WILP IR Assignment (50)

Prajna Upadhyay

October 2023

1 Problem Statement and Datasets

In this assignment, you will implement **Vector Based Models** to rank academic research papers. Refer to the code and dataset that can be downloaded from the zip file here. The code is written and tested in `Python 3.10`. The dataset consists of 8K research papers (or documents) and 100 queries downloaded from Semantic Scholar [1]. The zip file consists of:

1. `main.py` file
2. `requirements.txt` file
3. `README.md`
4. `s2` folder with:
   (a) `s2/s2_doc.json`: All (8K) research papers in the corpus. Each research paper has 3 fields.
      i. `docno`: A unique document id for each research paper
      ii. `title`: Title of the research paper
      iii. `paperAbstract`: Abstract of the research paper

   A single paper entry looks like:

```
{
"docno": "6e4eddf4d6671c37537bb5d1c9623353b62e8531",
    "title": [
    "Duality Theorems for Finite Structures (
        Characterising Gaps and Good Characterisations)"
          ],
    "paperAbstract": [
    "We provide a correspondence between the subjects of
        duality and density in classes of finite
        relational structures. ......"
    ]
}
```

[1] https://www.semanticscholar.org/

(b) `s2/s2_query.json`: All the 100 queries. Each query has 2 fields:

i. `qid`: The unique id of the query

ii. `query`: The text of the query

Some query entries from the file are:

```
{
      "qid": "1",
      "query": "deep learning"
},
{
      "qid": "2",
      "query": "artificial intelligence"
},
{
      "qid": "3",
      "query": "information retrieval"
},
.
.
.
```

(c) `s2/s2.qrel`: The relevance values of research papers for each query is given in this file. The columns are:

i. `qid`: id of the query for which the relevance value is marked

ii. `docno`: document id of the research paper for which the relevance is marked

iii. `relevance`: Relevance of the research paper `docno` for the query `qid`. It can take values between '4' (most relevant) to '0' (least relevant) with '3', '2', and '1' in between. Few entries from the dataset are:

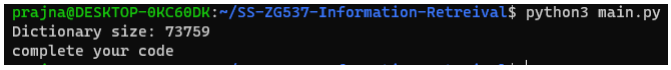
```
1 227759bc318163b2f2490690b828263f3f020cfb 2

1 373f76633cc1f6c7a421e31c989842021a52fca4 4

1 72d32c986b47d6b880dad0c3f155fe23d2939038 3

1 39f63dbdce9207b87878290c0e3983e84cfcecd9 1

1 5ca4abab527f6b0270e50548f0dea30638c9b86e 0
.
.
.
```

2 Prerequisites (Steps to be followed before starting to code)

1. Download the zip file.
2. The code has been tested with `Python 3.10`, so to ensure that you do not run into any problems related to versions, install and use `Python 3.10` for subsequent steps.
3. The top of the directory contains `main.py` and `requirements.txt`. Run the following command to first install all dependencies from there:

   ```
   pip install -r requirements.txt
   ```

4. Run `python main.py`. If everything goes well, you will see the following output:

   This means your indexes were successfully created.

5. Navigate to `s2/intermediate/` folder where the indexes are stored.

   `s2/intermediate/postings.tsv` is the inverted index file. This postings can be read in memory and used to construct vector representations.

3 What functions should I implement?

You will find an incomplete function named `rank_and_evaluate()` already defined in `main.py`. It is called from the `main` function. Complete this function to implement ranking of documents using vector based models and their evaluation for each query. Feel free to add functions or libraries of your choice. The high level steps for ranking and evaluation are:

1. Compute score for each document and query mentioned in `s2/s2.qrel`. Implement the score computation for these 4 vector-based notations:

   - `nnn.nnn`,
   - `lnn.nnn`,
   - `ntc.nnn`,
   - `ltc.nnn`

   You can concatenate the paper title and abstract to construct vector representations for each document. Feel free to remove stop-words and lemmatization of terms according to the need.

2. Rank the documents per query in decreasing order of scores. Compute time taken to rank the document per query.

3. Compute NDGC values for ranking obtained per query per notation. The perfect ranking for each query can be obtained from `s2/s2.qrel` file.

4. Report average NDCG score over the set of 100 queries for each notation.

5. Compare these NDCG scores and time taken for ranking for Vector Based Retrieval you implemented with NDCG scores and time taken for Boolean retrieval. Use the function already defined `and_query()` to rank the documents for boolean model. Assume the boolean retrieval assigns a score of '4' (highest) to every document retrieved.

However, do not delete any of the existing functions which may lead to your code not running successfully.

4 What should I submit?

Files to be submitted.

1. A zip file with only code, no indexes

2. A report.pdf: Document your findings in the report. You should analyze how each of the 4 vector based notations work on the dataset, why they give good/bad results, etc, how much time they take for ranking each query, and how do they compare to boolean retrieval models.

Checklist for zip file.

1. **main.py**: This is the python code that contains your implementation. Every `main.py` submission file should call 2 functions in its `main` function:

```
index("s2/")
rank_and_evaluate("s2/", "s2/s2_query.json", "s2/s2.qrel")
```

2. **requirements.txt**: You may include extra libraries for implementation. To be able to run your submission, we have to install those libraries too. So, all your dependencies should be written to a 'requirements.txt' file and submitted. To create your requirements.txt file, run the following command after you have finished implemention:

```
pip freeze > requirements.txt
```

5 Grading Principles

We will use a script to run everyone's `main.py` using `Python 3.10`. Scores will be awarded based on:

1. Whether or not all files are included in the submission (30% penalty is any of the files are missing)
2. Whether or not your code runs without any error (10% penalty if code runs into error)
3. If your code is found to be plagiarized, a penalty proportional to your plagiarism percentage will be imposed
4. Whether all 4 notations are implemented and whether comparison with boolean model is made ($5 \times 4 = 20$)
5. Report structure and content (30). Required sections in the report are:
   (a) Introduction to the Problem (2.5)
   (b) Vector Space Model description (2.5)
   (c) Experiments you did with Vector Space Model and Boolean Model (10)
   (d) Results obtained and Discussion, Comparison with Boolean Model (12.5)
   (e) Conclusion (2.5)
6. Bonus (20%) for improving the ranking using any other technique(s) that has been discussed apart from the 4 notations already asked for in this assignment.