MedCrawler Documentation

Release latest

October 18, 2016

Contents

Contents:

CHAPTER 1

Introduction

MedCrawler is a lightweight tool to analyze the public's perception of medical concepts. It retrieves blog posts about certain topics and extracts medical terms from them in order to find relevant connections.

The tool contains three sub-modules for different parts of the workflow, which can be used in conjunction or alone.

The **Scraper** looks for blog posts according to user-specified tags. Different plugins allow to include different sources in the search.

The **Indexer** finds and exctracts the medically relevant words and returns them as MeSH terms.

Finally, the **Grapher** analyzes the occurrence of the terms in the various blog posts and presents the correlations in a network graph.

MedCrawler was developed by the Bioactivity Screening Group at the University of Helsinki (2016).

CHAPTER 2

Installation

MedCrawler is available from Github (https://github.com/a-hel/MedCrawler.git).

You also need to download the MeSH descriptors as xml (e.g. <desc2016.xml>) separately and save/unzip it to the `MedCrawler` folder (not the `/src`). It is available from the National Library of Medicine at **‘<https://www.nlm.nih.gov/mesh/download_mesh.html>‘_** .

CHAPTER 3

Tutorial

3.1 Quickstart

In the terminal, navigate to the location of the tool. Then, change to the `/src` folder.

```
>>> cd src
```

To start the program in quickstart mode, just run the shell script:

```
>>> ./MedCrawler.sh
```

or

```
>>> bash MedCrawler.sh
```

depending on your operating system.

This will start the Scraper and send the results directly to the Indexer.

You will be prompted to enter the scraping parameters.

Project: A unique project name to save your results

Plugins: List the plugins you want to use for the web search. Standard pre-installed plugin is wp (WordPress). If you want to use more than one plugin, separate their names by a whitespace.

Search terms: List the search terms for your project. Separate the search terms by a whitespace. If your search term contains any whitespaces, replace them with a plus sign (+). Example: `car+insurance motorbike`

Number of entries: List the number of entries you want to retrieve for each search term and each plugin.

Confirm the start of the program with (y)es

Note: Data for each project will be stored in the `../projects/` folder.

3.2 Command line tool

Instead of quickstart, you can use command line parameters for crawling and plotting. Behaviour depends on the first parameter, which has to be (c)rawl or (p)lot:

Crawl mode parameters (arbitrary order):

-pr, –project: Unique project identifier
-p, –plugins: Plugins to include in search, separated by whitespace *Whitespaces within the search terms need to be replaced by a plus sign (+)*

Any other input will be interpreted as search keyword or number of terms (if numeric)

```
>>> ./MedCrawler.sh c -pr Example3 2000 -p wp pain migraine headache
```

Plotting mode parameters:

-pr, –project: Unique project identifier
-mw, –minweight: Minimum weights to plot
-hl, –highlight (optional): Term to highlight in the plot. With this option, only connections with the specified term will be displayed. This is case-sensitive.

To exclude unwanted terms, prepend them with a slash ('/').

Every argument is interpreted as a category. You can define the depth of filtering by typing the first characters of the tree number. You can find the category codes on the MeSH website.

```
>>> ./MedCrawler.sh p -pr Example1 -mw 10 B01.650 B03 B04 C01 C02 D
```

Note: If you are not sure, which minweight setting will produce useful results, choose a large one (e.g. 1000). If it is too large to find any connections, you will get a summary of the weight distribution.

Note: You can also directrly call the `main.py` script with the same parameters. This way, you cannot use the quickstart mode.

CHAPTER 4

Creating your own plugins

The MedCrawler is shipped with the *WordPress* plugin, which accesses blogs hosted with WordPress through their API. While this is probably the largest repository of blogs, you might want to include other sources as well.

You can extend the functionality of the module by writing your own plugins. A plugin is a short python script that will be loaded and executed during the scraping. For security reasons, only include plugins that you fully understand.

