# COL 764 Assignment 3 - Rank Aggregation

2023 - Version 1.0

**Deadline**

Deadline for final submission of the complete implementation, and the report on the algorithms as well as performance tuning: November 10th 2023, 11:59 PM.

### Weightage

This assignment is evaluated against 90 marks. The tentative breakup of marks is given in the end of the document.

## Instructions

1. This programming assignment is to be done by each student individually. Do not collaborate -either by sharing code, algorithm, and any other pertinent details- with each other.
2. All programs have to be written either using Python/Java/C/C++ programming languages only. Anything else requires explicit prior permission from the instructor. The machine we use to evaluate your submissions has the following versions:
   - **C**, **C++**: gcc12;
   - **Java**: java 11 (we will not use OpenJDK);
   - **Python:** version 3.7.

   We recommend you to use the same versions to avoid the use of deprecated/unsupported functions or take care to ensure required compatibility.
3. A single tar/zip of the source code has to be submitted. The zip/tar file should be structured such that
   - upon deflating all submission files should be under a directory with the student's registration number. E.g., if a student's registration number is `20XXCSXX999` then the zip submission should be named 20XXCSXX999.zip and upon deflating **all contained files** should be under a directory named ./20XXCSXX999 only (names should be in uppercase). Your submission might be rejected and not be evaluated if you do not adhere to these specifications.
   - apart from source files, the submission zip file should contain a build mechanism *if needed* (allowed build systems are Maven and Ant for Java, Makefile for C/C++). It is the responsibility of each student to ensure that it compiles and generates the necessary executables as specified. **Please include an empty file in case building is not required** *(e.g., if you are using Python)*. **Note that we will use only Ubuntu Linux machines to build and run your assignments.** So ensure that your file names, paths, argument handling, etc. are compatible.
4. You *should not* submit data files, index files, dictionaries, etc. If you are planning to use any other "special" library, please talk to the instructor first (or post on Teams).
5. **Note that there will be no deadline extensions.**

# 1 Assignment Description

In this assignment, the goal is to implement various rank aggregation techniques – **Reciprocal Rank Fusion (RRF), Bordacount, Condorcet, BayesFuse,** and **Weighted BordaFuse** – aiming for a better final ranked list by aggregating the multiple input lists. We will use data collection from LETOR 4.0. Specifically, this assignment will use the **MQ2008-agg** of LETOR 4.0 ( more detail can be found here `https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/letor-4-0/` ). It is available for download from ( `https://csciitd-my.sharepoint.com/:u:/g/personal/csz208507_iitd_ac_in/EVrKYs8ZbJ5CmGL74Ns0cDYBvi8GBqrTpXZ3pavK1WGg2Q?e=QSDwzD` ).

## 1.1 Data Description

**Document Collection:** There are about 784 unique queries in MQ2008-agg with labeled documents, with each query corresponding to one or more document IDs. The format and the description of the file are given at `https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/letor-4-0/`.

**Dataset Description**

1. **MQ2008-agg** folder contain 5 subfolders (*Fold1, Fold2, Fold3, Fold4, and Fold5*) and 6 files (*A1.txt, A2.txt, A3.txt, A4.txt, A5.txt, and agg.txt*).
2. This dataset uses a 5-fold cross-validation strategy and the 5-fold partitions are included in the package (*Fold1, Fold2, Fold3, Fold4, and Fold5*). In each fold, there are three subsets for learning: training set, validation set, and testing set, i.e., *train.txt, test.txt, and vali.txt* files.
3. *A1.txt, A2.txt, A3.txt, A4.txt, A5.txt, and agg.txt* contain the following number of unique queries and the number of lines.

| File Name | Unique Queries | Number of Lines |
|---|---|---|
| A1.txt | 157 queries | 2933 lines |
| A2.txt | 157 queries | 3635 lines |
| A3.txt | 157 queries | 3062 lines |
| A4.txt | 157 queries | 2707 lines |
| A5.txt | 156 queries | 2874 lines |
| agg.txt | 784 queries | 15211 lines |

4. **agg.txt** file is the aggregation of all the files (*A1.txt, A2.txt, A3.txt, A4.txt, and A5.txt*).

Note - Execute your algorithms using the **'agg.txt'** file and compile the outcomes in the report document.

**Format of Dataset-Files**

The format of the files is of the following form

```
0 qid:10002 1:1 2:30 3:48 4:133 5:NULL ... 25:NULL #docid = GX008-86-4444840
inc = 1 prob = 0.086622

0 qid:10002 1:NULL 2:NULL 3:NULL 4:NULL 5:NULL ... 25:NULL #docid = GX037-06
-11625428 inc = 0.0031586555555558 prob = 0.0897452

2 qid:10032 1:6 2:96 3:88 4:NULL 5:NULL ... 25:NULL #docid = GX029-35-5894638
inc = 0.0119881192468859 prob = 0.139842
```

1. The first column is the relevant label of this pair. The larger the relevance label, the more relevant

the query-document pair.

2. The second column is query id.
3. The following columns are ranks of the document in the input ranked lists. There are 25 input lists in the MQ2008-agg dataset. In the above example, 2:30 means that the document's rank is 30 in the second input list, and "NULL" means the document does not appear in a ranked list.
4. The end of the row is the comment about the pair, including the document id of the document.

