Python Task v2

Goal: To write the script in Python 3.x that can scrape a simple blog, save scrapped information about articles into PostgreSQL database and generate reports about similar articles against some criteria.

Details:

1. The script would read the criteria file in JSON format with the next fields:
    a. Structure and example data:
    ```
    {
      “cut_off_date”: “2015-02-02”,
       “number_of_words_interval”: [100, 150],
      “should_contain_words”: [“python”, “language”],
      “number_of_paragraphs_interval”: [10, 20]
    }
    ```
    b. cut_off_date - only articles issued after this date (inclusive, >=) should be collected
    c. number_of_words_interval - article should have number of words between these values to appear in results
    d. should_contain_words - article should contain these specific words to appear in results
    e. number_of_paragraphs_interval - article should have number of paragraphs between these values to appear in results

2. The script can scrape this specific root URL http://www.zazmic.com/blog-and-news/
    a. The script takes criteria file and root url as CLI arguments
        i. Example CLI call:
        python3 my-script.py scrape -c criteria.json --url http://www.zazmic.com/blog-and-news/

    b. The script traverses root URL and enters each article
        i. Do *not* use RSS
    c. An article should be analysed, and script parses article publish date: if it is lower than specified “cut_off_date” in criteria.json, then script rejects this article from processing, and moves to the next one.
    d. If article date is ok, then the script collects:
        i. all words from the main text
        ii. number of words precalculated
        iii. number of paragraphs in the main text (<p> tag)

iv. Script should take in account only alphanumeric words (i.e. /\w+/), not just space-separated
v. Script should convert words into lowercase before storing

e. Also the script collects the URL of the article

3. After scraping, the script writes data into PostgreSQL database
   a. DB setup
      i. Script uses default PostgreSQL setup
      ii. New database should be named “zazmic” and can be created manually (or by the script - BONUS 1)
      iii. Script creates tables if not exist
   b. The DB should contain 2 tables:
      i. Table “article” should contain fields:
         1. “url”,
         2. “date” (article publish date),
         3. “number_of_words”,
         4. “number_of_paragraphs”
      ii. Table “words” should contain fields:
         1. “words” (PostgreSQL array).

      iii. Also table “words” should be linked to “article”
   c. Tables should have necessary indexes (at least basic)
   d. Database records should be protected from duplication based on URL check

4. The script can generate a report about similar articles against match criteria
   a. CLI example:
      python3 my-script.py report -o filename.json -c criteria.json
   b. Data should be taken from the database.
   c. The Script takes matching criteria from the Criteria File
   d. The Script treats all criteria as required (in other words, assumes AND operator), i.e. article will appear in results only if matches all of the criteria
   e. The script fetches article data from DB against criteria, using specific queries.
   f. The script should find number of common words present in all articles from results (this is NOT the same, as should_contain_words, but should_contain_words is a subset of common words set)
   g. The script writes output into JSON file
      i. JSON Format:

```
{
   “criteria” : <copy of the criteria object from (4.b.i)>,
   “common_words” : [“w1”, “w2”,...],
    “articles”: [
      {
          “url”: <article_url>,
```

```
            "date": "2015-02-06",
            "number_of_words": <nn>,
            "number_of_paragraphs": <nn>
        },
        ....
    ]
}
```

ii. "date" - article's publication date
iii. "articles" - array of results, matching specified criteria
iv. common_words - common words list that is present in all articles from results (as in (4.e))

5. Manage Python dependencies you need in the recommended way.