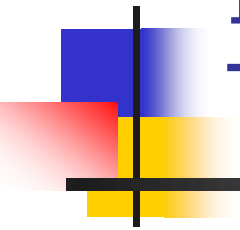


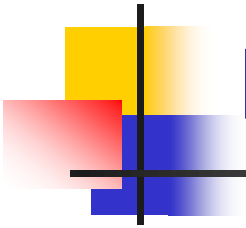
IT4853
Tìm kiếm và trình diễn thông tin

Bài 13. Cài đặt mô hình không gian vec-tơ

IIR.C7. Computing scores in a complete search system

TS. Nguyễn Bá Ngọc, *Bộ môn Hệ thống thông tin, Viện CNTT & TT*
ngocnb@soict.hust.edu.vn

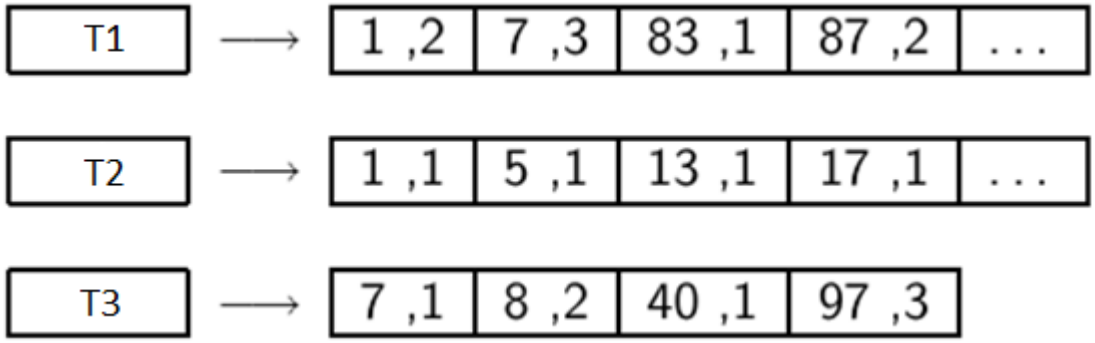
Hà Nội, 2016



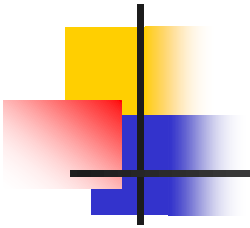
Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

Lưu tf trong danh sách thẻ định vị



- Cần lưu tf để xếp hạng theo VSM.
- Lưu $tf_{t,d}$ cùng với $docID_d$ trong thẻ định vị của d;
- Nên lưu tf, kiểu số nguyên:
  - Không nên lưu trọng số, vì khó nén số thực;
  - Kích thước chỉ mục có tf sau nén tăng không đáng kể.



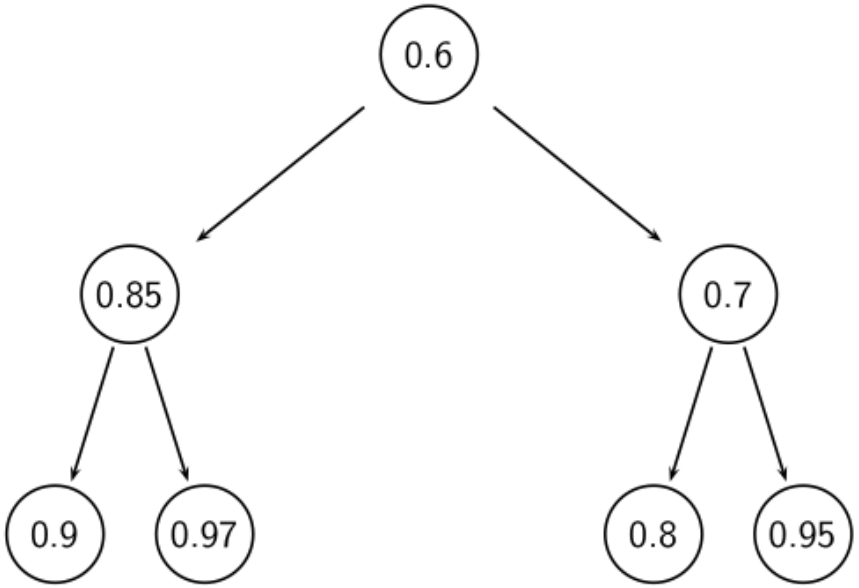
Tính top K như thế nào?

- Trong nhiều ứng dụng, chúng ta không cần xếp hạng toàn bộ văn bản;
  - Chỉ cần top K với K nhỏ (vd, K = 100);
- Cách đơn giản nhất:
  - Tính điểm cho N văn bản, sắp xếp, trả về top K;
  - Hạn chế của cách này? Có cách nào hiệu quả hơn để tính top K?

