kdap

Release 0.1.25

Oct 03, 2020

Contents

KDAP contains a collection of utilities for efficiently processing and analyzing the data of Wiki-based and QnA-based portals (eg., Wikipedia, Wikia, Stack Exchange, etc.). The function takes Knol-ML files as input. Most of the functions of this library are implemented in such a way that parallel processing can be achieved. The source code can be found on github.

CHAPTER 1

Overview / Install

This library contains a collection of utilities for efficiently processing Knol-ML database dumps. There are two important features that this module intends to address: providing standard algorithms and efficient parsing of Knol-ML dump.

1.1 Requirements

- Python 3.4 or newer
- p7zip-full

1.2 Installing kdap

Installing kdap is easily done using pip. Assuming it is installed, just run the following from the command-line:

```
# pip install kdap
```

1.3 Source Code

kdap's git repo is available on GitHub, and can be cloned using:

```
$ git clone https://github.com/descentis/kdap
$ cd kdap
```

Optionally (but suggested), make use of virtualenv:

```
$ virtualenv -p python3 venv
$ source venv/bin/activate
```

Install the requirements:

```
$ pip install -r requirements.txt
$ pip install -r test-requirements.txt
$ unzip test-repos.zip
```

and run the tests using pytest:

```
$ pytest
```

CHAPTER 2

Getting Started

Using kdap is very simple. You only need to create *knol*: this class will create the knol object which can be further used to call the kdap methods. For example, let's get the revision history for the Wikipedia article on India

```
import kdap
knol = kdap.knol()
knol.get_wiki_article('India',[output_dir])
```

This will download the full revision history of India article in KnolMl format, where output_dir is an optional argument to be provided without the brackets as a string. kdap makes the data extraction process super simple.

Sampling dataset from Wikipedia or Stack Exchage requires only a few lines of code. For example, suppose we require random five articles from each category of Wikipedia classes. The following code will suffice:

```
from random import sample
category_list = ['FA', 'GA', 'B', 'C', 'Start', 'Stub']
articles = {}
for category in category_list:
    articles[category] = sample(knol.get_wiki_article_by_class(wiki_class=category),␣
↪5)
```

With the KnolMl dataset present in local system, we can perform various analyses on it. For example, suppose we need the monthly revisions for the India article we downloaded earlier:

```
revisions = knol.get_num_instances(file_list=['India'], granularity='monthly', start=
↪'2015-07-01')
```

StackExchange data can be analysed using similar methods. For example, let's find the question to answer ratio for several StackExchange portals, specified in the list `stack_list`:

```
stack_list = ['3dprinting', 'ai', 'arduino', 'boardgames', 'chemistry', 'chess']
atoq_ratio = []
for portal in stack_list:
    knol.download_dataset(sitename='stackexchange', portal=portal)
    questions = knol.get_num_instances(dir_path=portal+'/Posts', instance_type=
↪'question')
```

(continues on next page)

(continued from previous page)

```
    answers = knol.get_num_instances(dir_path=portal+'/Posts', instance_type='answer')
    atoq_ratio.append(questions['questions']/answers['answers'])
```

CHAPTER 3

Functions Overview

This page contains the list of functions and their implementation details

3.1 Extraction Methods

The following functions download/crawl/extract the data from collaborative knowledge building portals. Currently we support only the mining of Wikipedia and Stack Exchange network. the future release will support the extraction and analysis of portals such as GitHub, Reddit, and Quora.

kdap.analysis.knol.**download_dataset**(*self*, *sitename*, ***kwargs*)

Download dataset from site

Parameters

- **sitename** (*basestring*) – Name of portal to download from e.g wikipedia, stackexchange
- ****article_list** (*str or list[str]*) – Wikipedia article(s) to download in knol-ML format
- ****category_list** (*str or list[str]*) – Single/list of wikipedia categories. When this parameter is provided, all the articles under these lists will be extracted
- ****template_list** (*str or list[str]*) – Single/list of wikipedia templates. When this parameter is provided, all the articles under these lists will be extracted
- ****destdir** (*str*) – Path to destination folder where the dataset will be downloaded
- ****wikipedia_dump** (*str*) – Path to wikipedia full dump. When provided, the articles will be extract directly from the dump
- ****download** (*bool*) – if true, the articles will be downloaded
- ****portal** (*str*) – stackexchange portal name. Provided when sitename='stackexchange'

