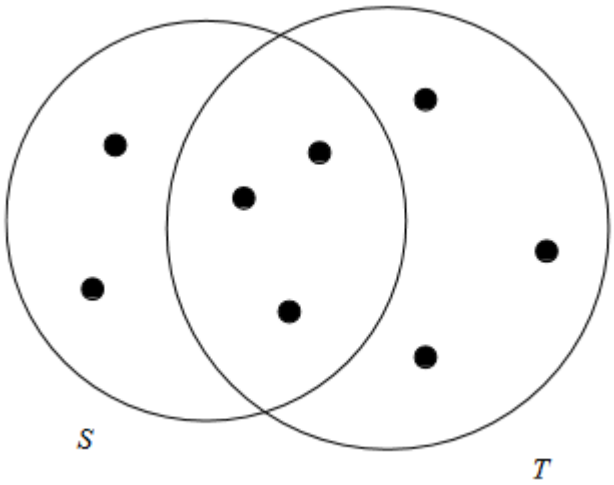
Độ đo tương đồng jaccard



Jaccard của tập S và T là | S ∩ T | / | S ∪ T |, nghĩa là, tỷ số giữa kích thước của giao điểm của S và T với kích thước của hợp của chúng. Người ta biểu thị sự giống nhau về Jaccard của S và T bằng SIM (S, T).

$$SIM(S,T) = \frac{|S \cap T|}{|S \cup T|} = \frac{|S \cap T|}{|S| + |T| - |S \cap T|} = \frac{3}{8}$$

In [14]:
```
def jaccard_similarity(Item1, Item2):
    s1 = set(Item1)
    s2 = set(Item2)
    return len(s1.intersection(s2)) / len(s1.union(s2))
```

In [6]:
```
A = ['NGUYEN DANH THAO', 'CHKHMT10B', '20125291', 'IUH']
B = ['LE THANH HOA', 'CHKHMT10B', '2012....', 'IUH']
print(jaccard_similarity(A, B))
```

```
0.3333333333333333
```

In [13]:
```
def simple_recommender_sys(Item1, Item2, Item3):
    J1 = jaccard_similarity(Item1, Item2)
    J2 = jaccard_similarity(Item2, Item3)
    J3 = jaccard_similarity(Item3, Item1)
    if J1 > J2:
        if J1 > J3:
            print("1 giống 2 hơn 3")
        elif J1 == J3:
            print("độ tương đồng giữa 1 và 2 với 1 và 3 là như nhau")
        else:
            print("1 giống 3 hơn 2")
    elif J1 == J2:
        if J1 > J3:
            print("độ tương đồng giữa 1 và 2 với 2 và 3 là như nhau")
        elif J1 == J3:
            print("độ tương đồng giữa 1 và 2, 2 và 3 với 1 và 3 là như nhau")
        else:
            print("1 giống 3 hơn")
    elif J1 < J2:
        if J2 > J3:
            print("2 giống 3 hơn")
        elif J2 == J3:
            print("độ tương đồng giữa 2 và 3 với 1 và 3 là như nhau")
        else:
            print("1 giống 3 hơn")
C = ['CHUNG DUC CUONG', 'CHKHMT10B', '2012....', 'IUH']
simple_recommender_sys(A, B, C)
```

```
2 giống 3 hơn
```

Shingling of Documents

Cách hiệu quả nhất để biểu diễn tài liệu dưới dạng tập hợp, nhằm mục đích xác định các tài liệu tương tự về mặt từ vựng là xây dựng từ tài liệu một tập hợp các chuỗi ngắn xuất hiện bên trong nó. Nếu chúng ta làm như vậy, thì các tài liệu chia sẻ các đoạn ngắn như câu hoặc thậm chí cụm từ sẽ có nhiều yếu tố chung trong tập hợp của chúng, ngay cả khi những câu đó xuất hiện theo thứ tự khác nhau trong hai tài liệu.

