

DATOS MASIVOS I

UNIDAD III MEDIDAS DE SIMILITUD Y DISTANCIA

RESÚMENES DE CONJUNTOS CON PRESERVACIÓN DE SIMILITUD

01 de Marzo de 2023

El Problema del Vecino Más Cercano Aproximado

- La tarea de encontrar vecinos más cercanos es muy común.

El Problema del Vecino Más Cercano Aproximado

- La tarea de encontrar vecinos más cercanos es muy común.
- Se puede pensar en aplicaciones como encontrar documentos duplicados o similares.
- Búsqueda de audio/video.

El Problema del Vecino Más Cercano Aproximado

Posible solución.

- El uso de la fuerza bruta para verificar todas las combinaciones posibles nos dará el vecino más cercano exacto.

El Problema del Vecino Más Cercano Aproximado

Posible solución.

- El uso de la fuerza bruta para verificar todas las combinaciones posibles nos dará el vecino más cercano exacto.

Problema.

- **No es escalable en absoluto.**

El Problema del Vecino Más Cercano Aproximado



Los algoritmos aproximados para lograr esta tarea han sido un área de investigación activa.



Aunque estos algoritmos no garantizan darte la respuesta exacta, la mayoría de las veces proporcionarán una buena aproximación.

- **Estos algoritmos son más rápidos y escalables.**

El Problema del Vecino Más Cercano Aproximado

Dado un conjunto de puntos $\mathcal{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$ y un punto de consulta \mathbf{q} , los cuales residen en un espacio de d dimensiones $\mathbf{x}^{(k)} \in \mathbb{R}^d, i = 1, \dots, n$ bajo una función de distancia $dist$, encontrar los puntos en \mathcal{X} que:

$$dist(\mathbf{q} - \mathbf{x}^{(k)}) \leq (1 + \epsilon) \cdot \min_{\mathbf{x}^{(j)} \in \mathcal{X}} dist(\mathbf{q} - \mathbf{x}^{(j)})$$

donde $\epsilon > 0$ y $\mathbf{x}^{(j)}$ es el verdadero vecino más cercano de \mathbf{q} .

Funciones Hash sensibles a la Localidad (LSH)



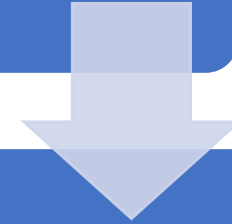
El hashing sensible a la localidad (LSH) es una técnica algorítmica que fragmenta los elementos de entrada similares en los mismos "cubetas" (buckets) con alta probabilidad.



La cantidad de buckets es mucho más pequeña que el universo de posibles elementos de entrada.

Funciones Hash sensibles a la Localidad (LSH)


De modo que los puntos de datos cercanos entre sí se ubicarán en los mismos buckets con alta probabilidad.




Mientras que los puntos de datos lejanos entre sí probablemente estén en cubos diferentes.

Funciones Hash sensibles a la Localidad (LSH)

De modo que los puntos de datos cercanos entre sí se ubicarán en los mismos buckets con alta probabilidad.

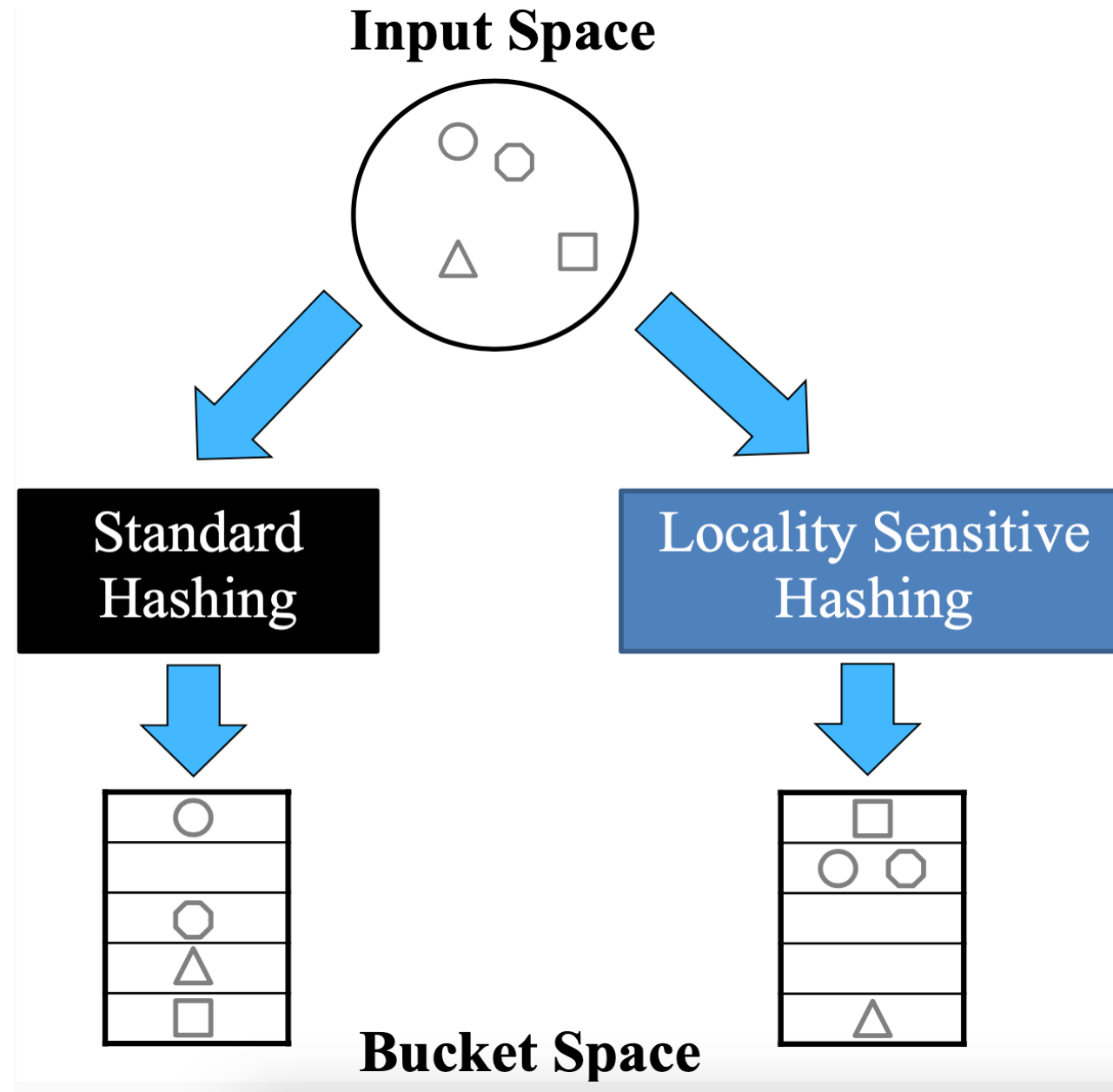


Mientras que los puntos de datos lejanos entre sí probablemente estén en cubos diferentes.



Esto facilita la identificación de observaciones con varios grados de similitud.

Funciones Hash sensibles a la Localidad (LSH)



Funciones Hash sensibles a la Localidad (LSH)

- Dado que los elementos similares terminan en los mismos buckets, esta técnica se puede usar para:

Funciones Hash sensibles a la Localidad (LSH)

- Dado que los elementos similares terminan en los mismos buckets, esta técnica se puede usar para:
 - Agrupación de datos.
 - Búsqueda del vecino más cercano.

Funciones Hash sensibles a la Localidad (LSH)

- Originalmente, LSH se propuso para la búsqueda eficiente de similitudes por pares en conjuntos de datos a gran escala.
- Se diferencia de las técnicas de hash convencionales en que las colisiones de hash se maximizan, no se minimizan.

Funciones Hash sensibles a la Localidad (LSH)

- Alternativamente, la técnica puede verse como una forma de reducir la dimensionalidad de los datos de alta dimensión.
- Los elementos de entrada de alta dimensión se pueden reducir a versiones de baja dimensión mientras se conservan las distancias relativas entre los elementos.

Funciones Hash sensibles a la Localidad (LSH)

- La idea general de LSH es definir una familia adecuada de funciones hash que preserven la similitud para proyectar aleatoriamente el espacio de alta dimensión en un subespacio de menor dimensión, de modo que las distancias entre los elementos se conserven aproximadamente.