Returns

- ****final_category_list** (*list[str]*) – A list of wikipedia article names. Only when category_list is provided as argument
- ****final_template_list** (*list[str]*) – A list of wikipedia article names. Only when template_list is provided as argument

kdap.analysis.knol.**get_wiki_article**(*self*, *article_name*, ***kwargs*)
Downloads the full revision history of an article in knol-ML format

Parameters

- **article_name** (*str*) – Name of the article to download revision history for
- ****output_dir** (*str, optional*) – Output directory for generated knol-ML file

kdap.analysis.knol.**get_wiki_article_by_class**(*self*, ***kwargs*)
Query database to extract articles based on category or project name

Parameters

- ****wikiproject** (*str*) – Name of the wikiproject for which you want to extract the articles. Should not be specified with wiki_class
- ****wiki_class** (*str*) – The Wikipedia quality class for which the articles has to be extracted. Should not be specified with wikiproject

Returns ****articles** – A list of wikipedia article names.

Return type list[str]

kdap.analysis.knol.**get_instance_date**(*self*, **args*, ***kwargs*)
Retrieve the instance dates for a list of articles. Includes multiprocessing

Parameters

- ****file_list** (*list[str] or str*) – String or List of Knol-Ml article(s)' path
- ****dir_path** (*str*) – Path of the directory which contains desired knol-Ml files
- ****c_num** (*int*) – Number of parallel threads you want

Returns ****instance_date** – A dictionary with keys as articles and values as dates

Return type dictionary

kdap.analysis.knol.**get_pageviews**(*self*, *site_name*, **args*, ***kwargs*)
Get pageviews for a particular article

Parameters

- **site_name** (*str*) – Site to get pageviews from
- ****article_name** (*str*) – Article to get pageviews for
- ****granularity** (*str*) – Granularity of pageviews data e.g. monthly
- ****start** (*str*) – Date to start counting pageviews from
- ****end** (*str*) – Date to count pageviews till

kdap.analysis.knol.**get_num_instances**(*self*, ***kwargs*)
Extract number of instances based on start and end dates

Parameters

- ****file_list** (*list[str]*) – List of Knol-Ml articles' path
- ****dir_path** (*str*) – Path of the directory which contains desired knol-Ml files

- ****c_num** (*int*) – Number of parallel threads you want
- ****granularity** (*str*) – Retrieve the instances monthly or yearly
- ****start** (*str*) – Start date in YYYY-MM-DD format
- ****end** (*str*) – End date in YYYY-MM-DD format

Returns **revisionLength** – A dictionary with keys as articles and values as instances

Return type dict

kdap.analysis.knol.**get_editors**(*self*, ***kwargs*)

Extract editors based on granularity. Includes parallel processing

Parameters

- ****file_list** (*list[str]*) – List of Knol-Ml articles' path
- ****dir_path** (*str*) – Path of the directory which contains desired knol-Ml files
- ****c_num** (*int*) – Number of parallel threads you want
- ****granularity** (*str*) – Retrieve the instances monthly or yearly
- ****start** (*str*) – Start date in YYYY-MM-DD format
- ****end** (*str*) – End date in YYYY-MM-DD format

Returns ****userList** – A dictionary with keys as articles and values as editor ids

Return type dictionary

kdap.analysis.knol.**get_author_edits**(*self*, ***kwargs*)

Get the edits of particular users for articles

get_author_edits(site_name,[article_list, dir_path, editor_list, all_wiki=False]) The following function is used to get the edits of each user

Parameters

- ****all_wiki** (*str*) – if *site_name* = wikipedia then setting this variable True will get all the edits of the users of article
- ****article_list** (*list[str]*) – list of file names (in knolml format)
- ****dir_path** (*str*) – path of the directory where all the files are present (in knolml format)
- ****editor_list** (*list[str]*) – list of editor usernames for which edits are required
- ****type** (*str*) – type of edit to be measured e.g. bytes, edits, sentences. bytes by default
- ****ordered_by** (*str*) – means of ordering e.g. editor, questions, answers or article

Returns **author_contrib** – A dictionary with keys as articles and values as author's contribution

Return type dict

3.2 Frame Methods

The following methods are used to extract the knolml articles in frames and use them to analyze each instance/revision/thread separately

kdap.analysis.knol.**frame**(*self*, ***kwargs*)

This method takes file names as an argument and returns the list of frame objects

Parameters

- ****file_name** (*str, optional*) – The name of the article for which the frame objects have to be created.
- ****dir_path** (*str, optional*) – The path of the directory containing the knolml files
- ****get_bulk** (*bool, optional*) – If this is true, all the frames are returned as a list instead of an iterator. This will require extra memory

