gydelt Documentation

Release 1.0

Mrinal Jain

Jan 05, 2018

Contents

CHAPTER 1

Welcome to gydelt

It's pretty complicated to get data from GDELT. Not only accessing the data but preprocessing the data is also a tedious task. **'gydelt'** is a wrapper that can be used to access and perform the basic preprocessing operations on the data obtained from GDELT. It has several datasets. This package is particularly for the **Global Knowledge Graph (GKG)**

View full Documentation

1.1 Why use gydelt?

The user can:

1. Collect the data -
    - by reading from a file (obtained from the GKG Exporter).
    - by querying the GDELT's GKG table hosted on BigQuery.
2. Pre-process the collected data: The data collected needs to be cleaned and processed before it can be used for analysis. The package has several functions to do the job.
3. Storage of the data.

1.2 Installation

1. Download the wheel file (gydelt-1.0-py2.py3-none-any.whl)
2. Run the command -

```
pip install gydelt-1.0-py2.py3-none-any.whl
```

Requirement :

You need Python (2.x or 3.x) to run gydelt (Python 3.x is recommended). Also, if you want to query the data from Google's BigQuery, proper authentication will be required in order to do the job.

1.3 Tutorial

A notebook that shows how to use the package and also, a sample use-case

Sample Notebook

CHAPTER 2

gydelt

2.1 GetData

class gydelt.gydelt.**GetData**
For collecting and storing the data

read_from_file(*path*, *seperator='\t'*, *parse_dates=False*, *encoding=None*, *header=0*)
Read data from a saved file into a pandas DataFrame

Parameters :

path (str) [(required)] Path of the file to be read (**Do not forget** to add the file extension)

separator (str) [default '\t'] Delimiter to use

parse_dates (list) [default False] eg - ['Date'] -> This column is parsed as a single Date column

encoding (str) [default None] Encoding to use for UTF when reading/writing (eg - 'ISO-8859-1')

header (int) [default 0] Determines what row in the data should be considered for the Heading (Column names)

If no headers are needed, pass the value of 'header' as 'None' (without the quotes)

Returns :

pandas.DataFrame A pandas DataFrame

fire_query(*project_id, fields_required=['DATE', 'Themes', 'Locations', 'Persons', 'Organizations', 'V2Tone'], is_search_criteria=False, get_stats=True, auth_file='', search_dict={}, limit=None, save_data=True*)
Fire a query on Google's BigQuery according to your search criteria and get the data in a pandas DataFrame

Parameters :

project_id (str) [(required)] The ID of the project created on Google BigQuery, with which the query is to be executed

fields_required (list) [default ['DATE', 'Themes', 'Locations', 'Persons', 'Organizations', 'V2Tone']] The names of the columns to be extracted from the GKG Table on BigQuery

is_search_criteria (boolean) [default False] If True, then the user has to specify the Search Criteria (either through the console or by passing a dictionary)

get_stats (boolean) [default True] If True, then the amount of data that is to be processed will be displayed before executing the query (only if the location of 'auth_file' is given)

auth_file (str) [default '' (empty)] The path of the authorization file received from BigQuery.

Note: If the path of authorization file is provided, stats will be displayed before executing the query. If not, a message will be displayed and the user will be asked whether or not to proceeds

---

search_dict (dict) [default {} (empty)]

Contains the Search Criteria in the following format - Keys - Column from the GKG table, where the search is to be performed Values - The keywords that are needed to be searched in the specific fields/columns

The values are divided into 3 parts - Part 1 - similar to 'Include ALL of' (boolean 'and' is applied for each keyword)

Part 2 - similar to 'Include ATLEAST ONE of' (boolean 'or' is applied for each keyword)

Part 3 - similar to 'Must NOT have ANY of' (boolean 'not' is applied for each keyword)

Delimiter for the 3 parts is semi colon

Delimiter for keywords within each part is comma

Note: If No keywords is to be added in a certain part, leave it empty (BUT, DO NOT miss the semicolons)

Example - {'Persons': 'P1;P2,P3;P4', 'Organizations': ';O1,O2;'}

```
>>> {'Locations': 'United States,China;;', 'Persons': ';;Donald Trump'}
# This would mean that the 'Locations' should have BOTH 'United States
↪' and 'China' and 'Persons' should NOT have 'Donald Trump'
```

Note: The format while taking the input via console is also the same.

First, enter the required fields/columns (delimited by semi colon)

Then, for each field, enter the Keywords in the same format as mentioned for the search_dict

---

Example

```
>>> Enter the Field(s) : Persons;Organizations
```

```
>>> Include ALL of these in Persons: sundar pichai,narendra modi
>>> Include ATLEAST ONE of these in Persons: larry page,andrew ng
>>> Include NONE of these in Persons: donald trump
```

```
>>> Include ALL of these in Organizations: google
>>> Include ATLEAST ONE of these in Organizations: allen institute for␣
↪artificial intelligence
>>> Include NONE of these in Organizations :
```

Note: If a dictionary is passed, it is **case-sensitive**. Therefore, give the values in the proper casing.