Hashing

- La idea de Hashing es convertir cada documento en un espacio más pequeño, usando una función hash H .
- Supongamos que un documento en nuestro corpus se denota con d . Entonces:

Hashing

- Si la similitud $(d1, d2)$ es alta, entonces la probabilidad $(H(d1) == H(d2))$ es alta.
- Si la similitud $(d1, d2)$ es baja, entonces la probabilidad $(H(d1) == H(d2))$ es baja.

Hashing

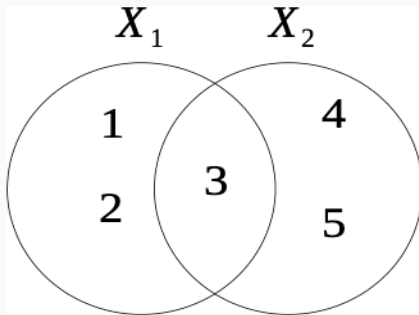
- La elección de la función hash está estrechamente relacionada con la métrica / medida de similitud que estamos usando.
- Por ejemplo, para la similitud de Jaccard, la función hash adecuada es min – hashing.

Min – hashing (Broder 1997)

- Genera permutación aleatoria del conjunto universo \mathbb{U}
- Asigna a cada conjunto su 1er elemento bajo la permutación

$$h(\mathcal{C}^{(1)}) = \min(\pi(\mathcal{C}^{(1)}))$$

- Ejemplo:



$$\pi_1 = \{2, 4, 5, 3, 1\} \longrightarrow (h_1(\mathcal{C}^{(1)}) = 2, h_1(\mathcal{C}^{(2)}) = 4)$$

$$\pi_2 = \{4, 3, 1, 5, 2\} \longrightarrow (h_2(\mathcal{C}^{(1)}) = 3, h_2(\mathcal{C}^{(2)}) = 4)$$

$$\pi_3 = \{3, 1, 4, 2, 5\} \longrightarrow (h_3(\mathcal{C}^{(1)}) = 3, h_3(\mathcal{C}^{(2)}) = 3)$$

$$\pi_4 = \{3, 4, 1, 5, 2\} \longrightarrow (h_4(\mathcal{C}^{(1)}) = 3, h_4(\mathcal{C}^{(2)}) = 3)$$

Min – hashing (Broder 1997)

- Probabilidad de colisión de 2 conjuntos es igual a su **similitud de Jaccard**:

$$P[h(\mathcal{C}^{(1)}) = h(\mathcal{C}^{(2)})] = \frac{|\mathcal{C}^{(1)} \cap \mathcal{C}^{(2)}|}{|\mathcal{C}^{(1)} \cup \mathcal{C}^{(2)}|} \in [0, 1]$$

Ejercicio

- Considera los conjuntos

$$\mathcal{C}^{(1)} = \{1, 2, 5, 7, 9\}$$

$$\mathcal{C}^{(2)} = \{3, 4, 5, 8, 9\}$$

$$\mathcal{C}^{(3)} = \{2, 5, 7, 8\}$$

- Y las permutaciones

$$\pi_1 = \{5, 6, 9, 2, 3, 4, 8, 0, 7, 1\}$$

$$\pi_2 = \{3, 6, 0, 1, 8, 2, 7, 5, 4, 9\}$$

$$\pi_3 = \{9, 2, 6, 7, 4, 3, 1, 8, 5, 0\}$$

$$\pi_4 = \{2, 4, 0, 7, 9, 3, 8, 1, 5, 6\}$$

- Encuentre los valores MinHash para $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$ y $\mathcal{C}^{(3)}$

¿Por qué realizar permutaciones?

¿Qué se obtiene al realizarlo?

