Practical Assignment PR02
Document Handling, Stop Words, Basic Boolean Retrieval

Instructions

1. This assignment is a programming assignment. Your task is to implement all functionalities listed in the following sections.
2. On Moodle you will find a template ZIP file, which contains an incomplete implementation of a rudimentary information retrieval system. The remaining assignments (this one included) require you to add certain functionalities to this program by inserting your own code into the designated places. Here are some tips on how to use the template:
    a) Before you start coding, familiarize yourself with the structure and program logic of the complete template.
    b) The places where you need to add your own code are marked by lines such as:

```python
# TODO: Implement this function. (PR02)
raise NotImplementedError()
```

    c) The comments give you an idea in which assignment you need to complete which functions. For the current assignment, you only need to look at places that mention `PR02`.
    d) The `NotImplementedError` *must* be removed when you are done with the corresponding implementation, otherwise the program will terminate there. The comments *may* be removed.
    e) In theory, you do not have to edit the method `main_menu()` in `ir_system.py` at all. It is expected that this method will work in its original version, once you implemented the other incomplete code sections!
3. Furthermore, your implementation must meet a set of requirements that are listed below. Otherwise, it will *automatically* be scored as unsatisfactory, so please pay special attention to this:
    a) The usage of the template is mandatory.
    b) The program has to run without crashing and without entering infinite loops.
    c) We require your submission to be runnable using Python 3.10.
    d) Implement *all* features listed in this document! Ignoring single tasks/parts will lead to immediate failure without a second attempt.
    e) Do not change the names, signatures or type hints of the given functions and class definitions! Also refrain from changing the existing program logic! You may add more classes and functions as you see fit, but leave the original structure of the template untouched unless specifically told to!

f) Include meaningful comments in the code and choose meaningful identifier names!
g) By default, the template program aims to be completely operable via command line. Do not change this behavior. If you wish to implement a GUI, it must be optional.
h) Avoid using absolute paths or platform-specific path syntax (e.g. slashes/backslashes) in the source code! The program should be executable without problems and completely even after changing the computer or operating system!

4. Submit your source code documents as a ZIP archive via Moodle. You will find the deadline for the submission there.

5. Name the file as follows: group_*n*_p*m*.zip, where *n* is your group number and *m* is the number of the practical task sheet.

6. If you use a virtual environment, do not include the `venv` folder in your submission!

7. If your code uses packages that are not included in a default Python installation, provide a `requirements.txt`[1]

8. Understand that this task is to be done in your groups (as you registered in). All submissions will be checked for similarity. If plagiarism is confirmed, your submission will be discarded. Excessive use of tools like ChatGPT is also considered plagiarism.

Task 1 – Preparation of the document collection

1. The text file `aesopa10.txt` contains some fables (stories) by the Greek poet Aesop. Complete the function `extract_collection()` in the file `extraction.py` to split the text file into separate fables. The introductory usage notes and the table of contents in the file should be ignored by the program (but not deleted by hand). The fables are separated as follows:
   - 3 blank lines
   - title of the fable
   - 2 blank lines
   - text of the fable

2. `extract_collection()` should return a list of `Document` objects. Each object should be given the following data:
   - `document_id (int)`: number of the fable, corresponding to its position in aesopa10.txt. Your enumeration should start with 0.
   - `title (str)`: Title of the fable
   - `raw_text (str)`: Full text of the fable without line breaks
   - `terms (list)`: A list that contains all terms of the fable. Duplicates are included.

Task 2 – Stop Word Removal

1. Implement a stop word removal functionality for your documents. A list of English stop words is provided with the file `englishST.txt`[2] in the subdirectory `raw_data` in the provided template.

[1] See e. g. https://www.geeksforgeeks.org/how-to-create-requirements-txt-file-in-python/
[2] Source: http://members.unine.ch/jacques.savoy/clef/englishST.txt

2. Make sure that the processing of the terms is case-insensitive. Also remove punctuation and line breaks, but be aware of the special meaning of the apostrophe in English.
3. The program should be able to load a new stop word list via both of the following options:
    a) Loading the file `englishST.txt`
    b) Generating a stop word list from the collection's high and low frequency terms according to J. C. Crouch's method[3]
4. To do this, edit the file `cleanup.py` and implement all the contained methods! The main file `ir_system.py` should not necessarily need changes for this particular task.
5. Note that the stop word removal on `Document` objects should *not* delete the stopwords from the object's attributes `terms` or `raw_text`! Instead, the filtered term list should be saved in the document's `filtered_terms` attribute!

Note: The `nltk` package is not needed here and should not be used.

Task 3 – Simple Linear Search

1. Implement the class `LinearBooleanModel` in `models.py`!
2. The class is supposed to provide a linear Boolean search in the document collection using a *single* search term. Linear search here means that you check sequentially for each individual document in the collection whether the search term is contained.
3. Keep in mind, that the class implements the abstract class `RetrievalModel`! It is required toimplement all abstract methods from `RetrievalModel`.

Make sure that your solution considers all requirements listed in this file and upload it on Moodle until the specified deadline!

[3]See https://www.sciencedirect.com/science/article/pii/030645739090106C

FAQ

Where can I find the resources to help me get started? If you are unfamiliar with Python, you might consider going through some online tutorials first:

- `https://www.python.org/about/gettingstarted/`
- `https://www.w3schools.com/python/python_intro.asp`

After that, you should first spend some time just reading and understanding the template before you start implementing. All required IR-related knowledge is covered in the lecture slides. For further reading, please refer to the listed literature on the Moodle page.
We do *not* recommend following online tutorials on how to implement an IR system, as there is a chance that you will expose yourself to plagiarism suspicions if your code resembles online code too closely.

Is there a specific IDE or tools that I need to use? No, you may use any IDE you like, or none. Suggestions:

- Visual Studio Code (`https://code.visualstudio.com/`)
- PyCharm (`https://www.jetbrains.com/pycharm/`)
- Spyder (`https://www.spyder-ide.org/`)

Beyond a standard Python installation, you will not need specific tools.

I found an error/bug in the template code. What should I do? If you encounter mistakes/errors/bugs in the template, do not ignore the affected tasks! Write about the bug in the forum or via email to the course instructor as soon as possible.

I do not understand line XY in the template. Please ask in the forum if you have questions about aspects of the template. As the template was written by the DBIS group, there are no other references to consult.

Are there speed/efficiency/elegance requirements? The mandatory requirements are listed in this document. Otherwise, your program does not have to be particularly efficient or elegantly coded.

Can I use external libraries or frameworks? In this first assignment, no external libraries are needed, therefore please refrain from using them. If you have a particular need for a certain library, please ask for permission in the forum first.

I have existing code from the last time I took the course / from some other occasion. Can I submit that instead? No, the task requirements of previous instances of this class are not compatible with the assignments of SS2024.[4]

Before submitting, how can I check myself whether my program works reliably? We recommend that you write your own unit tests to verify the correct functioning of your code. However, this is not mandatory. For more informatiom, see: `https://docs.python.org/3/library/unittest.html`

[4]Also, if you passed in the last summer, you do not have to do the assignments again!