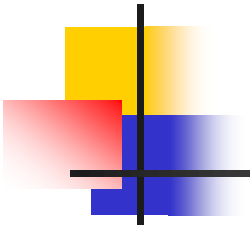
Lựa chọn top K từ N bằng min heap

- Mục đích: Lưu top k văn bản trong min heap;
- Cần O(N log k) thao tác để xây dựng (trong đó N là số lượng văn bản) . . .
- . . . sau đó lấy k giá trị tốt nhất với độ phức tạp O(k log k)

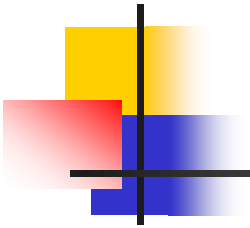
min heap



Lựa chọn top K từ N bằng min heap (2)



- Xử lý văn bản mới d’ với điểm s’:
  - Lấy cực tiểu hiện thời hm (O(1));
  - Nếu s’ < hm bỏ qua và xử lý văn bản tiếp theo;
  - Nếu s’ > hm xóa nút gốc của heap (O(log k));
    - **Thêm vào heap d’/s’ (O(log k)).**



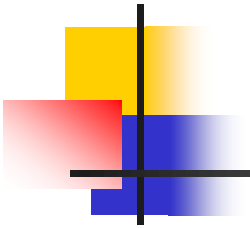
Tính điểm xếp hạng tìm kiếm

- Trong trường hợp đơn giản nhất chúng ta xử lý từng từ truy vấn:
  - Xử lý danh sách thẻ định vị của từ truy vấn đầu tiên;
  - Tạo biến tích lũy cho mỗi docID gặp trong danh sách;
  - Xử lý danh sách thẻ định vị của từ truy vấn thứ hai;
  - . . . và cứ tiếp tục như vậy.
- Trong trường hợp trật tự của danh sách thẻ định vị là ổn định chúng ta có thể xử lý từng văn bản:
  - Tương tự giải thuật lấy giao nhiều danh sách.

