Finding Similar Items 1

EE412: Foundation of Big Data Analytics

Hw 1

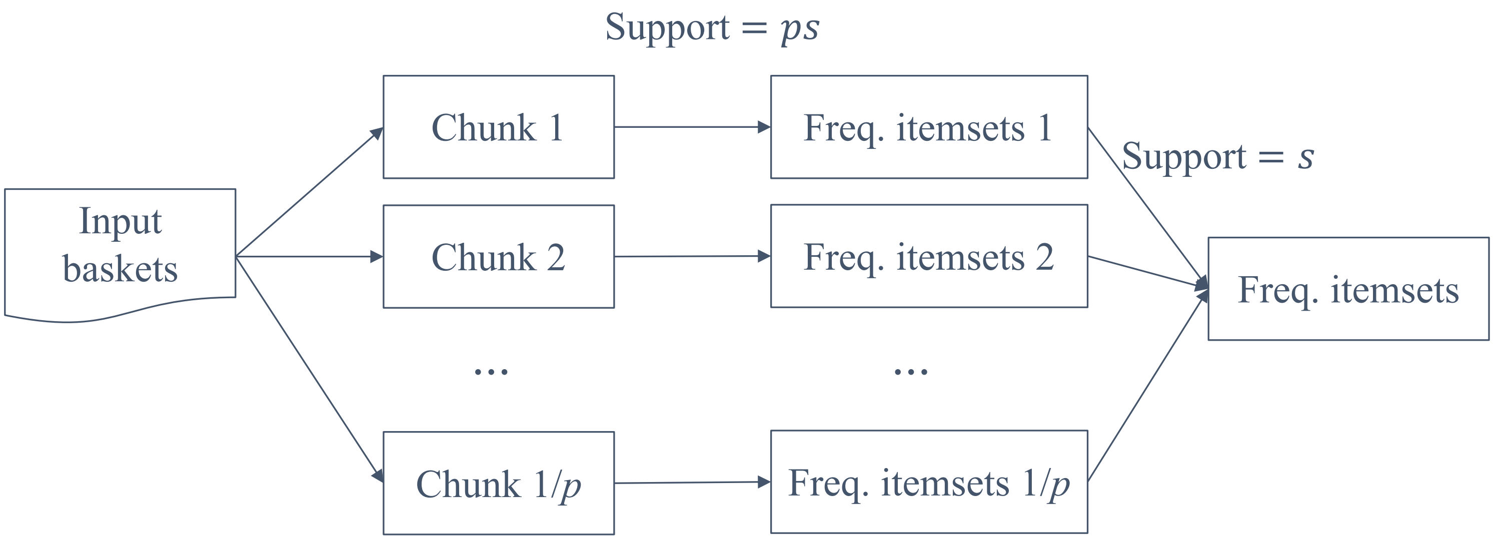
Recap: Frequent Itemsets

- A-Priori algorithm
- PCY Algorithm and extensions
  - Multistage algorithm
  - Multihash algorithm
- Limited-pass algorithms
  - Random sampling
  - Toivonen's algorithm
  - **SON algorithm**

Recap: Limited-Pass Algorithms

- **Many applications can sacrifice accuracy for speed**
  - E.g., enough to find most of the frequent itemsets in supermarket
- We can find all or most frequent itemsets using ≤ 2 passes
  - **Random sampling:** Simplest approach
  - **Toivonen:** Two passes on average, but also may not terminate
  - **SON:** Divide the data into multiple chunks

SON (Savasere, Omiecinski, Navathe)



SON Algorithm

- Avoids both false negatives and positives with two full passes
- **Pass 1:**
  - Divided input file into $1/p$ chunks
  - Run A-Priori with ps as support threshold
- **Pass 2:**
  - Take union of all frequent itemsets and select those with support $\geq s$
- **There are no false negatives**
  - A frequent itemset must be frequent in at least one chunk
  - If an itemset is not frequent in any chunk, its support is $< (1/p)ps \;=\; s$

$s < \frac{1}{p} ps = < s$

$s > s \Rightarrow$ at least in one chunk

$s(S) \geq ps$

SON with MapReduce

- MapReduce (or any other parallel computation) can be used
- **Pass 1:** Find candidate itemsets
  - **Map:** Take a chunk and return $(F, 1)$ for each frequent itemset F
    - Support threshold is ps
  - **Reduce:** Ignore value and produce itemsets that appear at least once
- **Pass 2:** Find true frequent itemsets → search count in whole data
  - **Map:**
    - Take candidate itemsets from Pass 1 and a portion of the input data
    - Return (C, v) where C is a candidate itemset and v is the support for this portion
  - **Reduce:** For each itemset, sum the values and output if the sum $\geq s$

Outline

1. **<u>Finding Similar Items</u>**
2. Shingling
3. Minhashing

Finding Similar Items

- **Example:** Looking at web pages and finding near-duplicate pages
- **Challenge:** What if $O(n^2)$ is infeasible due to too many pages?

