

**STEP 1: SHINGLING: CONVERT DOCUMENTS TO SETS**

**TÀI LIỆU DƯỚI DẠNG DỮ LIỆU HIGH-DIM**

Các cách tiếp cận đơn giản:

* Tài liệu = tập hợp các từ xuất hiện trong tài liệu
* Tài liệu = tập hợp các từ "quan trọng"
* Không hoạt động tốt cho ứng dụng này. Tại sao?

Chú ý: Cần phải tính đến thứ tự của các từ!

Một cách khác: **Shingles!**

**ĐỊNH NGHĨA: SHINGLES**

K-shingle (hoặc k-gram) cho 1 tài liệu là một chuỗi các tokens k xuất hiện trong tài liệu

* Token có thể là ký tự, từ ngữ hoặc thứ gì đó khác, tùy thuộc vào ứng dụng
* Giả sử tokens = ký tự cho ví dụ

**Ví dụ:** k =2; tài liệu D1 = abcab  
Tập hợp 2-shingles: S(D1) = {ab, bc, ca}

Tùy chọn: shingles dưới dạng 1 cái túi (nhiều bộ), đếm ab hai lần: S'(D1) = {ab, bc, ca, ab}

**NÉN SHINGLES**

Để **nén shingles dài**, chúng ta có thể **băm** chúng thành 4 bytes

Đại diện cho một tài liệu bằng tập các giá trị băm của k-shingles

* Ý tưởng: Hai tài liệu có thể (hiếm khi) có shingles chung, trong khi trên thực tế chỉ có các giá trị băm được chia sẻ

**Ví dụ:** k=2; tài liệu D1= abcab

Tập của 2-shingles: S(D1) = {ab, bc, ca}

Băm singles: h(D1) = {1, 5, 7}

**METRIC TƯƠNG TỰ CHO SHINGLES**

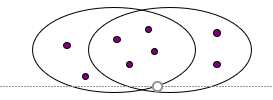
Tài liệu D1 là một tập các k-shingles của nó C1= S(D1)

Tương đương, mỗi tài liệu là một vectơ 0/1 trong không gian k-shingles

* Với shingle độc nhất là một chiều không gian
* Các vectơ rất thưa thớt

**Một biện pháp tự nhiên tương tự là sự tương đồng Jaccard:**

sim(D1,D2) -| C1∩C2|/| C1∪C2 |



**GIẢ ĐỊNH CÔNG VIỆC**

Các Tài liệu có nhiều shingles chung thì có văn bản tương tự, ngay cả khi thứ tự văn bản xuất hiện theo cách khác nhau

Cảnh báo: Bạn phải chọn **k** đủ lớn, hoặc hầu hết các tài liệu sẽ có shingles

* k = 5 là OK “ổn” cho các tài liệu ngắn
* k = 10 là better “tốt hơn” cho các tài liệu dài