Xử lý từng từ truy vấn

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d] += w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top k components of Scores[]
```

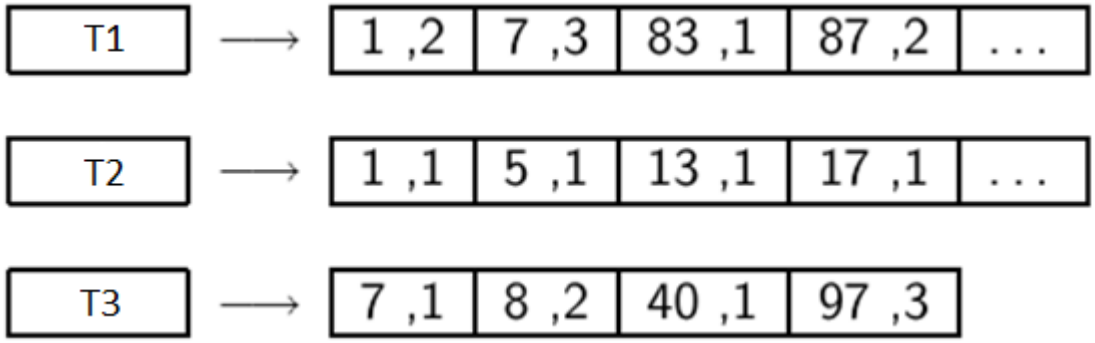
Biến thuộc mảng Scores được gọi là biến tích lũy



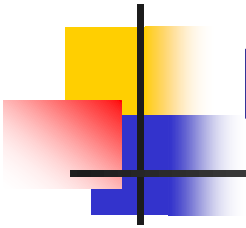
Lưu biến tích lũy

- Đối với web (20 tỉ văn bản), lưu biến tích lũy cho tất cả văn bản trong bộ nhớ là không khả thi.
  - Vì vậy: Chỉ tạo biến tích lũy cho văn bản xuất hiện trong danh sách thẻ định vị;
  - Đồng nghĩa với: Không tạo biến tích lũy cho văn bản với điểm số bằng 0 (vd, văn bản không chứa từ truy vấn nào).

Lưu biến tích lũy (2)



- Cho truy vấn: [T1 T2]:
- Chỉ cần biến tích lũy cho 1, 5, 7, 13, 17, 83, 87
- Không cần tích lũy cho 8, 40, 97

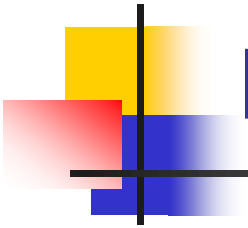


Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

Tối ưu hóa tính top k: Giải thuật tham lam

- Ý tưởng 1: Sắp xếp danh sách thẻ định vị
  - Thay vì sắp xếp theo docID . . .
  - . . . sắp xếp theo đại lượng độc lập với truy vấn, thể hiện tính ưu tiên của văn bản.
- Ý tưởng 2: Cắt tỉa không gian tìm kiếm
  - Xác định tập *A* chứa nhiều văn bản trong top K thỏa mãn: $K < |A| << N$
  - Tìm kiếm top K trong A;
  - Không đảm bảo tính đúng đắn;
  - Trong thực tế, thời gian thực hiện truy vấn gần như bất biến;



Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

Sắp xếp theo tiêu trí ưu tiên