The New York Times

Magnitude 6.8 Earthquake Shakes Chile

By The Associated Press

Aug. 1, 2019

SANTIAGO, Chile — A strong earthquake hit off the coast of central Chile on Thursday though there are no immediate reports of damage or injury.

The U.S. Geological Survey said the magnitude 6.8 quake struck at 2:28 p.m. local time (19:28 GMT) and was centered 59 miles (95 kilometers) northwest of San Antonio in the region of Valparaiso.

It was felt throughout the central part of the South American nation, but the national emergency office said there were no immediate reports of damage or injury and the navy discounted any threat of a tsunami.

AP

Magnitude 6.8 earthquake shakes Chile

Associated Press August 2, 2019

Click to copy

RELATED TOPICS

South America
General News
Latin America

SANTIAGO, Chile (AP) — A strong earthquake hit off the coast of central Chile on Thursday though there are no immediate reports of damage or injury.

The U.S. Geological Survey said the magnitude 6.8 quake struck at 2:28 p.m. local time (19:28 GMT) and was centered 59 miles (95 kilometers) northwest of San Antonio in the region of Valparaiso.

It was felt throughout the central part of the South American nation, but the national emergency office said there were no immediate reports of damage or injury and the navy discounted any threat of a tsunami.

Application 1: Finding Similar Documents

- **Example:** Finding textually similar documents in a large corpus
  말뭉치
  - E.g., Web, news articles, etc.
- Focus on character-level similarity instead of “similar meaning”
- Many documents overlap significantly, but are not identical
  - **Plagiarism:** Two homeworks overlap by 50%
  - **Mirror sites:** Pages of mirror sites contain information with their hosts
  - **Similar articles:** Articles of “same story” distributed to many newspapers

Application 2: Collaborative Filtering

- **Example:** Recommend items liked by others with similar tastes
- Online purchases (e.g., Amazon)
  - Two customers are similar if they have similar purchased items
- Movie ratings (e.g., Netflix)
  - Similar to Amazon, but customers watch and rate movies or series

Locality-Sensitive Hashing

approximate algorithm
hash function 잘 정하면 된다.

hopefully linear

- **Magic of Locality-Sensitive Hashing (LSH):**
  - Finds pairs of similar items in a large set without the quadratic cost
- **Basic idea of LSH:**
  - One throws items into buckets using several different hash functions
  - Examine only those pairs of items that share a bucket for at least one time
- **Upside:** Only a small fraction of pairs are ever examined
- **Downside:** False negatives – some pairs never get considered