`frame` returns a generator function that sequentially yields `instances` class objects to analyse each revision/thread separately

class kdap.analysis.**instances**(*instance*, *title*)

creating the instance of each object. The init function defined stores each instance's attribute which can be analyzed separately

get_bytes()

Returns the bytes detail

Returns ****bytes** – number of bytes given text has

Return type int

get_editor()

Returns the edior details

Returns ****editor** – Details related to the editor of this instance

Return type dictionary

get_score()

Returns the score details

Returns ****score** – A dictionary of score values, if available

Return type dictionary

get_tags()

Returns the tag details Works for QnA dataset

Returns ****tags** – List of tags, if available

Return type list

get_text(**args*, ***kwargs*)

Returns the text data

Parameters ****clean** (*bool, optional*) –

Returns ****text** – actual text of the instance

Return type str

get_text_stats(**args*, ***kwargs*)

Returns the email ids in the text

Parameters

- **title** (*bool, optional*) –
- **count_words** (*str, optional*) –
- **url** (*str, optional*) –

get_timestamp()

Returns the timestamp details

Returns **timestamp – Timestamp details of this instance

Return type dictionary

get_title()
Returns the title

Returns **title – Title of the Knowledge Data

Return type str

is_answer()
Returns True if the instance is an answer Works with QnA based knolml dataset

Returns **closed – Returns true if the post is an answer, if applicable

Return type bool

is_closed()
Returns True if the qna thread is closed Works with QnA based knolml dataset

Returns **closed – Returns true if the post is close, if applicable

Return type bool

is_comment()
Returns True if the instance is a comment Works with QnA based knolml dataset

Returns **closed – Returns true if the post is a comment, if applicable

Return type bool

is_question()
Returns True if the instance is a question Works with QnA based knolml dataset

Returns **closed – Returns true if the post is a question , if applicable

Return type bool

3.3 Analysis Methods

The following functions analysses the knol-ML dataset. For most of these methods, the dataset has to be provided in the argument.

kdap.analysis.knol.**get_author_similarity**(*self*, *editors*, ***kwargs*)

This method finds the similarity between the set of editors for a set of articles. Works on the returned dictionary by get_editors() method

Parameters

- ****editors** (*dictionary*) – A dictionary of editors returned by get_editors() method, granularity='daily'
- ****similarity** (*str*) – Similarity measure to be measured, e.g Jaccard

Returns **similarity – A dictionary with keys as years, months, and days and values as the similarity measures

Return type dictionary

kdap.analysis.knol.**get_local_gini_coefficient**(**args*, ***kwargs*)
This method finds the gini coefficient for each article/file provided in the argument.

Parameters

- ****file_list** (*list[str]*) – List of Knol-Ml articles' path
- ****dir_path** (*str*) – Path of the directory which contains desired knol-Ml files
- ****c_num** (*int*) – Number of parallel threads you want

Returns ****local_gini** – A dictionary with keys as articles and values as the gini coefficients

Return type dictionary

kdap.analysis.knol.**get_global_gini_coefficient**(*self*, **args*, ***kwargs*)

This method finds the global gini coefficient for a set of articles/files provided in the argument.

Parameters

- ****file_list** (*list[str]*) – List of Knol-Ml articles' path
- ****dir_path** (*str*) – Path of the directory which contains desired knol-Ml files
- ****c_num** (*int*) – Number of parallel threads you want

Returns ****global_gini** – A gini value for the given articles

Return type int

3.4 Graph Methods

The following methods are used to create the wiki graph using the wikilinks of the articles. Users can use one of these methods to create the wiki graph according to the requirement.

kdap.analysis.knol.**get_induced_graph_by_articles**(*self*, *article_names*)

Given a list of Wikipedia article names, the function returns the adjacency list of inter-wiki links

Parameters ****article_names** (*list[str]*) – List of Wikipedia article names

Returns ****adj_list** – An adjacency list of inter-wiki graph

Return type list

kdap.analysis.knol.**get_induced_graph_by_article**(*self*, *article_name*)

Given a Wikipedia article name, the function returns the adjacency list of inter-wiki links present in that article

Parameters ****article_name** (*str*) – Wikipedia article name

Returns ****adj_list** – An adjacency list of inter-wiki graph

Return type list

kdap.analysis.knol.**get_city_graph_by_country**(*self*, *country_name*)

Given a country name, the function returns the adjacency list of inter-wiki links for the cities in that country

Parameters ****country_name** (*str*) – Country name for which cities graph has to be created

Returns ****adj_list** – An adjacency list of inter-wiki graph

Return type list

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index