- Tới giờ: Danh sách thẻ định vị đã được sắp xếp theo docID.
- Giải pháp thay thế: Sự dụng đại lượng độc lập truy vấn như "độ ưu tiên" cho văn bản.
- Ví dụ: PageRank của d, ký hiệu là g(d), là đại lượng thể hiện có nhiều trang liên kết đến d (chapter 21)
- Sắp xếp văn bản theo PageRank:

  $$g(d1) > g(d2) > g(d3) > \ldots$$

- Sử dụng điểm tổng hợp:

  $$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

- Cách làm này cho phép ngắt sớm quá trình xử lý danh sách thẻ định vị: chúng ta không cần xử lý toàn bộ danh sách thẻ định vị để tìm top k.

Sắp xếp theo tiêu trí ưu tiên (2)

- Sắp xếp văn bản trong danh sách thẻ định vị theo PageRank:

$$g(d1) > g(d2) > g(d3) > \ldots$$

- Định nghĩa điểm tổng hợp:

$$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

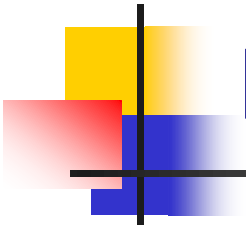
- Giả sử: (i) $g \in [0, 1]$; (ii) $g(d) < 0.1$ đối với văn bản hiện tại; (iii) điểm nhỏ nhất cho top k là 1.2
  - Tất cả các điểm tiếp theo sẽ < 1.2
  - Như vậy chúng ta đã tìm thấy top k và ngừng xử lý phần còn lại của danh sách thẻ định vị.
- Câu hỏi?

Xử lý từng văn bản

- Cả docID và PageRank đều cho kết quả sắp xếp là một danh sách thẻ định vị ổn định.
- Tính cosine theo sơ đồ này là xử lý từng văn bản.
- Chúng ta tính điểm cho d_i trước khi tính điểm cho d_{i+1};

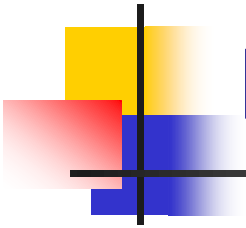
Sắp xếp thẻ định vị theo trọng số

- Ý tưởng: Không xử lý thẻ định vị có ít đóng góp cho kết quả cuối cùng;
