OEDA Scraper Documentation

Release .1

Open Event Data Alliance

July 07, 2014

Contents

This site hosts the documentation for the web scraper used by the Open Event Data Alliance. The scraper functions by specifying a whitelist of trusted RSS feed URLs and scraping the articles from these RSS feeds. The scraper makes use of goose in order to scrape arbitrary pages, and stores the output content in a MongoDB instance.

CHAPTER 1

Installation

You should probably create a virtual environment, but in any event doing `pip install -r requirements.txt` should do the trick. You might (probably will) have to specify something along the lines of `--allow-all-external pattern --allow-unverified pattern` for the pattern library since it gets downloaded from its homepage.

The scraper requires a running MongoDB instance to dump the scraped stories into. Make sure you have MongoDB installed and type `mongod` at the terminal to begin the instance if your install method didn't set up the MongoDB process to run automatically. MongoDB doesn't require you to prepare the collection or database ahead of time, so when you run the program it should automatically create a database called `event_scrape` with a collection called `stories`. Once you've run `python scraper.py`, you can verify that the stories are in the Mongo database by opening a new terminal window and typing *mongo*.

To interface with Mongo, enter `mongo` at the command line. From inside Mongo, type `show dbs` to verify that there's a database called `event_scrape`. Enter the database with `use event_scrape` and type `show collections` to make sure there's a `stories` collection. `db.stories.find()` will show you the first 20 entries.

CHAPTER 2

Running

After everything is installed, it's as simple as `python scraper.py`. That is assuming, of course, that you wish to use the configuration seen in the `default_config.ini` file. If not, just modify that. For the source type section of the config, the three types of sources are `wire`, `international`, and `local`. It is possible to specify any combination of those source types, with the source types separated by commas in the config file. For more information on the source types, see the Contributing page.

Contents:

2.1 Contributing

More RSS feeds are always useful. If there's something specific you want to see, just add it in and open a pull request with the source's raw XML RSS feed, a unique source ID, and a label indicating whether the source is "international" or "local."

We face a tradeoff between seeking the broadest geographic coverage we can get (meaning including every local paper we can find) and accuracy and relevance (which would lead us to include only large, well-known, and high quality news outlets). We're trying to balance the two objectives by including a third column indicating whether the source is one is a wire service, a dependable news source with solid international coverage, or a local source that may contribute extra noise to the data and may require specialized actor dictionaries. The distinction between the latter two is hazy and requires a judgement call. Eventually, these labels can be used to build event datasets that are either optimized for accuracy and stability (at the cost of sparseness), or micro-level, geographically dispersed (but noisy) coverage.

2.2 Scraper Package

2.2.1 `scraper` Module

`scraper.`**`call_scrape_func`**(*siteList*, *db_collection*, *pool_size*, *db_auth*, *db_user*, *db_pass*)

Helper function to iterate over a list of RSS feeds and scrape each.

Parameters **siteList: dictionary** :

Dictionary of sites, with a nickname as the key and RSS URL as the value.

db_collection : collection

Mongo collection to put stories

pool_size : int

Number of processes to distribute work

scraper.**get_rss**(*address*, *website*)
Function to parse an RSS feed and extract the relevant links.

Parameters **address: String.** :

Address for the RSS feed to scrape.

website: String. :

Nickname for the RSS feed being scraped.

Returns **results** : pattern.web.Results.

Object containing data on the parsed RSS feed. Each item represents a unique entry in the RSS feed and contains relevant information such as the URL and title of the story.

scraper.**parse_config**()
Function to parse the config file.

scraper.**parse_results**(*rss_results*, *website*, *db_collection*)
Function to parse the links drawn from an RSS feed.

Parameters **rss_results: pattern.web.Results.** :

Object containing data on the parsed RSS feed. Each item represents a unique entry in the RSS feed and contains relevant information such as the URL and title of the story.

website: String. :

Nickname for the RSS feed being scraped.

db_collection: pymongo Collection. :

Collection within MongoDB that in which results are stored.

scraper.**scrape_func**(*website*, *address*, *COLL*, *db_auth*, *db_user*, *db_pass*)
Function to scrape various RSS feeds.

Parameters **website: String** :

Nickname for the RSS feed being scraped.

address: String :

Address for the RSS feed to scrape.

COLL: String :

Collection within MongoDB that holds the scraped data.

db_auth: String. :

MongoDB database that should be used for user authentication.

db_user: String. :

Username for MongoDB authentication.

db_user: String. :

Password for MongoDB authentication.

2.2.2 pages_scrape Module

pages_scrape.**scrape**(*url*, *extractor*)
Function to request and parse a given URL. Returns only the "relevant" text.

Parameters **url** : String.

URL to request and parse.

extractor : Goose class instance.

An instance of Goose that allows for parsing of content.

Returns **text** : String.

Parsed text from the specified website.

meta : String.

Parsed meta description of an article. Usually equivalent to the lede.

2.2.3 `mongo_connection` Module

`mongo_connection.`**`add_entry`**(*collection*, *text*, *title*, *url*, *date*, *website*)

Function that creates the dictionary of content to add to a MongoDB instance, checks whether a given URL is already in the database, and inserts the new content into the database.

Parameters **collection** : pymongo Collection.

Collection within MongoDB that in which results are stored.

text : String.

Text from a given webpage.

title : String.

Title of the news story.

url : String.

URL of the webpage from which the content was pulled.

date : String.

Date pulled from the RSS feed.

website : String.

Nickname of the site from which the content was pulled.

Returns **object_id** : String

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

m

p

s

m

p

s