---

limit (int) [default None] The Maximum no. of rows to be returned from the result obtained by the Query.

Note: The max. size of the result is 128 MB.

If your query generates the data that exceeds 128 MB, you will need to specify the limit.

---

save_data (boolean) [default True] Will save the data in the current working directory

Returns :

pandas.DataFrame A pandas DataFrame

save_data_frame(*data_frame*, *path=None*, *seperator='\t'*, *index=False*)
Save a DataFrame (to a specified location/current working directory)

Parameters :

data_frame (pandas.DataFrame) : The DataFrame that needs to be stored (in the specified format)

path (str) [default None] The path at which the file is to be stored

It should be of the following format - <dir>/<sub-dir>/.../<filename.csv> (recommended)

Mention the file name in the path itself (along with the extension)

Note: If no path is provided, the data frame will be saved in the current working directory.

Format of file name - Result(YYYY-MM-DD HH.MM.SS).csv

---

separator (str) [default '\t'] Delimiter to use

index (boolean) [default False] If True, then the index of the DataFrame will also be stored in the file

Returns :

None A success message, along with the full path of the file.

2.2 ProcessData

class gydelt.gydelt.**ProcessData**(*data_frame*, *location='Locations'*, *person='Persons'*, *organization='Organizations'*, *tone='ToneData'*, *theme='Themes'*)

Contains wrappers to pre-process various fields of the data collected from GDELT

check_country_list()

Returns those Locations (countries) which were not present in the countries list

Returns :

list A list containing the locations for which there was no match in the default country list

clean_locations(*only_country=True*, *fillna='unknown'*)

Pre-process the 'Locations' column of the data (Extract either all details available, or just the Countries)

Parameters :

only_country (boolean) [`default True`] If True, will keep only the country names for each row in the `Locations` column

If False, will keep whatever details available (city, state or country)

fillna (str) [`default 'unknown'`] To fill the Null values (`NaN`) with the specified value

Returns :

pandas.DataFrame A pandas DataFrame (with additional fields for `Countries` and `States`, if required)

clean_persons(*fillna='unknown'*, *max_no_of_words=6*)

Filters out the `Persons` column of the data.

Only those names are kept in which the no. of words are within a certain limit

Parameters :

fillna (str) [`default 'unknown'`] To fill the Null values (`NaN`) with the specified value

max_no_of_words (int) [`default 6`] Removes all the names whose length is greater than this value from each record/row

Returns :

pandas.DataFrame A pandas DataFrame (with updated `'Persons'`)

clean_organizations(*fillna='unknown'*)

Pre-processes the `Organizations` column. Removes certain invalid Organizations

---

Note: Some Countries (eg. United States) have been mistaken as individual Organizations

This function removes those Organizations (which are actually Countries), from each record/row

---

Parameters :

fillna (str) [`default 'unknown'`] To fill the Null values (`NaN`) with the specified value

Returns :

pandas.DataFrame A pandas DataFrame (with updated `Organizations`)

seperate_tones()

Creates Separate columns for each value in `ToneData`

Returns :

pandas.DataFrame A pandas DataFrame

Note: The `ToneData` column has 7 vaules, which are converted into seperate columns in the data frame. The original `ToneData` remains intact

---

`clean_themes` (*fillna='unknown'*)

Fills the Null values (`NaN`) in the `Themes` column of data

Parameters :

fillna (str) [`default 'unknown'`] To fill the Null values (`NaN`) with the specified value

Returns :

pandas.DataFrame A pandas DataFrame (with Null values in `Themes` filled)

`flat_column` (*columns=[]*, *fillna='unknown'*)

The given list of columns are flattened (using one-hot encoding) and the resulting columns are added to the DataFrame

Parameters :

fillna (str) [default 'unknown'] To fill the Null values (`NaN`) with the specified value

Returns :

pandas.DataFrame A pandas DataFrame

Note: All the column names passed in the list `columns` are flattened (one-hot encoding is used).

The new data frame returned contains additional columns, which are the individual and unique values present in the respective columns which are required to be flattened.

`pre_process`()

A wrapper functions that does all the pre-processig. (Except - flattening)

Returns :

pandas.DataFrame A clean and processed pandas DataFrame

`save_data_frame` (*path=None*, *seperator='\t'*, *index=False*)

Save a DataFrame (to a specified location/current working directory)

Parameters :

path (str) [`default None`] The path at which the file is to be stored

It should be of the following format - `<dir>/<sub-dir>/.../<filename.csv>` (recommended)

Mention the file name in the path itself (along with the extension)

Note: If no path is provided, the data frame will be saved in the current working directory.

Format of file name - `Result(YYYY-MM-DD HH.MM.SS).csv`

---

separator (str) [`default '\t'`] Delimiter to use

index (boolean) [`default False`] If True, then the index of the DataFrame will also be stored in the file

Returns :

None A success message, along with the full path of the file.

Index