**Evaluation**

**1. Tìm kiếm trên TREC Test Collection**

Thực thi **TrecBatchSearch.java** dùng **trec\_index.rar** (download ở https://1drv.ms/f/s!AkRtIMFvMqP7hU18I6hDWpLCA082) để tìm kiếm trên TREC Test Collection ứng với các truy vấn (file **trec.queries.301-450** trong thư mục **trec\_queries\_qrels**) sử dụng index đã xây dựng được. Với mỗi truy vấn, thiết lập để chương trình trả về 1000 kết quả đầu tiên. Xuất kết quả cho tất cả các truy vấn ra một file. Đặt tên file kết quả là <tên thuật toán> (ví dụ: classic). Thiết lập một số tham số khi chạy file **TrecBatchSearch.java** như bên dưới:

[-index inDir] [-simfn similarity] [-field f] [-queries qfile] [-outDir outDir]

Trong đó,

inDir: Thư mục chứa các file index đã tạo

similarity: Thuật toán sử dụng

1. Trường muốn tìm kiếm (ta sẽ dùng field “contents”)

qfile: Đường dẫn đến file query

outDir: Thư mục chứa file kết quả

Mỗi dòng của file kết quả có dạng sau (mỗi dòng ứng với một kết quả):

<query id> <rank> <docno> <score> <algorithm>

Chạy chương trình với các giá trị similarity (các thuật toán) sau:

similarity Thuật toán

classic ClassicSimilarity

bm25 BM25Similarity

lmdrichlet LMDirichletSimilarity

lmjelinekmercer LMJelinekMercerSimilarity

dfr DFRSimilarity

ib IBSimilarity

Các tham số khác thì gán đường dẫn đến các file, thư mục phù hợp.

1. **Download, build, và thực thi trec\_eval**

Download **trec\_eval** ở link sau: https://github.com/usnistgov/trec\_eval/

Xem hướng dẫn về cách build **trec\_eval** ở link sau: https://github.com/usnistgov/trec\_eval/blob/master/README.windows.md

Xem cách sử dụng **trec\_eval** để đánh giá kết quả tìm kiếm ở link sau:

https://github.com/usnistgov/trec\_eval/blob/master/README (xem mục Usage)

Thực thi file trec\_eval với các tham số sau (các tham số khác thì gán đường dẫn đến các file, thư mục phù hợp):

* options = -q
* search\_results\_file = tập các file kết quả khi thực thi file **TrecBatchSearch.java** (classic, bm25, lmdrichlet, lmjelinekmercer, dfr, ib)
* trec\_eval\_file = tập các file kết quả khi thực thi file **trec\_eval.exe** (classic.eval, bm25.eval, …, ib.eval)

**Ví dụ**: trec\_eval -q "…/trec.qrels.301-450" "…/bm25" > "…/bm25.out"

Kết quả xuất ra bởi **trec\_eval** gồm các thông tin sau:

* Tổng số tài liệu trên tất cả các truy vấn num\_ret: tổng số tài liệu tìm được num\_rel: tổng số tài liệu liên quan

num\_rel\_ret: tổng số tài liệu tìm được và có liên quan

* Độ chính xác trung bình ứng với từng mức bao phủ được nội suy (Interpolated Recall - Precision Averages). Những giá trị này được dùng để vẽ đồ thị Recall-Precision.

ircl\_prn.0.00: Độ chính xác ở mức bao phủ 0.00

ircl\_prn.0.10: Độ chính xác ở mức bao phủ 0.10

...

ircl\_prn.1.00: Độ chính xác ở mức bao phủ 1.00

* Độ chính xác trung bình không nội suy (Average precision (non-interpolated)) cho mỗi truy vấn và cho tất cả các truy vấn.
* Độ chính tại tài liệu thứ thứ k (theo thứ tự trong danh sách các tài liệu tìm được)

P5: Độ chính tại tài liệu tìm được thứ 5

P10: Độ chính tại tài liệu thứ 10

...

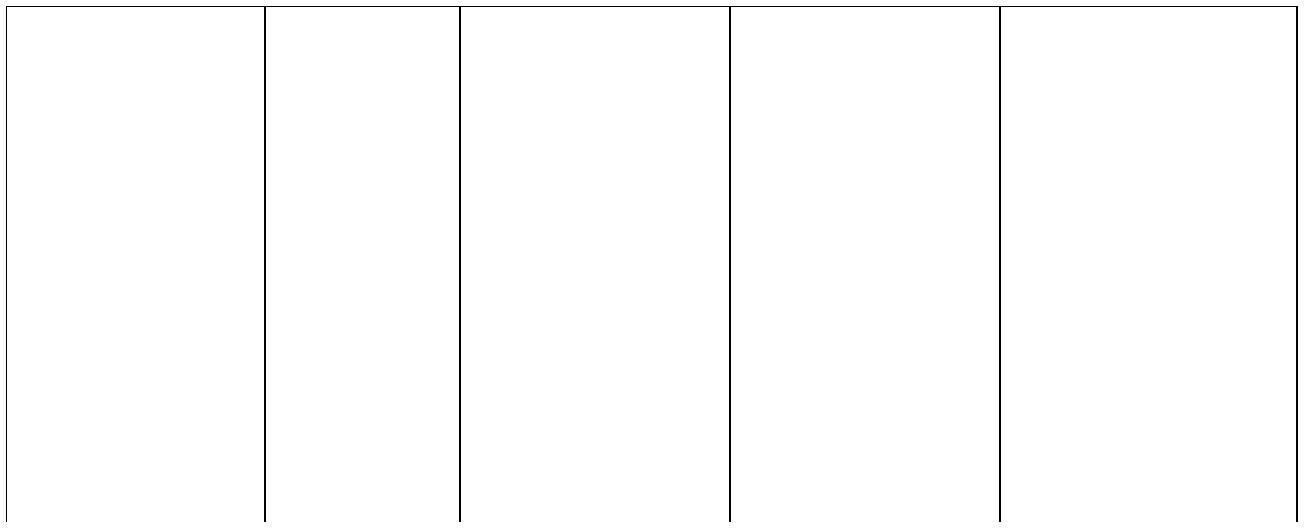
P1000: Độ chính tại tài liệu thứ 1000

* + R-prec: Độ chính xác tại tài liệu thứ R (R là tổng số tài liệu liên quan cho truy vấn)