Min – hashing (Broder 1997)

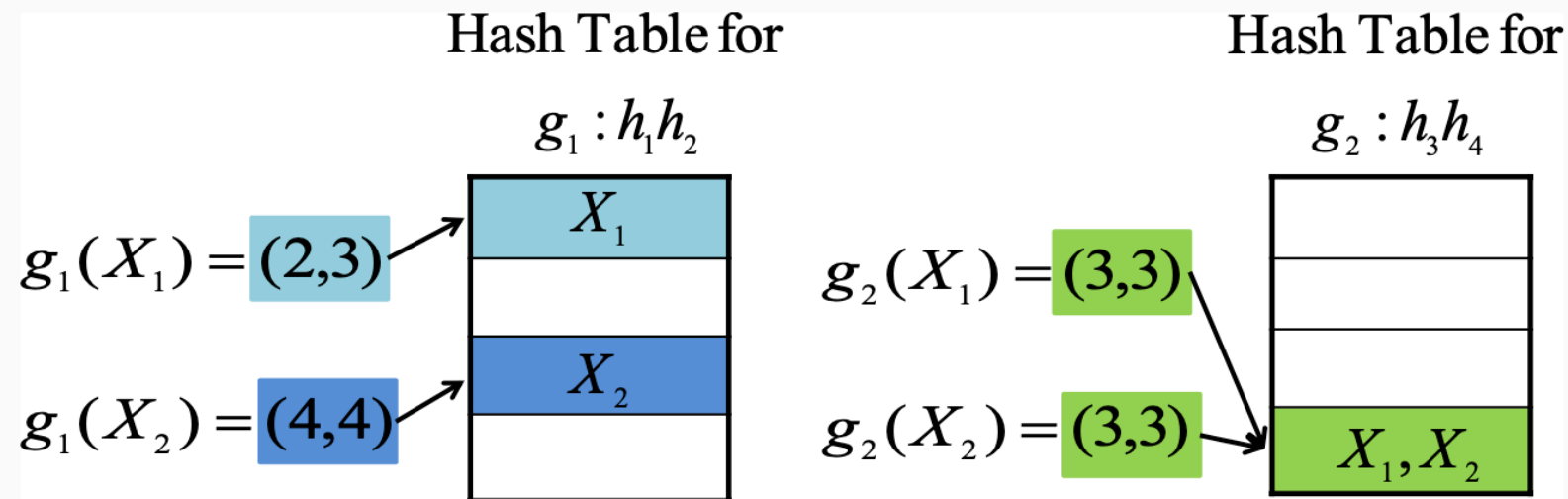
- Min – hashing busca encontrar la similitud de Jaccard sin tener que calcularla intrínsecamente.

- En otras palabras, si r es una variable aleatoria que es 1 cuando $h_{\min}(A) = h_{\min}(B)$ y 0 en otro caso, entonces:
- r es un estimador insesgado de $J(A,B)$, a pesar de que tiene una varianza muy alta para ser útil por sí solo.

- La idea del esquema de Min – Hash es reducir la varianza promediando juntas varias variables construidas de la misma manera.

Min – Hashing para Búsqueda de Conjuntos Similares

- Tuplas de valores MinHash
$$g_1(\mathcal{C}^{(1)}) = (h_1(\mathcal{C}^{(1)}), h_2(\mathcal{C}^{(1)}), \dots, h_r(\mathcal{C}^{(1)}))$$
$$g_2(\mathcal{C}^{(1)}) = (h_{r+1}(\mathcal{C}^{(1)}), h_{r+2}(\mathcal{C}^{(1)}), \dots, h_{2 \cdot r}(\mathcal{C}^{(1)}))$$
$$\dots$$
$$g_l(\mathcal{C}^{(1)}) = (h_{(l-1) \cdot r+1}(\mathcal{C}^{(1)}), h_{(l-1) \cdot r+2}(\mathcal{C}^{(1)}), \dots, h_{l \cdot r}(\mathcal{C}^{(1)}))$$
- Conjuntos con tupla idéntica se almacenan en el mismo registro



Probabilidad de Colisión

- La probabilidad de que los valores MinHash de 2 conjuntos sean idénticos es

$$P[g_k(\mathcal{C}^{(1)}) = g_k(\mathcal{C}^{(2)})] = \text{sim}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r$$

- La probabilidad de que no tengan ninguna tupla idéntica de l posibles es

$$P[g_k(\mathcal{C}^{(1)}) \neq g_k(\mathcal{C}^{(2)})] = (1 - \text{sim}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r)^l, \forall k$$

- Por lo tanto la probabilidad de que 2 conjuntos tengan al menos una tupla idéntica es

$$P_{\text{colisin}}[\mathcal{C}^{(1)}, \mathcal{C}^{(2)}] = 1 - (1 - \text{sim}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r)^l$$

Min – Hashing para Búsqueda de Conjuntos Similares

- Entonces, al usar Min – Hashing, se resuelve el problema de la complejidad del espacio al eliminar la dispersidad y al mismo tiempo preservar la similitud.