- Sắp xếp danh sách thẻ định vị theo trọng số;
  - Trường hợp đơn giản nhất: trọng số tf-idf đã chuẩn hóa (it sử dụng: khó nén);
  - Văn bản trong top k có xu hướng xuất hiện sớm trong danh sách này;
  - Kết thúc sớm có xu hướng ko làm thay đổi top K.
- Nhưng:
  - Chúng ta không còn trật tự ổn định của từ.
  - Không thể áp dụng kỹ thuật xử lý từng văn bản, chỉ có thể xử lý theo từng từ.



Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

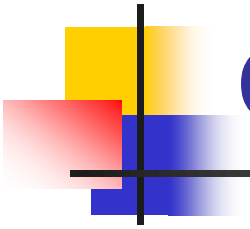


Lọc theo từ truy vấn

- Các lựa chọn lọc cơ bản:
  - Chỉ xét từ truy vấn có idf cao;
  - Chỉ xét các văn bản chứa ít nhất một từ truy vấn;
  - Chỉ xét các văn bản có chứa nhiều từ truy vấn.

Từ truy vấn với idf cao

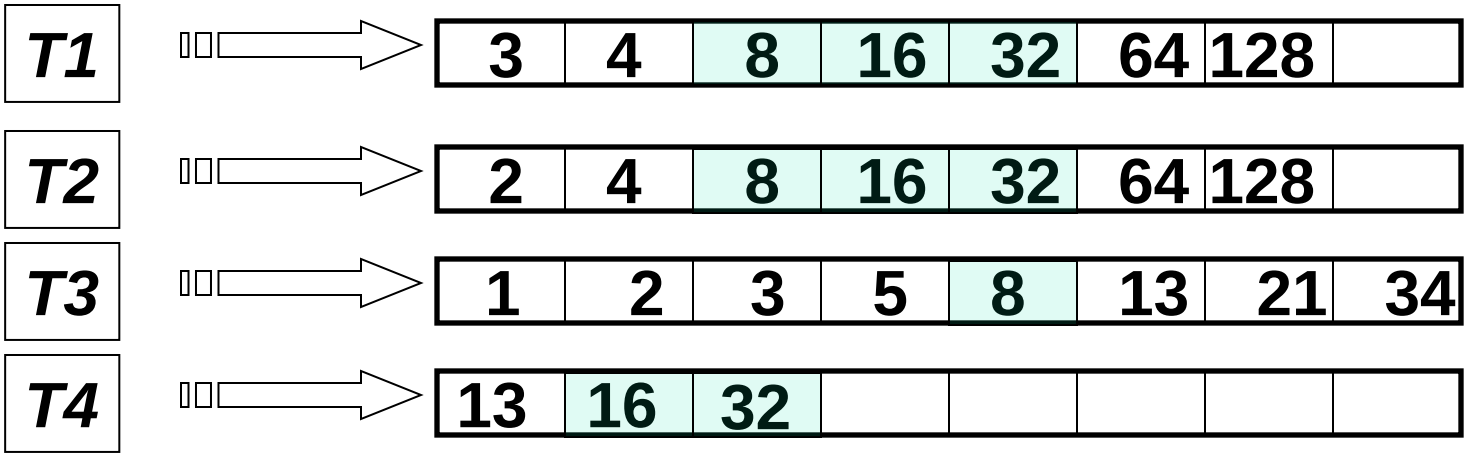
- Ví dụ, với truy vấn *catcher in the rye*
- Chỉ sử dụng *catcher* và *rye*
- Giả thuyết: ***in*** và ***the*** có idf thấp và không ảnh hưởng nhiều tới xếp hạng theo VSM.
- Lợi ích:
  - Các từ có idf thấp có danh sách thẻ định vị dài $\rightarrow$ giúp giảm thiểu đáng kể khối lượng tính toán.



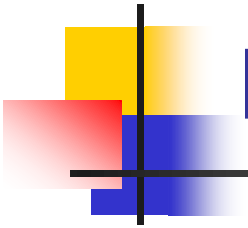
Chứa nhiều từ truy vấn

- Văn bản bất kỳ với ít nhất một từ truy vấn đều có khả năng nằm trong danh sách top K;
- Với truy vấn đa từ chỉ xếp hạng những văn bản chứa nhiều từ truy vấn
  - Có hiệu ứng gần với điều kiện AND trong mô hình Boolean.

Ví dụ, nếu chỉ xét văn bản có tối thiểu 3 từ truy vấn