# 2 Task

Given a query associated with a set of input ranked lists (25 in this case) you are supposed to implement the following rank aggregation methods to output a better final ranked list by aggregating the multiple input lists.

1. Reciprocal Rank Fusion (RRF) [2]
2. Bordacount (`http://www.opentextbookstore.com/mathinsociety/current/VotingTheory.pdf`)
3. Condorcet [3]
4. BayesFuse [1]
5. Weighted Bordafuse [1]

## 2.1 Evaluation Metrics

We will use **P@**$\{5, 10\}$ and **mAP** for evaluation.

## 2.2 Program Structure

You are expected to submit five implementations:

1. Reciprocal Rank Fusion (RRF) -
   `rrf.sh [collection-file] [output-file]`
2. Bordacount –
   `bordacount.sh [collection-file] [output-file]`
3. Condorcet –
   `condorcet.sh [collection-file] [output-file]`
4. BayesFuse –
   `bayesfuse.sh [collection-file] [output-file]`
5. Weighted BordaFuse –
   `bordafuse.sh [collection-file] [output-file]`

where,

**Scripts**

| | |
|---|---|
| `rrf`: | bash script file |
| `bordacount`: | bash script file |
| `condorcet`: | bash script file |
| `bayesfuse`: | bash script file |
| `bordafuse`: | bash script file |

**INPUT files**

| | |
|---|---|
| `collection-file`: | an input file with the specified format as above (agg.txt) |

**OUTPUT files**

| | |
|---|---|
| `output-file`: | file to which the output in the trec_eval format has to be written |

## 2.3 Output formats

We will make use of trec_eval tool for generating $P@\{5, 10\}$ and mAP scores. Therefore, it is important that you follow the exact format specification as required by the tool. To reiterate – white space is used to separate columns. The width of the columns in the format is not important, but it is important to have exactly six columns per line with at least one space between the columns.

Format of Output-File:

Lines of results are of the following form

```
1 Q0 pid1    1 2.73 runid1
1 Q0 pid2    1 2.71 runid1
1 Q0 pid3    1 2.61 runid1
1 Q0 pid4    1 2.05 runid1
1 Q0 pid5    1 1.89 runid1
```

where:

- The first column is the topic (query) number.
- The second column is currently unused and should always be "Q0".
- The third column is the identifier of the retrieved document in the context of the document ranking task.
- The fourth column is the rank the passage/document is retrieved.
- The fifth column shows the score (integer or floating point) that generated the ranking. This score must be in descending (non-increasing) order.
- The sixth column is the ID of the run you are submitting.

## 2.4 Submission Plan

All your submissions should strictly adhere to the formatting requirements given above.

- Submission of your source code on 18th October, and the final version of results.
- You should also submit a README for running your code, and a PDF document containing the implementation details as well as the resulting $P@\{5, 10\}$ and $mAP$ scores with various parameters that have been specified (preferably as a table).
  **Name them** README.{txt|md} **and** 20XXCSXX999.pdf **respectively**.

- The set of commands for evaluation of your submission roughly follows -

**Tentative Evaluation Steps (which will be used by us)**

```
$ unzip 20XXCSXX999.zip
$ cd 20XXCSXX999
$ bash build.sh
$ bash rrf.sh <arguments>
$ bash bordacount.sh <arguments>
$ bash condorcet.sh <argumnets>
$ bash bayesfuse.sh <arguments>
$ bash bordafuse.sh <arguments>
```

- Here is the link to TREC EVAL tool. `https://trec.nist.gov/trec_eval/`
- Only `BeautifulSoup`, `lxml`, `pandas`, `numpy/scipy`, `nltk`, `PorterStemmer` and libraries distributed with Python (e.g. `re`) will be available in the Python evaluation environment. You can also assume that NLTK libraries that are available through `''nltk.download('popular')''` are already downloaded. Include the source code of any imports in your submission for other languages (after getting prior permission from the instructor). No external downloads will be allowed for any language environment at runtime – i.e., there may not be internet connectivity during runtime, so tread carefully. If in doubt, please post it on the Teams channel.

### 2.5 Tentative breakup of marks assignment

In general, a submission qualifies for evaluation if and only if it adheres to the specifications given above (arguments, structure, use of external libraries, correct output format, input format adherence, etc.). Given this requirement, the marks assignment for correct implementation of:

| | |
|---|---|
| Bordacount | 10 |
| Condorcet | 15 |
| RRF | 15 |
| BordaFuse | 15 |
| BayesFuse | 20 |
| Report with algorithm details | 15 |
| Total | 90 |

## References


[1] Javed A. Aslam and Mark H. Montague. Models for metasearch. In *SIGIR*, 2001. URL: `https://www.khoury.northeastern.edu/home/jaa/CSG339.06F/resources/borda_metas.pdf`.

[2] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009. URL: `https://api.semanticscholar.org/CorpusID:12408211`.

[3] Mark H. Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *International Conference on Information and Knowledge Management*, 2002. URL: `https://api.semanticscholar.org/CorpusID:5201014`.