A plugin must be saved in the `/plugins` folder. When invoked, the crawler calls the plugin's `main()` function with a list of terms and number of entries as arguments.

```
>>> plugin.main(['term1', 'term2', 'term3'], size=200)
```

The main function returns, or even better yields, the retrieved blog posts as an iterable

Save the script in the `src/plugins/` folder. Use the filename (without the `*.py` extension) to access the plugin. Feel free to share your plugin with a pull request to github.

CHAPTER 5

Examples

The `projects` folder contains 3 example of already scraped data that you can use to get acquainted with the grapher.

5.1 Example 1

This Example contains scraped terms from the keywords *virus*, *bacteria*, *infection*, *flu*, *influenza*, *common cold*, and *fever*. Make a graph including only certain MeSH categories:

```
>>> ./MedCrawler.sh  p -pr Example1 -mw 10 B01.650 B03 B04 C01 C02 D
```

5.2 Example 2

These terms come from WordPress searches for the keywords *insomnia*, *sleepless* and *sleeping disorder*. Exclude the term 'Id' with the following syntax:

```
>>> ./MedCrawler.sh  p -pr Example2 -mw 5 /Id
```

5.3 Example 3

Here, you can visualize associations with the higlighted term *Headache* through the following example:

```
>>> ./MedCrawler.sh p -pr Example3 -hl Headache -mw 10
```

CHAPTER 6

Reference

6.1 mc_scraper

main (*tags*, *n_posts*[, *plugins=('wp')*, *target="new_project"*])
Retrieve blog posts and yield them as pure ASCII.

Parameters

- **tags** (*list of str*) – Keywords to look for
- **n_posts** (*int*) – Number of posts to retrieve per keyword and plugin
- **(list of str)** (*plugins*) – Plugins to include. Plugins must be saved in the 'plugins' folder under <plugin_name>.py
- **target** (*str*) – Project name

Return type list of str

6.2 mc_indexer

build_index (*sourcefile*)
Build index based on sourcefile and return first node.

Parameters **sourcefile** (*str*) – Path and file name of the MeSH database (e.g. desc2016.xml)

Return type mc_tree.Node

traverse (*index*, *posts*)
Find indexed words from posts and return the preferred term and its tree number

Parameters

- **index** (*mc_tree.Node*) – The tree node from where to start the search
- **posts** (*list of str*) – List of all the posts in pure ASCII

Return type list of tuples of str

6.3 mc_grapher

main (*project*[, *categories=[]*, *minweight=1*, *highlight=False*, *exclude=[]*, *color_scheme="default"*, *source="terms.txt"*])
Build and show the graph.

Parameters

- **project** (*str*) – The project name
- **categories** (*list of str*) – The MeSH categories to include. If the list is empty, all categories will be included.
- **minweight** (*int*) – Minimum weight necessary for connections to be displayed.
- **highlight** (*str*) – A specific term to highlight. If false, no term will be highlighted
- **exclude** (*list of str*) – List of terms to exclude from the analysis.
- **color_scheme** (*str*) – Color scheme for the plot, not implemented
- **source** (*src*) – Name of sourcefile within project folder

Return type None

build_matrix (*res_file*[, *categories=[]*, *highlight=False*, *exclude=[]*, *color_scheme="default"*])
Build and return the correlation matrix and node labels and their colors

param res_file File name and path to load

type res_file str

param categories List of categories to include

type categories list of str

param highlight MeSH term to highlight

type highlight str

param exclude List of MeSH terms to exclude

type exclude list of str

param color_scheme Color scheme for the plot, not implemented

type color_scheme str

rtype scipy.sparse.dok_matrix, list of str, list of str

create_plot (*corr_map*, *terms*, *colors*[, *minweight=1*, *dpi=600*])
Draw plot and create metadata.

Parameters

- **corr_map** (*scipy.sparse.dok_matrix*) – Correlation matrix as returned from build_matrix()
- **terms** (*list of str*) – List of unique terms in the same order as the corrmap axes
- **colors** (*list of str*) – List of colors according to MeSH category in the same order as the corrmap axes
- **minweight** (*int*) – Minimum number of co-occurrences to draw.
- **dpi** (*int*) – DPI for plot

Return type Matplotlib.Figure, list of str, list of str

B

C

M

T