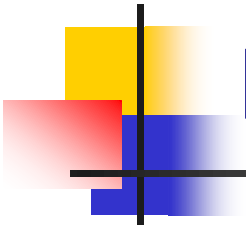


Chỉ xếp hạng các văn bản 8, 16 và 32.



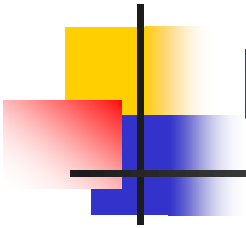
Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm



Danh sách ưu tiên

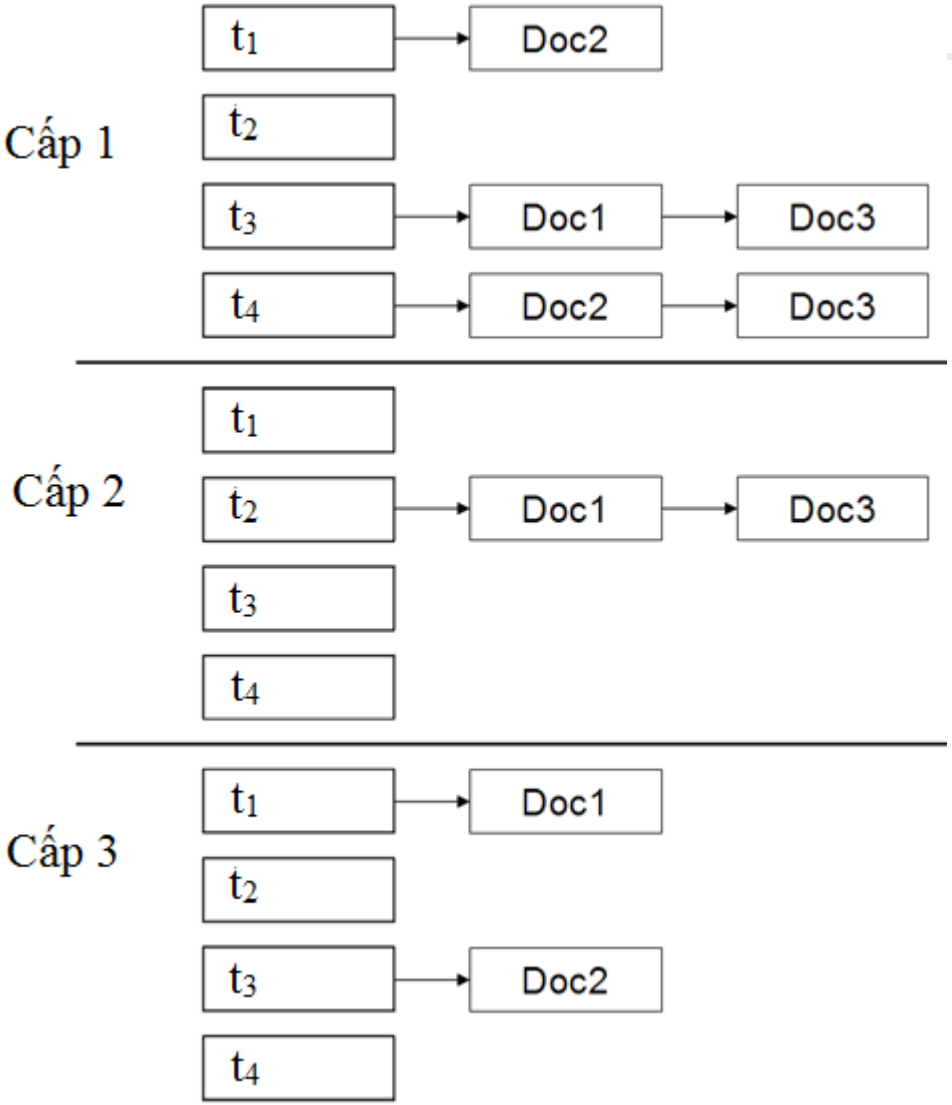
- Xác định trước danh sách r kết quả tiềm năng từ danh sách thẻ định vị của t
  - gọi đây là danh sách ưu tiên;
  - champion list, top docs for t.
- Cần lựa chọn r ở thời điểm xây dựng chỉ mục
  - Có khả năng $r < K$;
- Khi thực hiện truy vấn ưu tiên lựa chọn top-K từ những văn bản này.

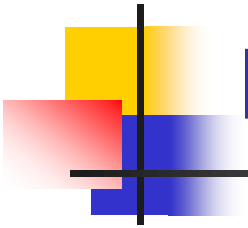


Phân cấp chỉ mục

- Chia chỉ mục thành nhiều chỉ mục con theo mức độ ưu tiên
- Trong quá trình thực hiện truy vấn thực hiện tìm kiếm trong những chỉ mục có độ ưu tiên cao trước
  - Dừng nếu tìm đủ K văn bản;
  - Nếu chưa đủ, tiếp tục tìm trong những danh sách có độ ưu tiên thấp hơn.
- Tiêu trí phân cấp
  - trọng số kết hợp: g(d) + $\text{tf-idf}_{t,d}$
  - tf-idf, v.v.

Ví dụ phân cấp chỉ mục





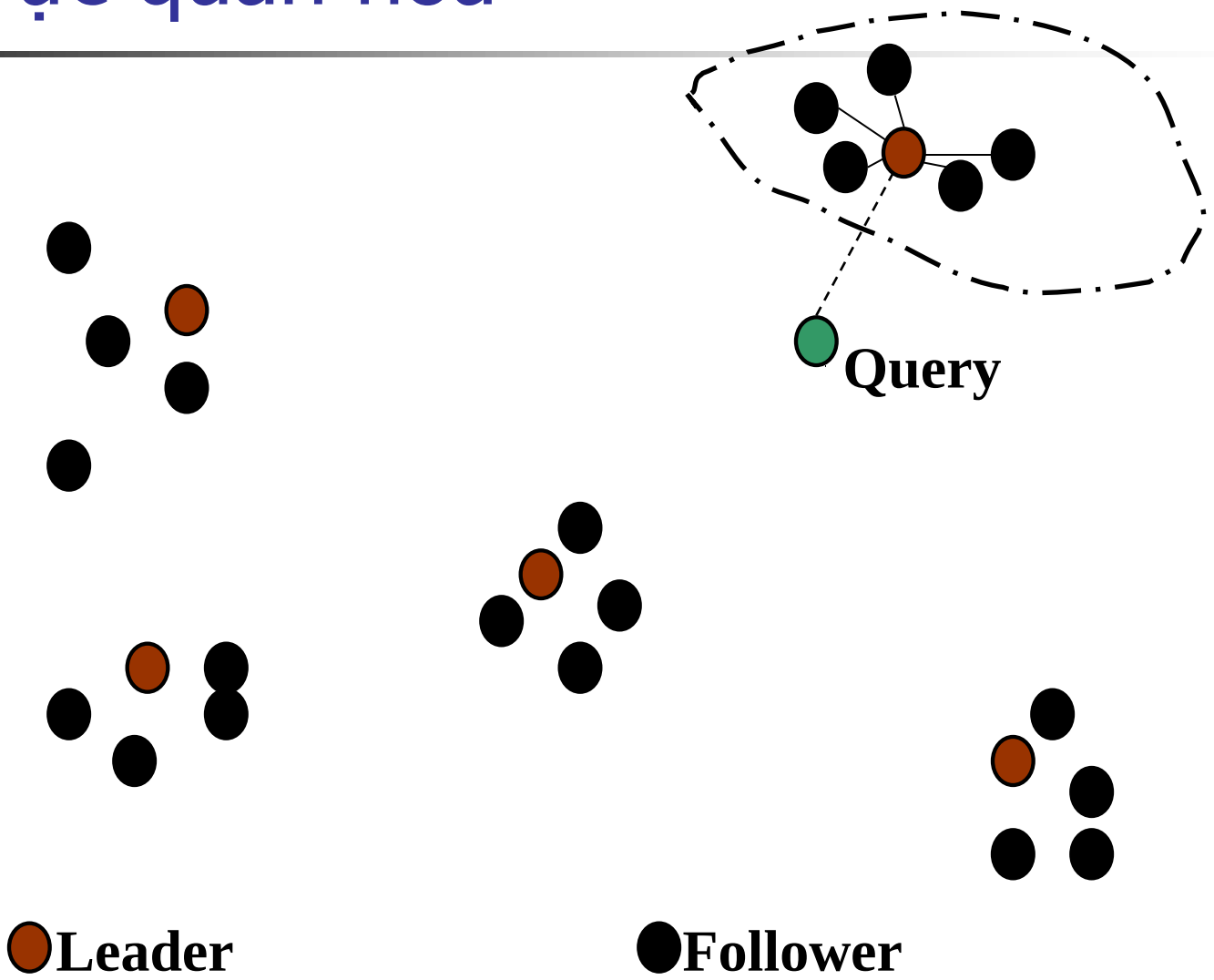
Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

Phân cụm ngẫu nhiên

1. Tiền xử lý:
   - Chọn ngẫu nhiên $\sqrt{N}$ văn bản: gọi là *leaders*
   - Thực hiện phân cụm văn bản:
     - Gọi các văn bản gắn với leader là *followers;*
     - Có thể: Mỗi leader có khoảng ~ $\sqrt{N}$ followers.
2. Thực hiện truy vấn:
   - Tìm leader L gần q nhất.
   - Tìm top K trong số followers của L.

