The desired chained query

Return type string

`execute_query`(*query*, *headers={'Accept': 'application/json'}*)

Executes a SPARQL query to the endpoint

Parameters **`query`** (*`str`*) – The SPARQL query

Returns A tuple compound of (http_status, json_or_error)

`exist_element`(*element*, *complete_list_dict*)

Check if element exists on a given list

Parameters

- **`element`** (*`str`*) – The element itself
- **`complete_list_dict`** (*`dict`*) – The dictionary to search in

Returns Wether the item was found or no

Return type bool

`extract_entity`(*entity*, *filters={'wdt-entity': True*, *'wdt-prop': True*, *'bnode': False*, *'wdt-reference': False*, *'wdt-statement': False*, *'literal': False}*)

Given an entity, returns the valid representation, ready to be saved

The filter argument allows to avoid adding elements into lists that will not be used. It is a dictionary with the shape: {'filter': bool}. The valid filters (and default) are:

- *wdt-entity* - True

•*wdt-reference* - False

•*wdt-statement* - True

•*wdt-prop* - True

•*literal* - False

•*bnode* - False

Parameters

- **`entity`** (*`dict`*) – The entity to be analyzed
- **`filters`** (*`dict`*) – A dictionary to allow filter entities

Returns The entity itself or False

`load_dataset_from_json`(*json*, *only_uri=False*)
Loads the dataset object with a JSON

The JSON structure required is: {'object': {}, 'subject': {}, 'predicate': {}}

Parameters

- **`json`** (*`list`*) – A list of dictionary parsed from JSON
- **`only_uri`** (*`bool`*) – Allow load objects distincts than URI's

`load_dataset_from_nlevels`(*nlevels*, *extra_params=''*, *only_uri=False*)
Builds a nlevels query, executes, and loads data on object

Deprecated

Parameters

- **`nlevels`** (*`int`*) – Deep of the search on wikidata graph
- **`extra_params`** (*`str`*) – Extra SPARQL instructions for the query
- **`only_uri`** (*`bool`*) – Allow load objects distincts than URI's

`load_dataset_from_query`(*query*, *only_uri=False*)
Receives a Sparql query and fills dataset object with the response

The method will execute the query itself and will call to other method to fill in the dataset object

Parameters

- **`query`** (*`str`*) – A valid SPARQL query
- **`only_uri`** (*`bool`*) – Allow load objects distincts than URI's

`load_dataset_recurrently`(*levels*, *verbose=1*)
Loads to dataset all entities with BNE ID and their relations

Due to Wikidata endpoint cann't execute queries that take long time to complete, it is necessary to consruct the dataset entity by entity, without using SPARQL CONSTRUCT. This method will start concurrently some threads to make several SPARQL SELECT queries.

Parameters

- **`levels`** (*`int`*) – The depth to get triplets related with original item
- **`verbose`** (*`int`*) – The level of verbosity. 0 is low, and 2 is high

Returns True if operation was successful

Return type bool

`load_entire_dataset`(*levels*, *where=''*, *batch=100000*, *verbose=True*)

Loads the dataset by quering to Wikidata on the desired levels

Deprecated

Parameters

- **`levels`** (`int`) – Deep of the search
- **`where`** (`str`) – Extra where statements for SPARQL query
- **`batch`** (`int`) – Number of elements returned each query
- **`verbose`** (`bool`) – True for showing all steps the method do

Returns True if operation was successful

Return type bool

`load_from_binary`(*filepath*)

Loads the dataset object from the disk

Loads this dataset object with the binary file

Parameters **`filepath`** (`str`) – The path of the binary file

Returns True if operation was successful

Return type bool

`save_to_binary`(*filepath*)

Saves the dataset object on the disk

The dataset will be saved with the required format for reading from the original library, and is prepared to be trained.

Parameters **`filepath`** (`str`) – The path of the file where should be saved

Returns True if operation was successful

Return type bool

`show`(*verbose=False*)

Show all elements of the dataset

By default prints only one line with the number of entities, relations and triplets. If verbose, prints every list. Use wisely

Parameters **`verbose`** (`bool`) – If true prints every item of all lists

`train_split`(*ratio=0.8*)

Split subs into three lists: train, valid and test

The triplets should have a specific name and size to be compatible with the original library. Splits the original triplets (self.subs) in three different lists: *train_subs*, *valid_subs* and *test_subs*. The 'ratio' param will leave that quantity for train_subs, and the rest will be a half for valid and the other half for test

Parameters **`ratio`** (`float`) – The ratio of all triplets required for *train_subs*

Returns A dictionary with splited subs

Return type dict

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX