NYU Center for Data Science

Week 08.1: Search and MinHash

DS-GA 1004: Big Data

Finding items in a large collection

- **Search** and **recommendation** rely on similarity calculation

- User provides a "**query**", e.g.:
  - Search string
  - Example document
  - Latent representation

- System returns a list of matching "**documents**" from the database

Examples

- **Text search:** search string ⇔ the web (documents)
- **Recommender systems:** user representation ⇔ item representation
- **Reverse image search:** photo ⇔ library
- **Copyright detection:** uploaded video ⇔ all of youtube

Basic approach

- Given a query *q*
- For each document *d* in collection
  - Compute **similarity**(*q*, *d*)
- Order collection by decreasing **similarity**
- Return top *k* documents

How can we do this efficiently?

Problems of scale

- Size of collection (N)
- Dimension of representation
- More generally: complexity of computing similarity
- ***Can we do better than brute force search?***

Approximate search

- Use a **fast method** to identify $n \ll N$ **candidate nearest neighbors**
- Use **true similarity** on **candidate set** to discard any false positives
- **Fast method** usually requires some kind of data structure
  - Search time should be strictly sub-linear: $\boldsymbol{n \sim o(N)}$
  - e.g. **log(*N*)** or **sqrt(*N*)**

MinHash

[Broder, 1997]

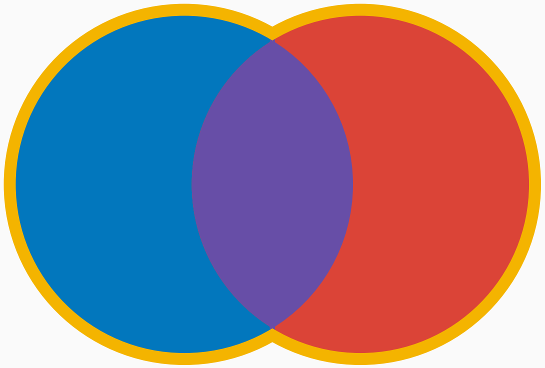
Jaccard similarity for sets

- Items are represented as **sets**, *e.g.*:
  - ***A*** = {words contained in document *A*}
  - ***A*** = {users who interacted with item *A*}

- ***Jaccard similarity*** is the ratio

  $$J(\boldsymbol{A}, \boldsymbol{B}) = |\boldsymbol{A} \cap \boldsymbol{B}| \,/\, |\boldsymbol{A} \cup \boldsymbol{B}|$$

- $D(A, B) = 1 - J(\boldsymbol{A}, \boldsymbol{B})$ is the ***Jaccard distance metric***



MinHash

$\pi(k)$		Doc 1	Doc 2	...
1	Item 3	1	0	
2	Item 75	0	0	
3	Item 21	0	1	
4	Item 1	0	0	
5	Item 2004	0	1	

- Fix a random ordering π of the elements
- Imagine a table of set memberships
- For each set $\boldsymbol{S}$, its hash is:

$$h(\boldsymbol{S} \mid \pi) = \textbf{min}\ \{k \mid \pi(k) \in \boldsymbol{S}\}$$

  - Index of the first (permuted) item belonging to set $\boldsymbol{S}$

$$h(\textbf{Doc 1} \mid \pi) = 1$$

$$h(\textbf{Doc 2} \mid \pi) = 3$$

Permutation indexing

A = {“T.rex”, “Stegosaurus”, “PDP-11”}

B = {“Apples”, “Bananas”, “Pine cones”}

C = {“T.rex”, “Bananas”, “Penguins”}

D = {“Apples”, “Turtles”, “ Pine cones”}

$\pi(k)$		**A**	**B**	**C**	**D**
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	Pine cones	0	1	0	1
4	Turtles	0	0	0	1
5	Apples	0	1	0	1
6	T.rex	1	0	1	0
7	Bananas	0	1	1	0
8	Stegosaurus	1	0	0	0

Permutation indexing

A = {"T.rex", "Stegosaurus", "PDP-11"} $\longrightarrow h(\mathbf{A}|\pi) = 1$

B = {"Apples", "Bananas", "Pine cones"} $\longrightarrow h(\mathbf{B}|\pi) = 3$

C = {"T.rex", "Bananas", "Penguins"} $\longrightarrow h(\mathbf{C}|\pi) = 2$

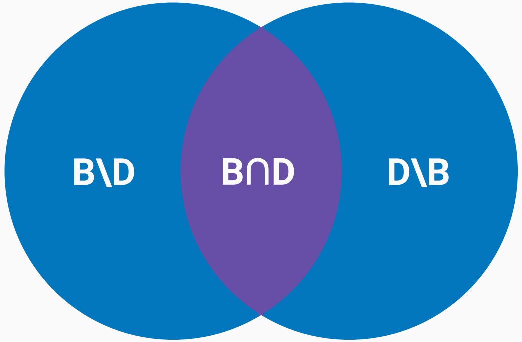
D = {"Apples", "Turtles", " Pine cones"} $\longrightarrow h(\mathbf{D}|\pi) = 3$

Hash collision is more likely when sets overlap.
Let's analyze this!

π(k)		A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	Pine cones	0	1	0	1
4	Turtles	0	0	0	1
5	Apples	0	1	0	1
6	T.rex	1	0	1	0
7	Bananas	0	1	1	0
8	Stegosaurus	1	0	0	0

$P[h(S_1) = h(S_2)] = Jaccard(S_1, S_2)$

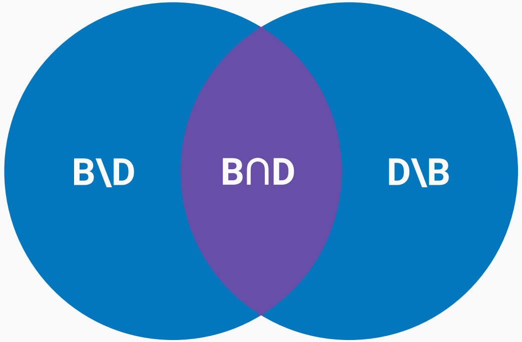
- For two sets S_1 and S_2, there are three types of rows:
  - **Type 1**: $\pi(k) \in S_1 \cap S_2$
  - **Type 2**: $\pi(k) \in S_1 \Delta S_2$
  - Type 3: $\pi(k) \notin S_1 \cup S_2$
- **Collision** ⇔ **type 1 row** occurs before all **type 2 rows**



$\pi(k)$		A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	**Pine cones**	0	**1**	0	**1**
4	**Turtles**	0	**0**	0	**1**
5	**Apples**	0	**1**	0	**1**
6	T.rex	1	0	1	0
7	**Bananas**	0	**1**	1	**0**
8	Stegosaurus	1	0	0	0

$P[h(S_1) = h(S_2)] = \text{Jaccard}(S_1, S_2)$

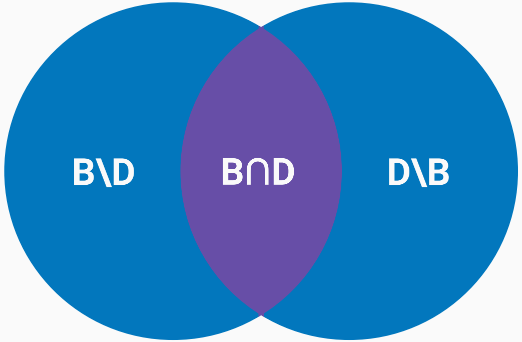
- For two sets S_1 and S_2, there are three types of rows:
  - **Type 1**: $\pi(k) \in S_1 \cap S_2$
  - **Type 2**: $\pi(k) \in S_1 \Delta S_2$
  - Type 3: $\pi(k) \notin S_1 \cup S_2$
- **Collision** ⇔ **type 1 row** occurs before all **type 2 rows**
- P[Collision] = (# Type 1) / (# Type 1 + # Type 2)



	$\pi(k)$	A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	**Pine cones**	0	**1**	0	**1**
4	**Turtles**	0	**0**	0	**1**
5	**Apples**	0	**1**	0	**1**
6	T.rex	1	0	1	0
7	**Bananas**	0	**1**	1	**0**
8	Stegosaurus	1	0	0	0

$P[h(S_1) = h(S_2)] = \text{Jaccard}(S_1, S_2)$

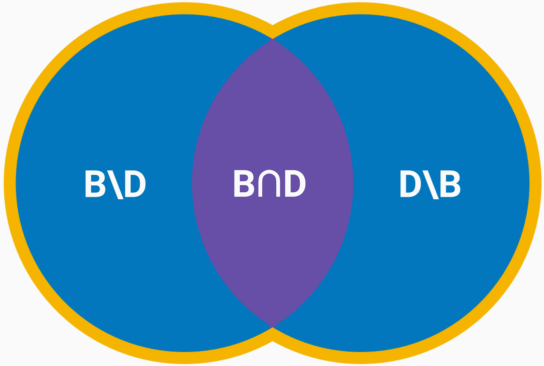
- For two sets S_1 and S_2, there are three types of rows:
  - **Type 1**: $\pi(k) \in S_1 \cap S_2$
  - **Type 2**: $\pi(k) \in S_1 \Delta S_2$
  - Type 3: $\pi(k) \notin S_1 \cup S_2$
- **Collision** ⇔ **type 1 row** occurs before all **type 2 rows**
- P[Collision] $= (\#\text{ Type } 1) / (\#\text{ Type } 1 + \#\text{ Type } 2)$
  $= |S_1 \cap S_2| / (|S_1 \cap S_2| + |S_1 \Delta S_2|)$



π(k)		A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	**Pine cones**	0	**1**	0	**1**
4	**Turtles**	0	**0**	0	**1**
5	**Apples**	0	**1**	0	**1**
6	T.rex	1	0	1	0
7	**Bananas**	0	**1**	1	**0**
8	Stegosaurus	1	0	0	0

$P[h(S_1) = h(S_2)] = \text{Jaccard}(S_1, S_2)$

- For two sets S_1 and S_2, there are three types of rows:
  - **Type 1**: $\pi(k) \in S_1 \cap S_2$
  - **Type 2**: $\pi(k) \in S_1 \Delta S_2$
  - Type 3: $\pi(k) \notin S_1 \cup S_2$



- **Collision** ⇔ **type 1 row** occurs before all **type 2 rows**

- P[**Collision**] $= (\#\text{ Type } 1) / (\#\text{ Type } 1 + \#\text{ Type } 2)$
  $= |S_1 \cap S_2| / (|S_1 \cap S_2| + |S_1 \Delta S_2|)$
  $= |S_1 \cap S_2| / (|S_1 \cup S_2|)$
  $= J(S_1, S_2)$ ■

$\pi(k)$		A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	**Pine cones**	0	**1**	0	**1**
4	**Turtles**	0	**0**	0	**1**
5	**Apples**	0	**1**	0	**1**
6	T.rex	1	0	1	0
7	**Bananas**	0	**1**	1	**0**
8	Stegosaurus	1	0	0	0

Monte carlo approximation

- **B** and **D** had **P[collision]** = ½ for **single permutation** π
- But we want the ***probability of collision*** over **all choices** of π
- **Idea**:
  - Generate m **random permutations** $\pi_1, \pi_2, \ldots, \pi_m$
  - **Count hash collisions** between A and B over all π_i's
  - J(***A***, ***B***) ≈ # collisions / m

π(k)		A	B	C	D
1	PDP-11	1	0	0	0
2	Penguins	0	0	1	0
3	Pine cones	0	1	0	1
4	Turtles	0	0	0	1
5	Apples	0	1	0	1
6	T.rex	1	0	1	0
7	Bananas	0	1	1	0
8	Stegosaurus	1	0	0	0

$h(S \mid \pi_i)$	A	B	C	D
π_1	0	1	2	1
π_2	2	1	2	5
π_3	0	0	3	6
π_4	9	5	1	1
...				

MinHash signatures

Searching with MinHash

$h(S \mid \pi_i)$	A	B	C	D
π_1	0	1	2	1
π_2	2	1	2	5
π_3	0	0	3	6
π_4	9	5	1	1
...				

- User provides query ***q***
- Candidates $\leftarrow \{\}$
- For each π_i:
  - Compute $h(q \mid \pi_i)$
  - Candidates $\leftarrow$ Candidates + **colliding documents** $S : h(q \mid \pi_i) = h(S \mid \pi_i)$
- Return Candidates ordered by
  - # Collisions (estimated similarity score), or
  - Or full Jaccard similarity (more accurate, but slower)

Note that we don't to compare to the full collection!

Only those that collide.

Extension to bags / Ruzicka approximation

[Chen, Philbin, Zisserman 2008]

- Idea: reduce bags to sets by uniquely identifying each repetition

  $\{dog\} \rightarrow \{dog_1\}$

  $\{dog, dog\} \rightarrow \{dog_1, dog_2\}$

  $\{dog, dog, dog\} \rightarrow \{dog_1, dog_2, dog_3\}$

- Jaccard on expanded sets = **Ruzicka similarity** on original bags

$$R(\boldsymbol{A}, \boldsymbol{B}) = \frac{\sum_i \mathbf{min}(\boldsymbol{A}[i], \boldsymbol{B}[i])}{\sum_j \mathbf{max}(\boldsymbol{A}[j], \boldsymbol{B}[j])}$$

Improving over word counts

- Word n-grams:
  - "T.rex stomps loudly" → {"T.rex", "stomps", "loudly", "T.rex stomps", "stomps loudly"}
- Character shingles:
  - "T.rex stomps loudly" → {"T.rex st", ".rex sto", "rex stom", "ex stomp", …}

Efficient approximation

- Permuting the entire universe is **expensive**
    - … and it would not support growing item sets!
- Instead, replace ***permutations*** π_i with ***hashes*** H_i
    - A permutation is a **perfect hash**: distinct elements cannot collide
    - Approximate this by an **imperfect hash**: distinct elements ***may collide***
- As long as H_i are unlikely to collide, this can still work

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	∞	∞	∞	∞
H_2	∞	∞	∞	∞

Signature table is initialized to infinity for each entry

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0 $\cancel{\infty}$	∞	∞	∞
H_2	0 $\cancel{\infty}$	∞	∞	∞

A, H_1: $0 < \infty \rightarrow$ update
A, H_2: $0 < \infty \rightarrow$ update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	∞	1 ~~∞~~	∞
H_2	0	∞	2 ~~∞~~	∞

C, H_1: 1 < ∞ → update
C, H_2: 2 < ∞ → update

MinHash example

x	*$H_1(x)$*	*$H_2(x)$*	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	$2\ \cancel{\infty}$	1	$2\ \cancel{\infty}$
H_2	0	$4\ \cancel{\infty}$	2	$4\ \cancel{\infty}$

B, H_1: $2 < \infty \rightarrow$ update
B, H_2: $4 < \infty \rightarrow$ update
D, H_1: $2 < \infty \rightarrow$ update
D, H_2: $4 < \infty \rightarrow$ update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	2	1	2
H_2	0	4	2	0 ~~4~~

D, H_1: $3 > 2 \longrightarrow$ no update
D, H_2: $0 < 4 \longrightarrow$ update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	0 ~~2~~	1	0 ~~2~~
H_2	0	1 ~~3~~	2	0

B, H_1: $0 < 2 \rightarrow$ update
B, H_2: $1 < 3 \rightarrow$ update
D, H_1: $0 < 2 \rightarrow$ update
D, H_2: $1 > 0 \rightarrow$ no update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0

A, H_1: $1 > 0 \rightarrow$ no update
A, H_2: $3 > 0 \rightarrow$ no update
C, H_1: $1 = 1 \rightarrow$ no update
C, H_2: $3 > 2 \rightarrow$ no update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0

B, H_1: $2 > 0 \rightarrow$ no update
B, H_2: $5 > 1 \rightarrow$ no update
C, H_1: $2 > 1 \rightarrow$ no update
C, H_2: $5 > 2 \rightarrow$ no update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0

A, H_1: $3 > 0 \rightarrow$ no update
A, H_2: $1 > 0 \rightarrow$ no update

MinHash example

x	$H_1(x)$	$H_2(x)$	A	B	C	D
PDP-11	0	0	1			
Penguins	1	2			1	
Pine cones	2	4		1		1
Turtles	3	0				1
Apples	0	1		1		1
T.rex	1	3	1		1	
Bananas	2	5		1	1	
Stegosaurus	3	1	1			

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0

Collisions:

- H_1: **A ≡ B ≡ D ≢ C**
- H_2: **A ≡ D ≢ B ≢ C**

Failure modes of MinHash

- Permutation MinHash:
  - **Collisions** are **more likely** when a **small set of items** are shared across **many documents**
  - ⇒ "Stop-words" can be deadly! *"The", "and", "or", etc...*
- **Hashing approximation** only makes things ***worse***
  - Two distinct items can now hash to the same value
  - Collision probability only increases with the hashing approximation
  - High collision likelihood ⇒ **large candidate sets** ⇒ **slow retrieval**

How can we reduce the size of candidate sets, but maintain high recall?

… come back for part 2!