- k-Shingles ?
  - Tài liệu là một chuỗi ký tự. Xác định k-shingle cho một tài liệu là bất kỳ chuỗi con nào có độ dài k được tìm thấy trong tài liệu.
  - Giả sử tài liệu D của chúng ta là chuỗi abcdabd, và chúng ta chọn k = 2. Khi đó tập hợp 2-shingles cho D là {ab, bc, cd, da, bd}.
  - Có một số tùy chọn liên quan đến cách xử lý khoảng trắng (trống, tab, dòng mới, v.v.). Có thể hợp lý khi thay thế bất kỳ chuỗi ký tự khoảng trắng nào bằng một ô trống duy nhất.
  - Vd Nếu chúng ta sử dụng k = 9, nhưng loại bỏ hoàn toàn khoảng trắng, thì chúng ta sẽ thấy một số điểm tương đồng về từ vựng trong các câu “The plane was ready for touch down”. và “The quarterback scored a touchdown”. Nếu giữ khoảng trống thì chuỗi thứ nhất có shingles "touch dow", "ouch down" và chuỗi thứ 2 là "touchdown". còn nếu loại bỏ khoảng trống thì cả 2 chuỗi có cùng shingle là "touchdown".
- Shingle size?
  - không nên chọn quá ngắn vì sẽ độ tương đồng jaccard sẽ lớn.
  - Kích thước k lớn như thế nào phụ thuộc vào độ dài của các tài liệu điển hình và tập hợp các ký tự điển hình lớn như thế nào
  - Thường email hay chọn k=5, các bài báo hay chọn k = 9 là an toàn.
- Hashing Shingles: Thay vì sử dụng các chuỗi con trực tiếp dưới dạng shingles, chúng ta có thể chọn một hàm băm ánh xạ các chuỗi có độ dài k với một số nhóm và coi số nhóm kết quả là một số nhóm. Tập hợp đại diện cho một tài liệu sau đó là tập hợp các số nguyên là số nhóm của một hoặc nhiều k-shingles xuất hiện trong tài liệu.

Similarity-Preserving Summaries of Sets

- Bộ Shingles rất lớn. Ngay cả khi chúng ta băm chúng thành bốn byte mỗi loại, không gian cần thiết để lưu trữ một tập hợp vẫn gần gấp bốn lần không gian mà tài liệu đã sử dụng. Nếu chúng ta có hàng triệu tài liệu, có thể không lưu trữ được tất cả các bộ shingle trong bộ nhớ chính.
- Vậy mục tiêu là thay thế các tập hợp lớn bằng các phép sửa đổi nhỏ hơn nhiều được gọi là “chữ ký”. Đặc tính quan trọng mà chúng ta cần cho chữ ký là chúng ta có thể so sánh chữ ký của hai tập hợp và ước tính sự giống nhau về Jaccard của các tập hợp cơ bản chỉ từ các chữ ký. Không thể là các chữ ký đưa ra sự giống nhau chính xác của các tập hợp mà chúng đại diện, nhưng các ước tính mà chúng cung cấp là gần nhau, và các chữ ký càng lớn thì các ước tính càng chính xác. Ví dụ: nếu chúng ta thay thế bộ shingle băm 200.000 byte có nguồn gốc từ các tài liệu 50.000 byte bằng các chữ ký của 1000 byte, chúng ta thường có thể nhận được trong vòng một vài phần trăm.
- Trong hình dưới là một ví dụ về ma trận biểu diễn các tập được chọn từ tập phổ quát {a, b, c, d, e}. Ở đây, S1 = {a, d}, S2 = {c}, S3 = {b, d, e} và S4 = {a, c, d}.

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Minhashing

- Để tối thiểu hóa một tập hợp được đại diện bởi một cột của ma trận đặc trưng, hãy chọn một hoán vị của các hàng. Giá trị minhash của bất kỳ cột nào là số của hàng đầu tiên, theo thứ tự hoán vị, trong đó cột có giá trị 1.
- Giả sử chúng ta chọn thứ tự các hàng beadc cho ma trận trên. Hoán vị này xác định một hàm minhash h mà ánh xạ đặt thành các hàng. Giá trị minhash của tập S1 theo h. Cột đầu tiên, là cột của tập S1, có 0 ở hàng b, vì vậy chúng ta tiến hành chuyển sang hàng e, cột thứ hai theo thứ tự hoán vị. Lại có một số 0 trong cột cho S1, vì vậy chúng ta tiếp tục đến hàng a, nơi chúng ta tìm thấy một 1. Như vậy. h(S1) = a.

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

- vì vậy nó trở thành ma trận như hình trên. Trong ma trận này, chúng ta có thể đọc các giá trị của h bằng cách quét từ trên xuống cho đến khi chúng ta đến giá trị 1. Như vậy, chúng ta thấy rằng h(S2) = c, h(S3) = b, và h(S4) = a.