$D_1 \ \ldots \ D_{100} \Rightarrow 100^2$

h_1 ... h_{10}

10 buckets

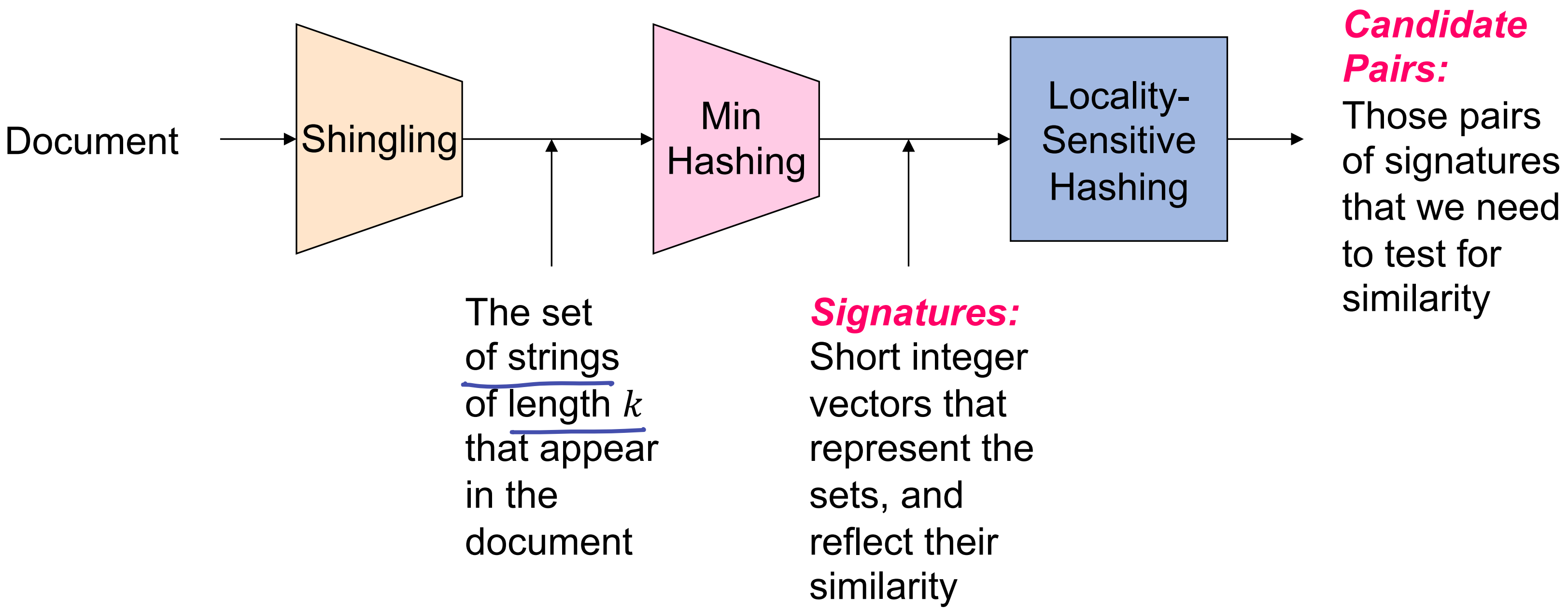
$\rightarrow r_1$
$\rightarrow r_2$
$\rightarrow r_{10}$

combine

Essential Techniques

1. **Shingling:** Convert documents, emails, etc., to sets
2. **Minhashing:** Convert large sets to short signatures (lists of integers)
   - Aim to preserve the similarity between sets
3. **LSH:** Focus on pairs of signatures that are likely to be similar

The Big Picture



Source: Stanford CS246 (2018)

Outline

1. Finding Similar Items
2. **<u>Shingling</u>**
3. Minhashing

Shingling of Documents

- ***k*-Shingles:** Represent a document as a set of strings of length k
  - E.g., for "abcdabd" and $k = 2$, the 2-shingles are {ab, bc, cd, da, bd}
  - Good for identifying lexically similar documents
- Several options for handling white space (blank, tab, newline)
  - E.g., replace any sequence of one or more whitespaces with a single blank

→ single blank로 변경

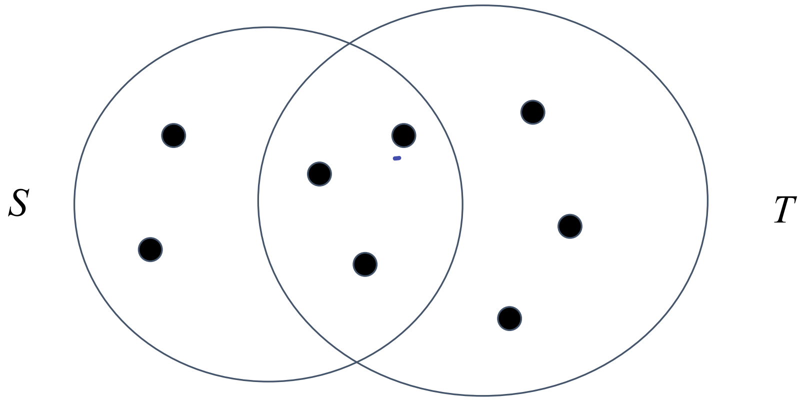
k=9

```
The   plane was ready
for touch down
```

```
{The plane, he plane , e plane w,  plane wa,
 plane was, lane was , ane was r, ne was re,
 ...}
```

Jaccard Similarity between Sets

- Similarity measure of two sets based on relative size of intersection
- **Jaccard similarity** between S and T is $\mathrm{sim}(S, T) = |S \cap T| \,/\, |S \cup T|$
- In the example below, $\mathrm{sim}(S, T) = 3/8$