Min-Hashing is an LSH scheme in which hash functions are defined with the property that the probability of any pair of sets $\{S_i, S_j\}$, $i, j \in \{1, \dots, N\}$ having the same value is equal to their Jaccard Similarity, i.e.,

$$P[h(S_i) = h(S_j)] = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = J(S_i, S_j) \in [0, 1]. \quad (1)$$

Muestreo Consistente

eral, it has been established that in order for a hash function h to have the property that $P[h(B_i) = h(B_j)] = J_B(B_i, B_j)$, it must be an instance of Consistent Sampling ([Manasse et al., 2010](#)) (see [Definition 3.1](#)).

Definition 3.1 (Consistent Sampling [Manasse et al., 2010](#)). Given a bag B_i with multiplicities $B_i^w, w = 1, \dots, D$, consistent sampling generates a sample $(w, z_w) : 0 \leq z_w \leq B_i^w$ with the following two properties.

1. *Uniformity*: Each sample (w, z_w) should be drawn uniformly at random from $\bigcup_{w=1}^D \{\{w\} \times [0, B_i^w]\}$, where B_i^w is the multiplicity of the element w in B_i . In other words, the probability of drawing w as a sample of B_i is proportional to its multiplicity B_i^w and z_w is uniformly distributed.
2. *Consistency*: If $B_j^w \leq B_i^w, \forall w$, then any sample (w, z_w) drawn from B_i that satisfies $z_w \leq B_j^w$ will also be a sample from B_j .

Min – Hashing para Búsqueda de Relaciones de Orden Mayor

- El esquema Min - Hashing original asume una representación establecida (es decir, la presencia o ausencia de palabras/palabras visuales) de documentos o imágenes que no es adecuado para muchas aplicaciones donde la frecuencia de aparición es importante.
- Por esta razón, se han propuesto extensiones del esquema Min-Hashing original.

Min – Hashing para Búsqueda de Relaciones de Orden Mayor

- La partición del espacio definida por Min – Hashing no solo conserva aproximadamente similitudes por pares, sino también relaciones de orden superior basadas en el coeficiente de co-ocurrencia de Jaccard, una extensión de la similitud de Jaccard.