Computing Minhash Signatures in Practice

1. Tính h1(r), h2(r), . . . , hn(r)
2. Đối với mỗi cột c làm như sau:
   - Nếu c có 0 trong hàng r, không làm gì cả.
   - Nếu c có 1 trong hàng r, thì với mỗi i = 1, 2 ,. . . , n đặt SIG (i, c) nhỏ hơn giá trị hiện tại của SIG (i, c) và hi (r).

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Đầu tiên ta chọn 2 hàm băm có dạng h1 = x + 1 mod 5 và h2 = 3x + 1 mod 5. giá trị của 2 hàm này được tính và điền ở 2 cột cuối (hình trên).

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

khởi tạo

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

h1(0) = 1, h2(0) = 1

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

h1(1) = 2, h2(1) = 4

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

h1(2) = 3, h2(2) = 2

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

h1(3) = 4, h2(3) = 0

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

h1(4) = 0, h2(4) = 3

- Diễn giải: Để tính toán ma trận signature ta làm như sau:

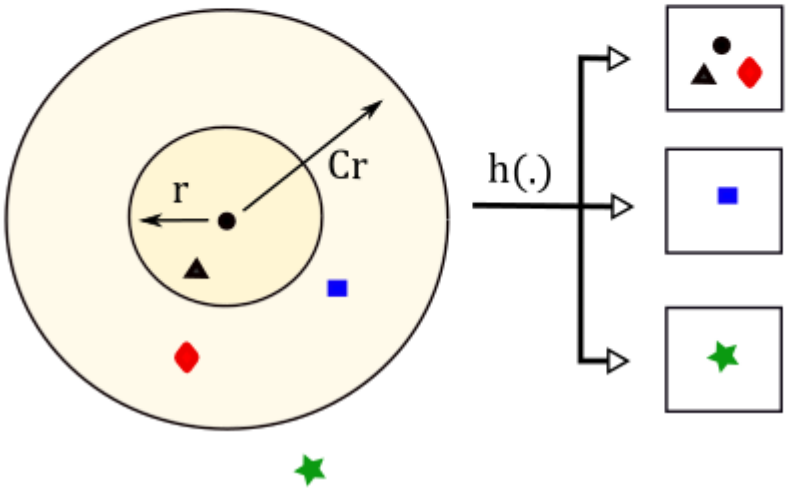
- khởi tạo: đặt tất cả các giá trị trong các chuỗi là ∞ cho cả 2 hàng h1 và h2.
- Bước 1: duyệt hàng 0 trên ma trận đặc trưng và xét các cột có giá trị là 1. Thay thế các vị trí có giá trị là 1 bằng giá trị của hàm băm tương ứng. ví dụ: dòng 0 có S1 = 1 và S4 = 1, ta thay thế h1(0) = 1 và h2(0) = 1 tương ứng.
- tiếp tục duyệt các dòng tiếp theo như bước 1 và cập nhật lại giá trị hàm băm nhỏ nhất ta được bảng cuối cùng

Locality Sensitive Hashing

Ý tưởng của LSH là sử dụng một hàm băm h(.) để băm các điểm dữ liệu trong P vào một bảng T[.] sao cho nếu hai điểm p,q gần nhau thì khả năng cao p,q sẽ được băm vào cùng một ô. Ngược lại, p và q sẽ bị băm vào hai ô khác nhau. Khi trả lời truy vấn, ta chỉ việc tính mã băm h(p) và đưa ra tất cả các điểm lưu trong ô T[h(p)] . Hàm băm h(.) như vậy được gọi là kiểu locality sensitive.

LSH family: Cho trước hai số thực $r > 0$ và $C \geq 1$. Một họ các hàm băm $\mathcal{H}$ là $(r, Cr, p_{\text{close}}, p_{\text{far}})$-sensitive nếu ta chọn ngẫu nhiên ra một hàm băm $h(.) \in \mathcal{H}$, với bất kì hai điểm riêng biệt $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$, ta có:

1. Nếu $d(\mathbf{p}, \mathbf{q}) \leq r$ thì $\Pr[h(p) = h(q)] \geq p_{\text{close}}$.
2. Nếu $d(\mathbf{p}, \mathbf{q}) \geq Cr$ thì $\Pr[h(x) = h(y)] \leq p_{\text{far}}$.



In [ ]: