MinHash và Locality Sensitive Hashing – LSH

Phân cụm văn bản sử dụng hàm băm với thời gian xấp xỉ O(n) (độ chính xác sẽ thấp hơn so với TF-IDF hoặc Word2Vec/Doc2Vec). Ý tưởng cơ bản là:

- Biểu diễn lại văn bản bằng tập ký tự liên tiếp hoặc cụm từ liên tiếp (shingles/n-grams).
- Dùng tập hàm băm để biến đổi 1 văn bản về 1 signature ngắn.
- LSH để phân cụm: nhóm các văn bản vào các bucket nếu có signature tương đồng.
- Dùng khoảng cách Jaccard để tính độ tương đồng giữa các văn bản.

Ví dụ bước chạy của MinHash và LSH trên 3 văn bản:

"Machine learning is fun", "Machine learning is hard", "Data structure is easy"

A. Dùng n-grams

Bước 1: Chuẩn hóa và chuyển thành tập từ (shingling), dùng shingling đơn giản theo từ (1-gram).

Tài liệu	Tập từ
D1	{machine, learning, is, fun}
D2	{machine, learning, is, hard}
D3	{data, structure, is, easy}

Tập từ điển tổng hợp (universe) gồm:
→ {machine, learning, is, fun, hard, data, structure, easy}
→ đánh số từ:

0: machine

1: learning

2: is

3: fun

4: hard

5: data

6: structure

7: easy

- **Bước 2: Biểu diễn mỗi văn bản thành vector đặc trưng (bit vector)**

Tài liệu	0	1	2	3	4	5	6	7
D1	1	1	1	1	0	0	0	0
D2	1	1	1	0	1	0	0	0
D3	0	0	1	0	0	1	1	1

🔸 **Bước 3: Tính MinHash signature**

Giả sử ta dùng 2 hàm băm đơn giản để minh họa:

h1(x) = (x + 1) % 8

h2(x) = (3 * x + 1) % 8

→ Tính chữ ký MinHash (MinHash Signature) với mỗi tài liệu:
(với mỗi hàm băm, tìm index nhỏ nhất mà bit = 1 sau khi hoán vị theo hàm đó)

Với h1(x):

Index gốc	h1(x) = (x+1)%8
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0

→ Theo thứ tự hoán vị [7,0,1,2,3,4,5,6]

- D1: các vị trí có bit = 1: [0,1,2,3] → sau hoán vị: [0,1,2,3]
  → Min index đầu tiên = 0
- D2: [0,1,2,4] → sau hoán vị: [0,1,2,4] → min = 0
- D3: [2,5,6,7] → sau hoán vị: [7,2,5,6] → min = 7

Với h2(x):

Index	h2(x) = (3x + 1)%8
0	1
1	4
2	7
3	2
4	5
5	0
6	3
7	6

→ Hoán vị: [5,0,1,6,3,4,7,2]

- D1: vị trí bit = 1: [0,1,2,3]
  → sau hoán vị: [0,1,2,3] → hoán vị tương ứng: [5,0,1,6] → bit = 1 tại vị trí 0 → giá trị = 0
- D2: [0,1,2,4] → [5,0,1,4] → min = 0
- D3: [2,5,6,7] → [5,6,7] → h2(5)=0 → 5 là index → min = 5

MinHash Signatures:

Tài liệu	h1	h2
D1	0	0
D2	0	0
D3	7	5

- **Bước 4: Dùng LSH để phân cụm**
  - Chia chữ ký thành băng (band): dùng 2 hash signature, chia mỗi band có 1 giá trị (2 bands)
  - Hash mỗi band vào bảng băm:

Band 1 (h1):

Hash value	Documents
0	D1, D2
7	D3

Band 2 (h2):

Hash value	Documents
0	D1, D2
5	D3

Kết quả phân cụm:

- D1 và D2 giống nhau, nằm cùng cụm (vì chung cả 2 band)
- D3 khác, ở cụm riêng

B. Dùng Shingles

Bước 1: Tiền xử lý và tạo Shingles

- **Shingle size (k)**: Chọn k=2 (bigram).
- **Chuyển văn bản thành tập shingles** (bỏ qua khoảng trắng và chữ hoa/thường):

