CS60092: Assignment 1

Inverted Index, Boolean Document Retrieval

[Deadline: 25.08.2024, 11:59 PM IST]

IMPORTANT INSTRUCTIONS

- **Plagiarism:** We will be employing strict plagiarism checking. If your code matches with another student's code, all those students whose codes match will be ***awarded zero marks*** without any evaluation. Therefore, it is your responsibility to ensure you neither copy anyone's code nor anyone is able to copy your code.
- **Code error:** If your code doesn't run or gives error while running, marks will be awarded based on the correctness of logic. If required, you might be called to meet the TAs and explain your code.
- **Python library restrictions**: You can use simple python libraries like `nltk`, `numpy`, `os`, `sys`, `collections`, `timeit`, etc. However, YOU CANNOT USE LIBRARIES like `lucene`, `elasticsearch`, or any other search API. If your code is found to use any such library, you will be ***awarded zero marks*** for this assignment without any evaluation. You also cannot use parsing libraries either for parsing the corpus and query files (You must write your own code for the same).

SUBMISSION INSTRUCTIONS

Submit the following files:

Assignment1_<ROLL_NO>_indexer.py
Assignment1_<ROLL_NO>_parser.py
Assignment1_<ROLL_NO>_bool.py
Assignment1_<ROLL_NO>_results.txt
README.txt

in a zipped folder named: **Assignment1_<ROLL_NO>.zip (or tar.gz)**

Your README.txt file should contain the following information-

1. **[Mandatory]** Mention your **Roll Number** on the first line of your README.
2. **[Mandatory]** Any specific library requirements to run your code and the specific Python version you are using.
3. **[Mandatory]** Provide details of your design, such as the vocabulary length, pre-processing pipeline, etc.
4. **[Optional]** Any other special information about your code or logic that you wish to convey.

IMPORTANT: PLEASE FOLLOW THE EXACT NAMING CONVENTION OF THE FILES AND THE SPECIFIC INSTRUCTIONS IN THE TASKS CAREFULLY. ANY DEVIATION WILL RESULT IN DEDUCTION OF MARKS. PLEASE NOTE THE EVALUATION GUIDELINES ON PAGE 5.

NOTE: YOUR Assignment1_<ROLL_NO>_results.txt FILE WILL NOT BE EVALUATED UNLESS YOUR CODE WORKS PROPERLY.

This assignment is on building a simple inverted index and using it to retrieve documents.
You have to use the Python programming language for this assignment.
The total marks for this assignment is 50.

Dataset:

The data for this assignment can be found at this link (Datasets/CISI folder):
https://drive.google.com/drive/folders/1AbXhFyBu_Qzkm6lr6dGUD836sJtwyrTH?usp=sharing

You will require two files for this assignment:

- **`CISI.ALL`:** This file contains information for 1460 documents. Each document is present in a specific format and you must parse it carefully by sequentially reading the records as follows-
    - Each document starts with the **.I** field, indicating the ID
    - Followed by the **.T** field, indicating the title
    - Followed by the **.A** field, indicating the author
    - Followed by the **.W** field, which contains the actual text of the document
    - Followed by the **.X** field, which has a list of cross-references to other documents.
      Example Document--

```
.I 1
.T
18 Editions of the Dewey Decimal Classifications
.A
Comaromi, J.P.
.W
The present study is a history of the DEWEY Decimal
Classification.  The first edition of the DDC was published
in 1876, the eighteenth edition in 1971, and future editions
will continue to appear as needed.  In spite of the DDC's
long and healthy life, however, its full story has never
been told.  There have been biographies of Dewey
that briefly describe his system, but this is the first
attempt to provide a detailed history of the work that
more than any other has spurred the growth of
librarianship in this country and abroad.
.X
1       5       1
92      1       1
262     1       1
556     1       1
1004    1       1
1024    1       1
1024    1       1
```

- **`CISI.QRY`***:* This is the main query file, containing the information for 112 queries. To parse each query, you must read, sequentially, the following records from the file.
    - Each query starts with the **.I** field, indicating the ID
    - Followed by the **.W** field, which actually contains the text of the query
      Example Query--

```
.I 1
.W
What problems and concerns are there in making up descriptive titles?
What difficulties are involved in automatically retrieving articles from
approximate titles?
What is the usual relevance of the content of articles to their titles?
```

Task A (Building Index) – use `CISI.ALL`

1. Extract the document ID in the **.I** field and extract the text in the **.W** field of each document to use as your overall document (Discard the title and cross-reference field)
2. Remove stop words, punctuation marks and perform lemmatization (without POS tags) to generate tokens from the corpus. (you can use **nltk/spaCy** library in python)
3. Build an Inverted Index (Dictionary with tokens as keys, and document IDs as postings)
4. Save your Inverted Index in the main code directory as ***model_queries_<ROLL_NO>.bin*** using Pickle (binary)
5. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_indexer.py
6. Running the file : Your code should take the path to the dataset as input and it should run in the following manner:

```
$>> python Assignment1_<ROLL_NO>_indexer.py <path to the CISI folder>
```

Task B (Query Preprocessing) – use `CISI.QRY`

1. Extract the text in the **.W** field of each query to use as your overall query. Extract the query ID in the **.I** field.
2. Remove stop words, punctuation marks and perform lemmatization (without POS tags) to generate tokens from the corpus. (you can use **nltk/spaCy** library in python)
3. Save the list of queries as:
   <query id> [TAB] <query text>
   in a .txt file in your main code directory. Name the .txt file as : ***queries_<ROLL_NO>.txt***
4. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_parser.py
5. Running the file : Your code should take the path to the query file as input and it should run as:

```
$>> python Assignment1_<ROLL_NO>_parser.py <path to the query file>
```

Task C (Boolean Retrieval)

1. In this part, you will use the inverted index and the processed query file to retrieve documents
2. Treat each query as an **AND** of the individual words. For example:
   Australian embassy bombing = Australian AND embassy AND bombing
3. Using this encoding (**AND**) and the inverted index, retrieve all the documents for that query (write your own merge routine for the lists)
4. Store the results in a file named: ***Assignment1_<ROLL_NO>_results.txt*** as a list of:
   <query id> : <space separated list of document IDs>
   Store this file in your main code directory
5. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_bool.py
6. Running the file : Your code should take the path to the saved inverted index and the query file as input and it should run in the following manner:
   ```
   $>> python Assignment1_<ROLL_NO>_bool.py <path to model> <path to query file>
   ```

EVALUATION GUIDELINES

1. Task A **[15 marks]**
    a. Extracting fields from text file: **5 marks**
    b. String preprocessing: **5 marks**
    c. Building inverted index: **5 marks**
2. Task B **[10 marks]**
3. Task C **[20 marks]**
    a. Individual term processing: **5 marks**
    b. Union of lists using AND: **5 marks**
    c. Quality of results (Results for all students should be fairly similar): **10 marks**
4. README **[5 marks]**
5. Deductions:
    a. Plagiarism: **-50 marks**
    b. Using libraries that are not allowed: **-50 marks**
    c. Not following naming conventions: **-2 marks for every violation**
    d. Small bugs in code, etc. that are beyond the overall logic of the code workflow, e.g., not following the input format specification for running code, or anything else that does not fall into the marking scheme above: **-2 marks for every violation**
