Information Retrieval and Data Mining (COMP0084) Coursework 1

Deadline: March 3, 2023 at 16:00 (UK time)

Tutor: Vasileios Lampos, `v.lampos@ucl.ac.uk`

Task definition

An information retrieval model is an essential component for many applications such as search, question answering, and recommendation. In this coursework you are going to develop information retrieval models that solve the problem of passage retrieval, i.e. given a text query, return a ranked list of short texts (passages). More specifically, you are going to build a passage re-ranking system: given a candidate list of passages for a text query, you should re-rank these candidate passages based on an information retrieval model.

qid	pid	query	passage
523270	2818345	toyota of plano plano tx	DART's Red Line runs along North Central…
527433	1537731	types of dysarthria from cerebral palsy	In some cases, further testing will also be…
1113437	5194230	what is physical description of spruce	Source: *U.S. Rehab Aide Job Description…
833860	5043973	what is the most popular food in switzerland	The national currency in Switzerland is the…
1056204	2328890	who was the first steam boat operator	a relatively small usually open craft of a…

Figure 1: Sample rows from the `candidate-passages-top1000.tsv` file.

Data

Data can be downloaded from this online repository.[1] The data set consists of 3 files:

- `test-queries.tsv` is a tab separated file, where each row contains a test query identifier (`qid`) and the actual query text.
- `passage-collection.txt` is a collection of passages, one per row.
- `candidate-passages-top1000.tsv` is a tab separated file with an initial selection of at most 1000 passages for each of the queries in `test-queries.tsv`. The format of this file is `<qid pid query passage>`, where `pid` is the identifier of the passage retrieved, `query` is the query text, and `passage` is the passage text (all tab separated). The passages contained in this file are the same as the ones in `passage-collection.txt`. However, there might be

[1]Link to the data, https://www.dropbox.com/s/z15xt8og1a7bt3q/coursework-1-data.zip?dl=0

some repetitions, i.e. the same passage could have been returned for more than 1 query. Figure 1 shows some sample rows from that file.

It is important that you do **not** change the above filenames, i.e. your source code must use the above filenames, otherwise we might not be able to assess your code automatically and a penalty of 5 marks will be applied.

Coursework tasks and deliverables

Please read carefully as all deliverables (**D1** to **D12**) are described within the tasks. If a deliverable requests an answer in a specific format, follow these guidelines carefully as marking may be automated (incorrect format will be penalised and might result to 0 marks). Although we strongly suggest using Python, you could also use Java. Please make one consistent choice for this coursework. Do not submit your source code as a Jupyter Notebook or similar. **All source code must be your own.** You are not allowed to reuse any code that is available from someone or somewhere else. Do not use external functions or libraries that can solve (end-to-end) the tasks of building an inverted index, TF-IDF, the vector space model, BM25, or any of the query likelihood language models. You could use functions or libraries that enhance numeric operations, data storage, and retrieval when this does not violate the aforementioned conditions. However, do make sure that your code is not very inefficient. Very inefficient submissions will be penalised.[2] Use only unigram (1-gram) text representations to solve this coursework's tasks.[3] Write your report using the ACL LaTeX template,[4] and submit 1 `PDF` file named `report.pdf`. All plots in your report must be in vector format (e.g. `PDF`). Your report should not exceed 6 pages. In total, your submission should be made of 4 source code files (and optionally a `requirements.txt` file), 5 output `CSV` files, and 1 `PDF` file with the report. Do not deploy any internal directory structure in your submission, i.e. just submit the aforementioned 10 or 11 files. The data files should not be included in your submission. You should assume that the data files will be in the same directory as the source code (again no directory structure). If you impose a directory structure, then we will need to remove this by editing your source code to be able to assess your submission automatically. This will result in a 5 mark penalty. Your source code may generate intermediate files when it runs (e.g. to transfer an outcome of a previous step to the next one). These should be removed when submitting your solution.

Task 1 – Text statistics (30 marks). Use the data in `passage-collection.txt` for this task. Extract terms (1-grams) from the raw text. In doing so, you can also perform basic text preprocessing steps. However, you can also choose not to.

D1 Do not remove stop words in the first instance. Describe and justify any other text preprocessing choice(s), if any, and report the size of the identified index of terms (vocabulary).

Then, implement a function that counts the number of occurrences of terms in the provided data set, plot their probability of occurrence (normalised frequency) against their frequency ranking, and qualitatively justify that these terms follow Zipf's law.

[2] The expected completion time also depends on computing power. As an approximate rule of thumb, the maximum task runtime should not exceed 30 to 40 minutes when using a modern laptop (e.g. a 2020 MacBook Pro with a first generation M1 processor). The equivalent runtime for very efficient implementations could be 5 minutes or less. If we are not able to timely run your source code on our machines (i.e. the code for a task is still running after about 30 to 40 minutes), a penalty will be applied. The amount of the penalty will be decided after inspecting the source code.

[3] n-gram representations with $n > 1$ can improve performance, but are out-of-scope for this coursework.

[4] The ACL LaTeX template is available as an OverLeaf project, overleaf.com/read/crtcwgxzjskr.

As a reminder, Zipf's law for text sets $s = 1$ in the Zipfian distribution defined by

$$f(k; s, N) = \frac{k^{-s}}{\sum_{i=1}^{N} i^{-s}}, \tag{1}$$

where $f(\cdot)$ denotes the normalised frequency of a term, k denotes the term's frequency rank in our corpus (with $k = 1$ being the highest rank), N is the number of terms in our vocabulary, and s is a distribution parameter.

How does your empirical distribution compare to the actual Zipf's law distribution? Use Eq. 1 to explain and to quantify where their difference is coming from, and also compare the two in a log-log plot.

How will the difference between the two distributions be affected if we also remove stop words? Justify your answer by quantifying this difference.

D2 Submit your source code for Task 1 as a single Python or Java file titled `task1.py` or `task1.java`, respectively.

Task 2 – Inverted index (15 marks). Use `candidate-passages-top1000.tsv` for this task (unique instances of column pairs `pid` and `passage`). Using the vocabulary of terms identified in Task 1 (you will need to choose between removing or keeping stop words), build an inverted index for the collection so that you can retrieve passages in an efficient way.

D3 Provide a description of your approach to generating an inverted index, report the information that you decided to store in it, and justify this choice.
Hint: *Implementing Tasks 3 and 4 should inform this choice.*

D4 Submit the source code for Task 2 as a single Python or Java file titled `task2.py` or `task2.java`, respectively.

Task 3 – Retrieval models (35 marks). Use `test-queries.tsv` and `candidate-passages-top1000.tsv` for this task. There is no need to describe your approach in the report. You will be marked based on your output only.

D5 Extract the TF-IDF vector representations of the passages using the inverted index you have constructed. Using the IDF representation from the corpus of the passages, extract the TF-IDF representation of the queries. Use a basic vector space model with TF-IDF and cosine similarity to retrieve at most 100 passages from within the 1000 candidate passages for each query (some queries have fewer candidate passages). Store the outcomes in a file named `tfidf.csv`. Each row of `tfidf.csv` must have the following format:

```
qid,pid,score
```

where `qid` denotes the query's identifier, `pid` denotes the passage identifier, and `score` is their cosine similarity score. Note that there should be no spaces between commas. The output `.csv` files should not have headers. Strictly report the identifiers from the raw data (do not replace them with your own). `qid,pid` pairs should be ranked by similarity score in descending order, i.e. starting with the highest similarity score. You should first report the top `qid,pid` pairs for one `qid`, then move on to the next, following the same query order as in the `text-queries.tsv` file. The same file format should be used for reporting the outputs of all the retrieval or language models that you will implement as part of this coursework. The last column should report the corresponding similarity/relevance score of each model. Each output `.csv` file is expected to have 19,290 rows.

D6 Use the inverted index to also implement BM25 while setting $k_1 = 1.2$, $k_2 = 100$, and $b = 0.75$. Retrieve at most 100 passages from within the 1000 candidate passages for each test query. Store the outcomes in a file named `bm25.csv`.

D7 Submit the source code for Task 3 as a single Python or Java file titled `task3.py` or `task3.java`, respectively.

Task 4 – Query likelihood language models (20 marks). Use `test-queries.tsv` and `candidate-passages-top1000.tsv` for this task. Implement the query likelihood language model with (a) Laplace smoothing, (b) Lidstone correction with $\epsilon = 0.1$, and (c) Dirichlet smoothing with $\mu = 50$, and retrieve 100 passages from within the 1000 candidate passages for each test query. There is no need to describe your approach in implementing these language models in the report; you will be marked based on your output only. However, please note that **D11** needs to be answered in the report.

D8-10 Store the respective outcomes in the files `laplace.csv`, `lidstone.csv`, and `dirichlet.csv`. In these files, the column `score` should report the natural logarithm of the probability scores.

D11 Which language model do you expect to work better? Give a few empirical examples based on your data and experiments.

Which language models are expected to be more similar and why? Give a few empirical examples based on your data and experiments.

Comment on the value of $\epsilon = 0.1$ in the Lidstone correction. Is this a good choice and why?

If we set $\mu = 5000$ in Dirichlet smoothing, would this be a more appropriate value and why?

D12 Submit the source code for Task 4 as a single Python or Java file titled `task4.py` or `task4.java`, respectively.