Văn bản	**Shingles (k=2)**
"Machine learning is fun"	{'ma', 'ac', 'ch', 'hi', 'in', 'le', 'ea', 'ar', 'rn', 'ni', 'is', 'sf', 'fu', 'un'}
"Machine learning is hard"	{'ma', 'ac', 'ch', 'hi', 'in', 'le', 'ea', 'ar', 'rn', 'ni', 'is', 'sh', 'ha', 'ar', 'rd'}
"Data structure is easy"	{'da', 'at', 'ta', 'as', 'st', 'tr', 'ru', 'uc', 'ct', 'tu', 'ur', 're', 'is', 'se', 'ea', 'as', 'sy'}

Bước 2: Tạo MinHash Signatures

- **Số lượng hàm băm (**num_perm**)**: Chọn 3 (để minh họa, thực tế dùng 128–256).
- **Giả sử 3 hàm băm** h1, h2, h3 (ví dụ đơn giản):

- h1(x) = (x + 1) % 10
- h2(x) = (2x + 3) % 10
- h3(x) = (3x + 5) % 10
  (Lưu ý: Trong thực tế, dùng hàm băm như MurmurHash.)

Tính MinHash cho từng văn bản:

1. **Mã hóa shingles thành số** (ví dụ: 'ma' → 1, 'ac' → 2, ...).
2. Với mỗi hàm băm, lấy **giá trị băm nhỏ nhất** của các shingles trong văn bản.

Văn bản	**h1(x)**	**h2(x)**	**h3(x)**	**MinHash Signature**
"Machine learning is fun"	min{2,3,4,...} = 1	min{5,7,9,...} = 2	min{8,1,4,...} = 0	[1, 2, 0]
"Machine learning is hard"	min{2,3,4,...} = 1	min{5,7,9,...} = 2	min{8,1,4,...} = 0	[1, 2, 0]
"Data structure is easy"	min{4,5,6,...} = 1	min{9,1,3,...} = 1	min{2,6,0,...} = 0	[1, 1, 0]

(Giá trị cụ thể phụ thuộc vào hàm băm thực tế.)

Bước 3: Áp dụng LSH để phân cụm

- **Ngưỡng tương đồng (**threshold**):** Chọn 0.5 (tức 2/3 signature giống nhau).
- **Chia signature thành các band** (ví dụ: 3 band, mỗi band 1 hàng):
  - Band 1: Hàng 1
  - Band 2: Hàng 2
  - Band 3: Hàng 3

Văn bản	**Band 1**	**Band 2**	**Band 3**
"Machine learning is fun"	1	2	0

Văn bản	Band 1	Band 2	Band 3
"Machine learning is hard"	1	2	0
"Data structure is easy"	1	1	0

- **Nhóm vào cùng bucket nếu ít nhất 1 band trùng nhau**:
    - Bucket (1): "Machine learning is fun", "Machine learning is hard", "Data structure is easy"
      (Vì Band 1 của cả 3 văn bản đều là 1)
    - Bucket (2,0): "Machine learning is fun", "Machine learning is hard"
      (Band 2 + Band 3 trùng)

→ **Kết quả phân cụm**:

- **Cụm 1**: "Machine learning is fun", "Machine learning is hard"
- **Cụm 2**: "Data structure is easy"

(Do 2 văn bản đầu có 2/3 signature giống nhau, vượt ngưỡng 0.5.)

Bước 4: Đánh giá độ tương đồng Jaccard thực tế

- **Tính Jaccard similarity** giữa các tập shingles:
    - Jaccard("fun", "hard") = 9/15 ≈ 0.6
      (9 shingles chung: {'ma','ac','ch','hi','in','le','ea','ar','is'})
    - Jaccard("fun", "easy") = 1/20 ≈ 0.05
    - Jaccard("hard", "easy") = 1/20 ≈ 0.05

→ MinHash + LSH đã phân cụm chính xác các văn bản tương đồng.

Chú ý. Bên cạch jarcard, bạn có thể dùng cosin để tính độ tương tự. Hãy thử nghiệm cả 2 phương pháp này!

Ứng dụng.

- Đầu vào là tập văn bản tiếng Việt
- Nếu n-gram thì chọn 2-4 gram
- Nếu shingles thì chọn 5-10
- Số lượng hàm băm 128–256
- Ngưỡng tương đồng 0.3–0.7 tùy chỉnh

Hãy đưa ra các cụm văn bản dùng độ tương đồng theo jarcard, và cosin.

Chú ý. Tự code lại bằng C/C++, KHÔNG dùng thư viện!