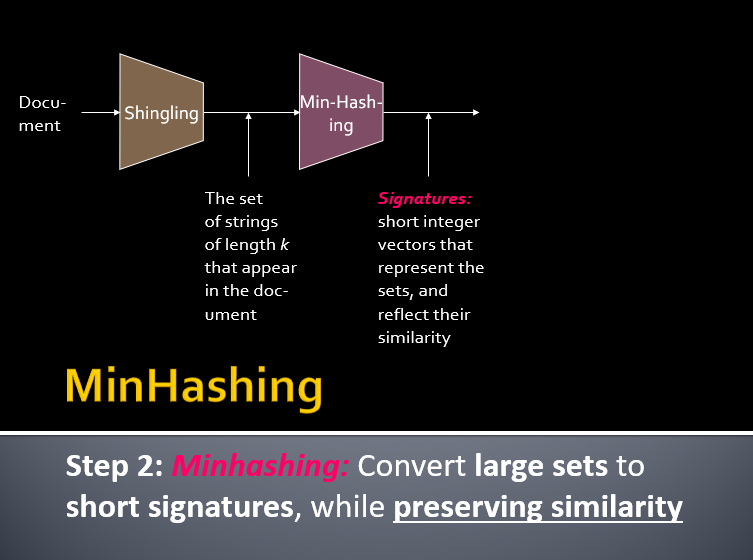
**ĐỘNG LỰC CHO MINHASH / LSH**

Giả sử chúng ta cần tìm các tài liệu gần trùng lặp trong số hàng triệu tài liệu

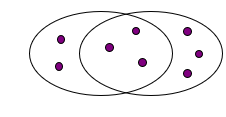
Chúng ta sẽ phải tính toán **pairwise Jaccard similarities (các cặp tương đồng Jaccard)** cho mỗi cặp tài liệu

* ≈ 5\*1011 so sánh
* Với tốc độ 105 giây/ngày và 106 so sánh/giây, nó sẽ mất **5 ngày**

Đối với **N = 10** hàng triệu, phải mất hơn một năm



**MÃ HÓA SETS DƯỚI DẠNG BIT VECTORS**

****Nhiều vấn đề tương tự có thể được chính thức hóa như **tìm các tập hợp con có giao điểm quan trọng**

Bộ mã hóa sử dụng vectơ 0/1 (bit, boolean)

* Một kích thước trên mỗi phần tử trong tập hợp phổ quát

Chứng minh đặt giao điểm (intersection) theo bitwise AND và đặt phép hợp (union) theo bitwise OR

**Ví dụ:** C1 = 10111; C2 = 10011

* Kích thước intersection = 3; kích thước của union = 4,
* **Jaccard** **similarity** (không phải khoảng cách) = 3/4
* **Khoảng cách**: d(C1,C2) = 1 – (**Jaccard** **similarity**) = 1/4

**TỪ SETS ĐẾN BOOLEAN MATRICES**

Hàng = phần tử (shingles)

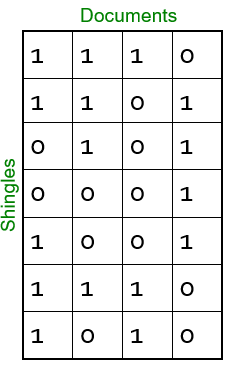
Cột = tập hợp (tài liệu)

* 1 trong hàng e và cột s nếu và chỉ khi e là thành viên của s
* Tương tự cột là Jaccard similarity của các tập hợp tương ứng (hàng có giá trị 1)
* Ma trận điển hình là ma trận thưa thớt!

**Mỗi tài liệu là một cột:**

**Ví dụ:** sim(C1, C2) = ?

* Kích thước intersection = 3; kích thước của union = 6, **Jaccard** **similarity** (không phải khoảng cách) = 3/6
* d(C1,C2) =1 – (**Jaccard** **similarity**) = 3/6



**OUTLINE: TÌM CÁC CỘT TƯƠNG TỰ**

**Cho đến nay:**

* Tài liệu → Tập hợp cỉa shingles
* Đại diện cho các tập hợp dưới dạng vectơ boolean trong ma trận

**Mục tiêu tiếp theo**: Tìm các cột tương tự trong khi tính toán chữ ký nhỏ

* Sự giống nhau của cột == sự giống nhau của chữ ký

**Mục tiêu tiếp theo**: Tìm các cột tương tự, Chữ ký nhỏ

Cách tiếp cận **Naïve**:

1) Chữ ký của các cột: tóm tắt nhỏ các cột

2) Kiểm tra các cặp chữ ký để tìm các cột tương tự

*Điều cần thiết*: Điểm tương đồng của chữ ký và cột có liên quan

3) Tùy chọn: Kiểm tra xem các cột có chữ ký tương tự có thực sự giống nhau không

Cảnh báo:

So sánh tất cả các cặp có thể tốn quá nhiều thời gian: **Công việc cho LSH**

* Các phương pháp này có thể tạo ra phủ định sai và thậm chí là khẳng định sai (nếu kiểm tra tùy chọn không được thực hiện)