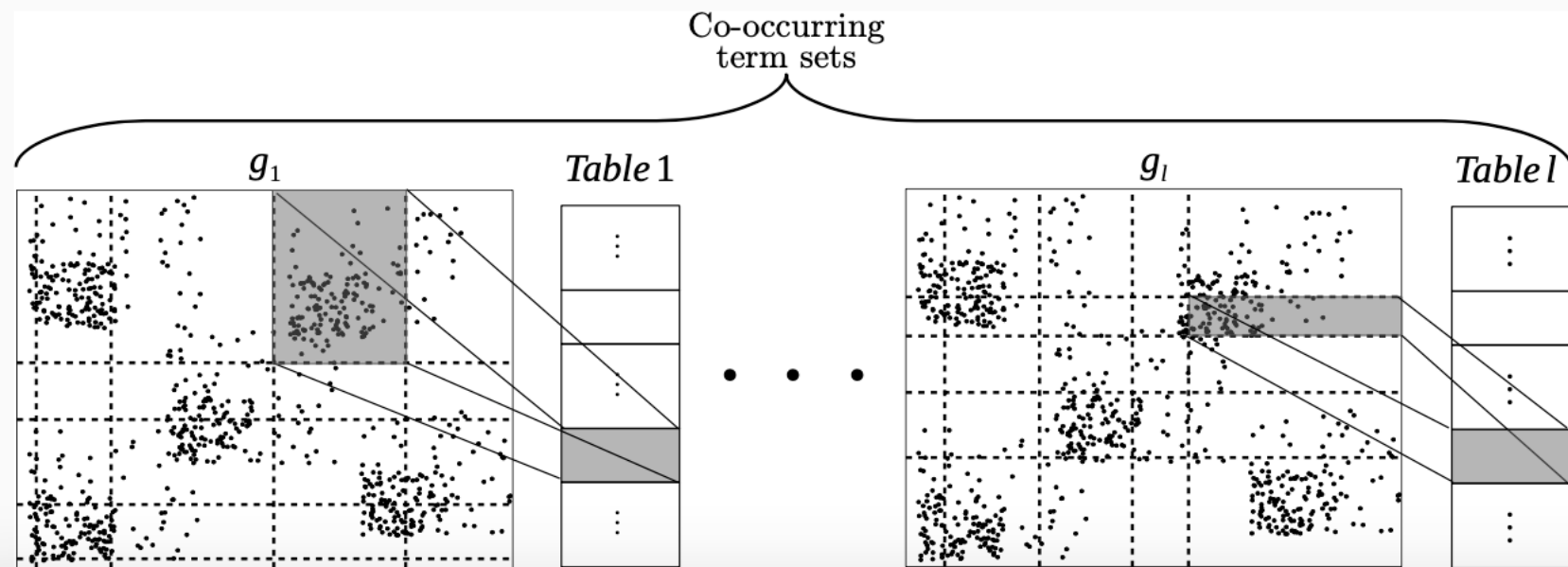
Min – Hashing para Búsqueda de Relaciones de Orden Mayor

- Particiones inducidas por Min-Hashing preservan relaciones de orden mayor dadas por el coeficiente de co-ocurrencia de Jaccard

$$JCC(\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(k)}) = \frac{|\mathcal{C}^{(1)} \cap \mathcal{C}^{(2)} \cap \dots \cap \mathcal{C}^{(k)}|}{|\mathcal{C}^{(1)} \cup \mathcal{C}^{(2)} \cup \dots \cup \mathcal{C}^{(k)}|}$$

Min – Hashing para Búsqueda de Relaciones de Orden Mayor

- Una tupla de *hash* se puede ver como una partición del conjunto universo basada en la co-ocurrencia de sus elementos



- Se plantea la hipótesis de que las palabras que aparecen juntas con frecuencia en el mismo documento en un corpus determinado probablemente pertenezcan al mismo tema y, por lo tanto, podemos descubrir temas aplicando Min – Hashing a las listas de archivos del corpus, que representan ocurrencias de palabras.

Min – Hashing para Búsqueda de Relaciones de Orden Mayor

More generally, we can define the *Jaccard Co-Occurrence Coefficient* for k bags $\{B^{(1)}, \dots, B^{(k)}\} \subseteq \{B_1, \dots, B_N\}$, $k \in 2, \dots, N$ as

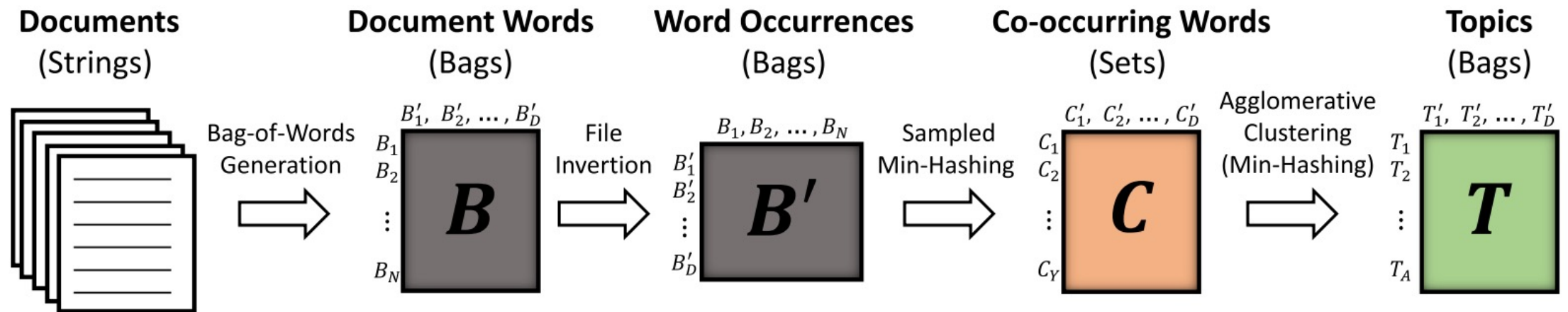
$$JCC_B(B^{(1)}, \dots, B^{(k)}) = \frac{\sum_w \min(B^{(1)w}, \dots, B^{(k)w})}{\sum_w \max(B^{(1)w}, \dots, B^{(k)w})} \in [0, 1], \quad (7)$$