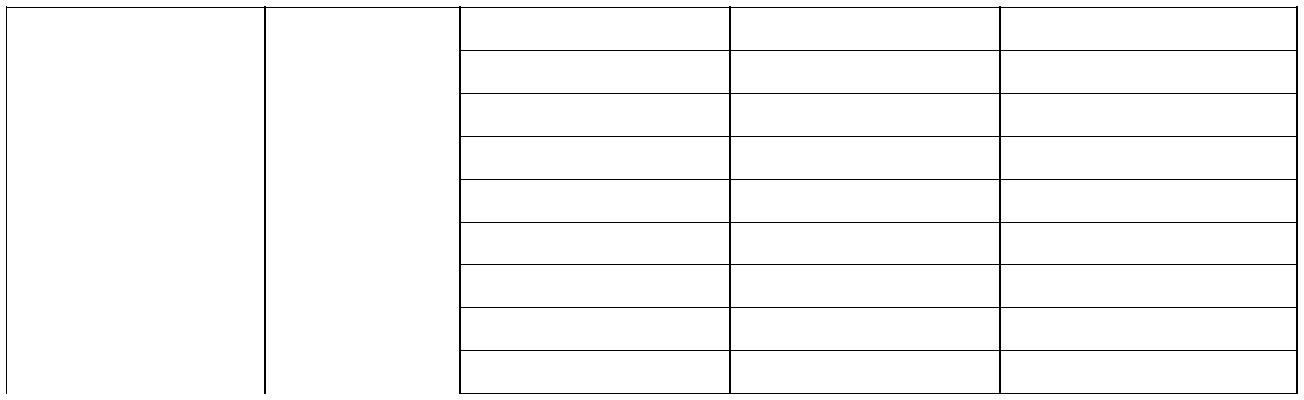
1. **Sử dụng R để kiểm định ý nghĩa thống kê (statistical hypothesis testing)**

**(HoVanPhong\_15110277\_lab5.R)**

Với mỗi độ đo (map, R-prec, bpref, recip\_rank) và với mỗi ngưỡng p (p < 0:05, p < 0:01, và p < 0:001), ghi lại kết quả số cặp thuật toán (trong tất cả 6 thuật toán classic, bm25, lmdrichlet, lmjelinekmercer, dfr, ib) có sự khác biệt là có ý nghĩa thống kê theo bảng sau:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kiểm định** |  | **Độ đo** | **Số cặp p < 0.05 Số cặp p < 0.01 Số cặp p < 0.001** |  |
|  |  | Map |  |  |
| **t-test** |  | R-prec |  |  |
|  | bpref |  |  |
|  |  |  |  |
|  |  | recip\_rank |  |  |
|  |  | Map |  |  |
| **rank test** |  | R-prec |  |  |
|  | bpref |  |  |
|  |  |  |  |
|  |  | recip\_rank |  |  |
| **sign test** |  | Map |  |  |
|  | R-prec |  |  |
|  |  |  |  |
|  |  | bpref |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
|  |  | recip\_rank |  |
| **randomization** |  | Map |  |
|  | R-prec |  |
| **test** |  |  |  |
|  | bpref |  |
|  |  |  |
|  |  | recip\_rank |  |
| **bootstrap test** |  | Map |  |
|  | R-prec |  |
|  |  | bpref |  |
|  |  | recip\_rank |  |

**Các hàm kiểm định trong R**

**Student's t-test**

t.test(x, y, mu=0, alternative="two.sided", paired=TRUE, conf.level=0.95)

Thực hiện t-test cho giả thuyết rằng không có sự khác biệt giữa hai phương pháp (sự khác biệt là do ngẫu nhiên). x và y là các vectơ kết quả đánh giá.

**Wilcoxon signed rank test**

wilcox.test(x, y, mu=0, alternative="two.sided", paired=TRUE, conf.level=0.95)

Thực hiện Wilcoxon signed rank test cho giả thuyết rằng không có sự khác biệt giữa hai phương pháp (sự khác biệt là do ngẫu nhiên). x và y là các vectơ kết quả đánh giá.

**Sign test**

binom.test(s, n, p=0.5, alternative="two.sided", conf.level=0.95)

Thực hiện sign test cho giả thuyết rằng xác xuất thành công là 0.5, s là số lần thành công (nghĩa là số lần phương pháp B tốt hơn phương pháp A, phải tự tính), n là tổng số lần (ở đây ứng với số câu truy vấn). Nếu x và y là vector kết quả, có thể đếm số lần thành công sử dụng sum(x>y).

**Randomization test**

rand.test <- function(x, y, B=10000) {

dist <- replicate(B, mean((2\*rbinom(length(x), 1, 1/2)-1)\*(x-y)))

return mean(mean(x-y) > dist)

}

**Bootstrap test**

bootstrap.test <- function(x, y, B=10000) {

dist <- replicate(B, mean(sample(x-y, length(x-y), replace=T)))

return mean(mean(x-y) > 0)

}