Trực quan hóa

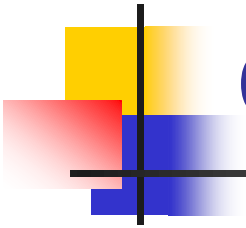


Giải pháp tổng quát

- Sử dụng các tham số b_1 và b_2:
  - b_1 là số leader của mỗi follower.
  - b_2 là số cụm sẽ sử dụng trong thực hiện truy vấn.
- Có thể thiết lập lại danh sách leader và follower

Nội dung chính

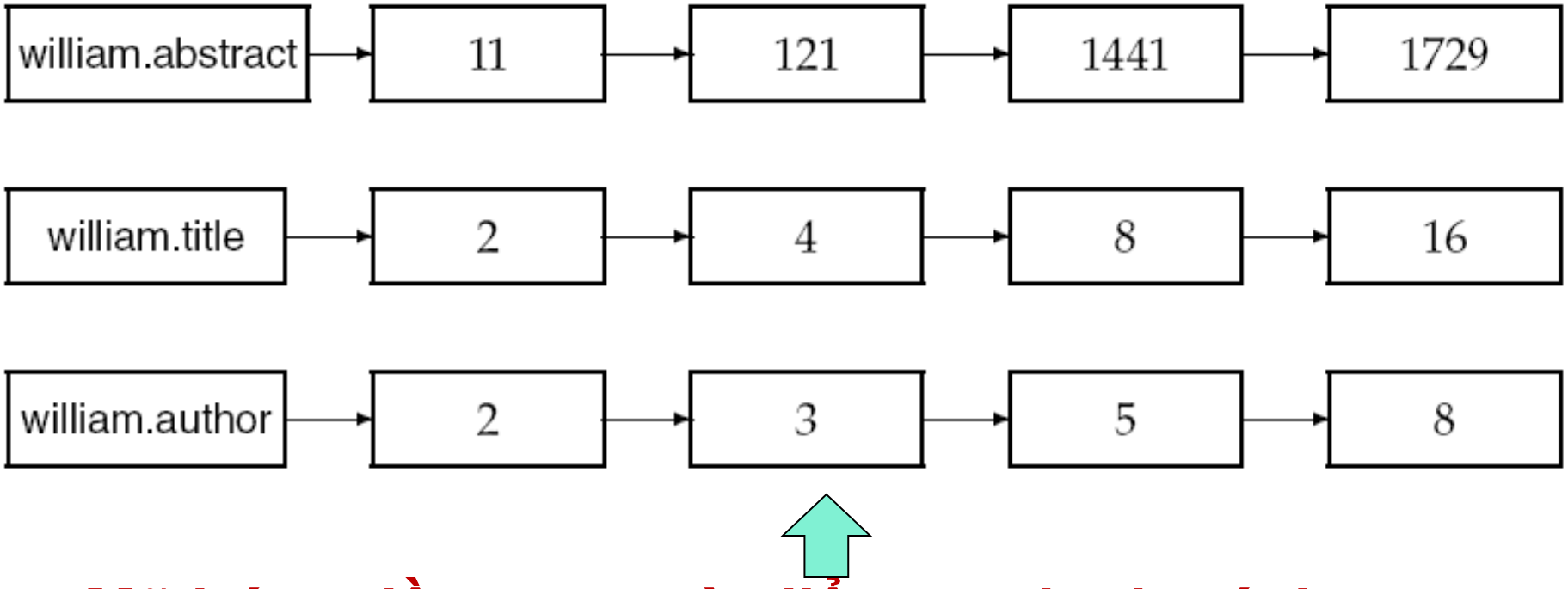
- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm



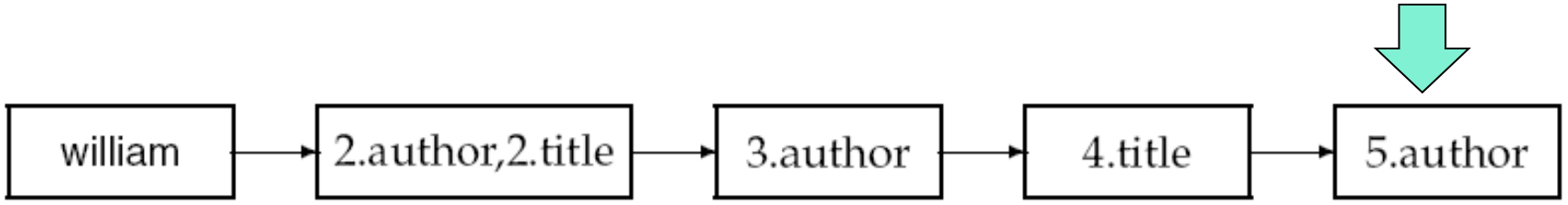
Chia miền

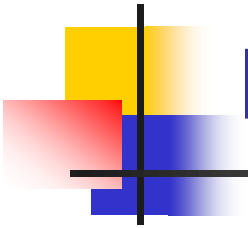
- Một miền là một phần văn bản, có thể có độ dài tùy ý:
  - Tiêu đề
  - Tóm tắt
  - Tài liệu tham khảo …
- Tổ chức chỉ mục ngược trên miền sẽ rất hữu ích trong tìm kiếm
  - v.d., "tìm văn bản có *merchant* trong tiêu đề và phù hợp với *gentle rain*"

Ví dụ chỉ mục chia miền



Mã hóa miền trong từ điển vs. danh sách

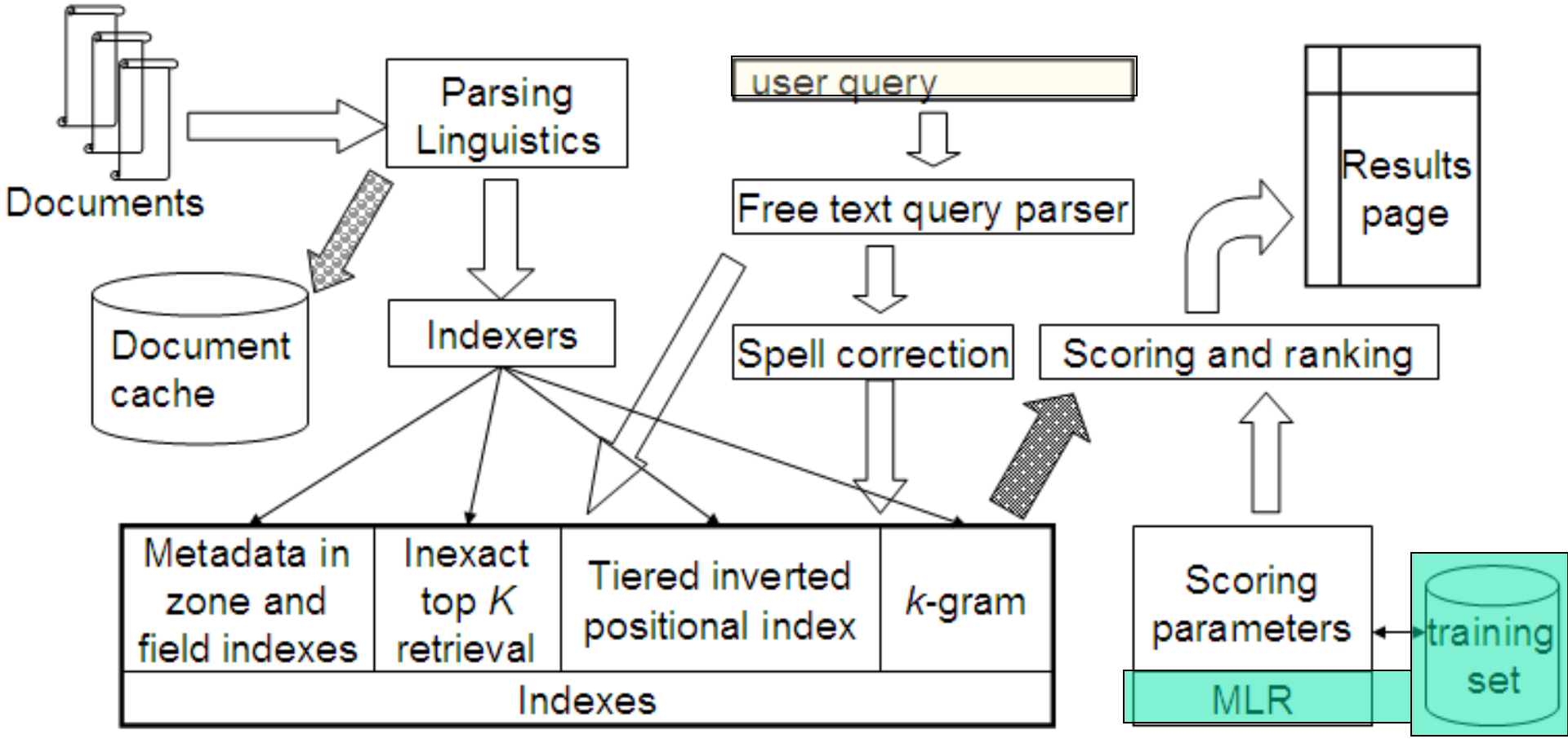


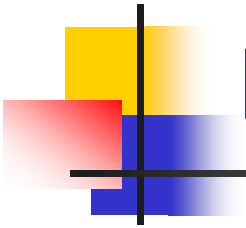


Nội dung chính

- Xếp hạng theo cosine
- Tối ưu hóa tính top K
  - Sắp xếp theo tiêu trí ưu tiên
  - Lọc theo từ truy vấn
  - Phân cấp chỉ mục
  - Phân cụm ngẫu nhiên
- Chỉ mục đa miền
- Tổng quan hệ thống tìm kiếm

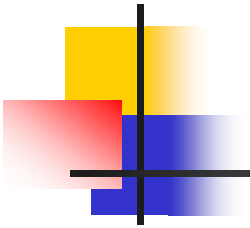
Sơ đồ khái quát





Nhận diện kiểu truy vấn

- Một chuỗi từ có thể được hiểu theo nhiều cách khác nhau:
  - Một truy vấn dạng câu;
  - Nhiều câu nhỏ hơn;
  - Hoặc vec-tơ trên từ.
- Người dùng thường ưa thích văn bản có từ truy vấn xuất hiện gần nhau