where $B^{(1)w}, \dots, B^{(k)w}$ are the multiplicities of the element w in bags $B^{(1)}, \dots, B^{(k)}$ respectively. From [Definition 3.1](#), it follows that

$$P[h(B^{(1)}) = \dots = h(B^{(k)})] = JCC_B(B^{(1)}, \dots, B^{(k)}), \quad (8)$$

for any hash function h generated with consistent sampling.

Min – Hashing para Búsqueda de Relaciones de Orden Mayor



Min – Hashing para Búsqueda de Relaciones de Orden Mayor

Sample topics discovered by SMH from the 20 Newsgroups, Reuters, and Spanish and English Wikipedia corpora with vocabulary size (D) of 20K, 100K, 1M, and 1M words, respectively. For each topic, its size and the top-10 words are presented.

Size	Top 10 words
20 Newsgroups ($D = 20K$)	
36	religion, atheist, religious, atheism, belief, christian, faith, argument, bear, catholic,...
13	os, cpu, pc, memory, windows, microsoft, price, fast, late, manager,...
31	game, season, team, play, score, minnesota, win, move, league, playoff,...
23	rfc, crypt, cryptography, hash, snefru, verification, communication, privacy, answers, signature,...
45	decision, president, department, justice, attorney, question, official, responsibility, yesterday, conversation,...
19	meter, uars, balloon, ozone, scientific, foot, flight, facility, experiment, atmosphere,...
61	dementia, predisposition, huntington, incurable, ross, forgetfulness, suzanne, alzheimer, worsen, parkinson,...
Reuters ($D = 100K$)	
93	point, index, market, high, stock, close, end, share, trade, rise,...
85	voter, election, poll, party, opinion, prime, seat, candidate, presidential, hold,...
79	play, team, match, game, win, season, cup, couch, final, champion,...
68	wrongful, fujisaki, nicole, acquit, ronald, jury, juror, hiroshi, murder, petrocelli,...
12	window, nt, microsoft, computer, server, software, unix, company, announce, machine,...
28	mexico, mexican, peso, city, state, trade, foreign, year, share, government,...
87	spongiform, encephalopathy, bovine, jakob, creutzfeldt, mad, cow, wasting, cjd, bse,...
Wikipedia Spanish ($D = 1M$)	
179	amerindios, residiendo, afroamericanos, hispanos, isleños, asiáticos, latinos, pertenecían, firme, habkm ² , ...
32	river, plate, juniors, boca, racing, libertadores, clubes, posiciones, lorenzo, rival,...
162	depa, billaba, obiwan, kenobi, padmé, haruun, vaapad, amidala, syndulla, skywalker,...
28	touchdowns, touchdown, quarterback, intercepciones, pases, yardas, nfl, patriots, recepciones, jets,...
58	canción, disco, álbum, canciones, sencillo, unidos, reino, unido, número, música,...
69	poeta, poesía, poemas, poetas, mundo, escribió, literatura, poema, nacional, siglo,...

Funciones Hash sensibles a la Localidad (LSH): Aplicaciones



Nearest neighbor search



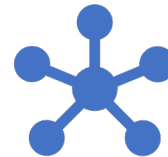
Audio fingerprint



Digital video
fingerprinting



Physical data organization
in database management
systems



Training fully connected
neural networks



Computer security

Funciones Hash sensibles a la Localidad (LSH): Aplicaciones

Near-duplicate detection

Hierarchical clustering

Genome-wide association study

Image similarity identification

- VisualRank

Gene expression similarity identification

Audio similarity identification