**CỘT BĂM (CHỮ KÝ)**

**Ý tưởng chính**: "băm" mỗi cột **C** đến một **chữ ký nhỏ** **h(C),** sao cho:

(1) h(C) đủ nhỏ để chữ ký phù hợp với RAM

(2) sim(C1, C2) giống như "sự tương đồng" của chữ ký h(C1) và h(C2)

**Mục tiêu**: Tìm hàm băm h(·) như vậy:

* Nếu sim (C1,C2) cao, thì với prob cao**. h(C1) = h(C2)**
* Nếu sim (C1,C2) thấp, thì với prob cao. **h(C1) ≠ h(C2)**

Băm docs vào buckets. Mong đợi rằng "hầu hết" các cặp tài liệu gần trùng lặp băm vào cùng một bucket!

**MIN-HASHING**

**Mục tiêu**: Tìm hàm băm h(·) như vậy:

* Nếu sim (C1,C2) cao, thì với prob cao. **h(C1) = h(C2)**
* Nếu sim (C1,C2) thấp, thì với prob cao**. h(C1) ≠ h(C2)**

Rõ ràng, hàm băm phụ thuộc vào metric tương tự:

Không phải tất cả các chỉ số tương tự đều có hàm băm phù hợp

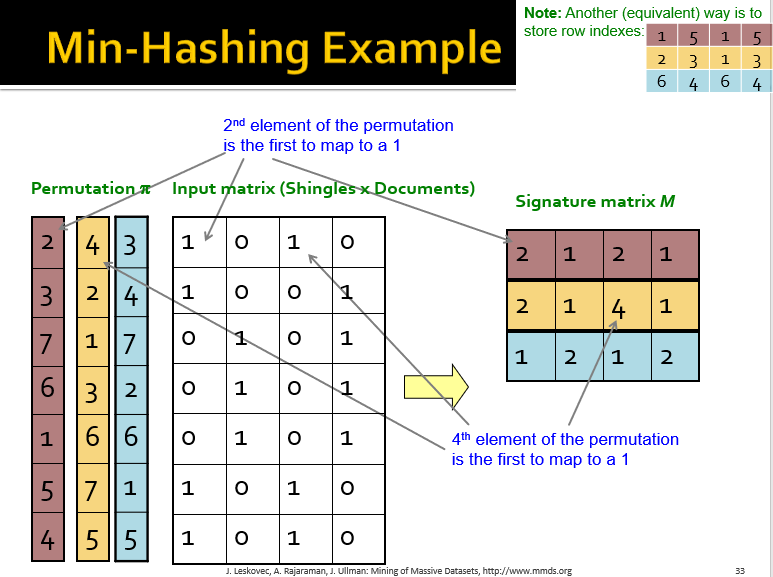
Có một hàm băm phù hợp cho **Jaccard similarity**: Nó được gọi là **Min-Hashing**

Hãy tưởng tượng các hàng của ma trận boolean được hoán vị dưới dạng hoán vị ngẫu nhiên π

Xác định hàm "băm" hπ(C) = chỉ mục của hàng đầu tiên (theo thứ tự hoán vị π) trong đó cột C có giá trị là 1:

hπ(C) = minππ(C)

Sử dụng một số hàm băm độc lập (nghĩa là hoán vị) để tạo chữ ký của một cột



Chọn hoán vị ngẫu nhiên π

Yêu cầu bồi thường: Pr[hπ(C1)- hπ(C2)]- sim(C1,C2)

Tại sao?

Hãy để X là một doc (bộ bệnh zona), y X là một∈ shingle

Sau đó: Pr[(y) = min((X))] = 1/| X|ππ

Cũng có khả năng bất kỳ y X nào ∈ được ánh xạ đến phần tử min

Hãy để y được s.t. (y) = min((Cππ1∪C2))

Sau đó: π (y) = min((Cπ1))nếu y C∈1 , hoặc

π (y) s min ((Cπ2))nếu và C∈2

Vì vậy, prob. rằng cả hai đều đúng là prob. y ∈ C1 ∩ C2

Pr[min((Cπ1))-min ((Cπ2)|)] C1∩C2|/| Simđiện thoạiC∪1C| (C1,C2)