- Nhận diện kiểu truy vấn là quá trình xác định cách thực hiện truy vấn tối ưu.



Các thành phần chưa xét

- Cache văn bản: cần sử dụng để sinh trích đoạn (tóm tắt động)
- Hàm xếp hạng học máy
- Proximity ranking (vd, xếp hạng văn bản có từ truy vấn xuất hiện gần nhau cao hơm những văn bản có từ truy vấn xuất hiện xa nhau)
- Nhận diện kiểu truy vấn: Query parser

Bài tập 13.1

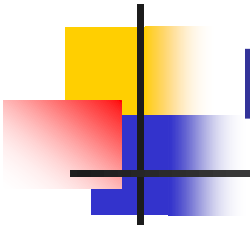
Giả sử đại lượng ưu tiên cho Doc1, Doc2, Doc3 lần lượt là 0.25, 0.5 và 1. Hãy vẽ danh sách thẻ định vị nếu thẻ định vị được sắp xếp theo tổng của đại lượng ưu tiên và tf đã chuẩn hóa Euclid.

tf

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

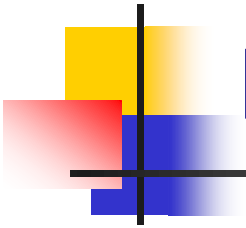
tf đã chuẩn hóa Euclid.

	Doc1	Doc2	Doc3
car	0.88	0.09	0.58
auto	0.10	0.71	0
insurance	0	0.71	0.70
best	0.46	0	0.41



Bài tập 13.2

Nếu truy vấn chỉ chứa một từ, hãy giải thích vì sao sử dụng danh sách ưu tiên với r = K là đủ để xác định K văn bản hàng đầu. Hãy gợi ý một điều chỉnh đơn giản cho trường hợp có s từ truy vấn với s > 1.



Bài tập 13.3

Vấn đề láng giềng gần nhất trên mặt phẳng có thể phát biểu như sau: Cho N điểm trên mặt phẳng, và một điểm truy vấn Q. Cần tìm tất cả các điểm trong N điểm gần với Q nhất. Để tránh tính khoảng cách từ Q đến tất cả các điểm có thể áp dụng phương pháp cắt tỉa không gian tìm kiếm dựa trên phân cụm. Hãy lấy ví dụ với hai tâm cụm sao cho phương pháp cắt tỉa trả về kết quả sai.