Shingles as Tokens

- **Idea:** Hash shingles to (say) 4-byte integers (called **tokens**)
  - Document is now a set of bucket numbers of the shingles
- Hashing 9-shingles to 4 bytes is better than using 4-shingles
  - Most sequences of 4 characters are not found in typical documents
    - # of distinct shingles $\ll 2^{32}$
  - If we hash down 9-shingles to 4 bytes, almost all 4 bytes are used

Matrix Representation of Sets

- **Characteristic matrix:** Rows are elements; columns are sets
  - 1 in row e and column S if and only if $e \in S$
  - Jaccard similarity can be computed by the column similarity
  - **Warning:** Not the actual way how data is stored (due to sparsity)

→ sparse (너무 비효율로 저장)

Row (Token)	*Element*	S_1	S_2	S_3
0	"The plane"	1	0	0
1	"The cow j"	0	0	1
2	"Alice and"	0	1	0
3	"Roses are"	1	0	1

$\mathrm{sim}(S_1, S_3) = ?$ $\frac{1}{3}$

num of token

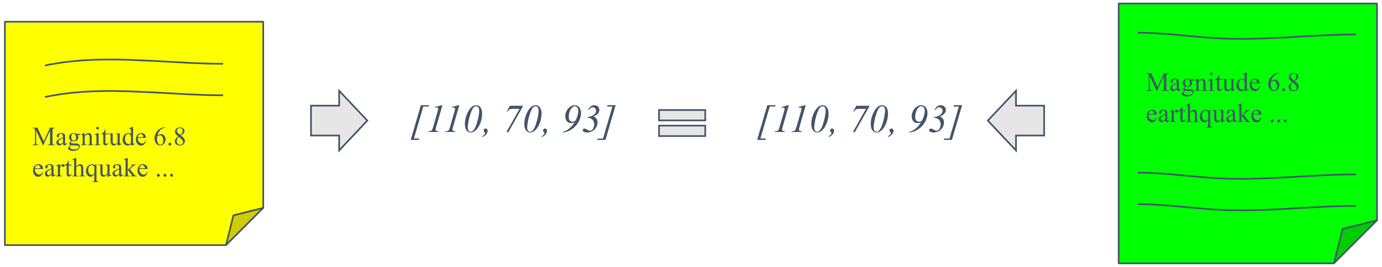
θ x num of shingles " (hashing의 충돌 낮으므로 일단 그건 영향 없다고 가정)

Outline

1. Finding Similar Items
2. Shingling
3. **Minhashing**

Similarity-Preserving Summaries of Sets

- For millions of documents, the characteristic matrix is too large
- **Goal:** Replace large sets by **signatures** (smaller representations)
  - Compare signatures of two sets to estimate the Jaccard similarity
  - The larger the signatures, the more accurate the estimate



Minhashing

- **Minhashing:** Technique to quickly estimate the similarity
  - Permute the rows of the matrix
  - The **minhash value** is the index of the first row that has a 1
- **Signature** is created by applying several random permutations
  - Result is a **signature matrix**
    - Columns are sets; rows are minhash values
    - Normally much smaller than the characteristic matrix

Example: Minhashing

document

1	0	1	1	0
2	0	0	1	1
3	1	0	0	0
4	0	1	0	1
5	0	0	0	1
6	1	1	0	0
7	0	0	1	0

Input Matrix

3	1	1	2

Signature Matrix

Source: Stanford CS246 (2022)

Example: Minhashing

↓ give each random permutation.

7	1	0	1	1	0
6	2	0	0	1	1
5	3	1	0	0	0
4	4	0	1	0	1
3	5	0	0	0	1
2	6	1	1	0	0
1	7	0	0	1	0

Input Matrix

3	1	1	2
2	2	1	3

Signature Matrix

Source: Stanford CS246 (2022)

Example: Minhashing

6	7	1
3	6	2
1	5	3
7	4	4
2	3	5
5	2	6
4	1	7

0	1	1	0
0	0	1	1
1	0	0	0
0	1	0	1
0	0	0	1
1	1	0	0
0	0	1	0

Input Matrix

3	1	1	2
2	2	1	3
1	5	3	2

Signature Matrix

Source: Stanford CS246 (2022)

Subtle Point

- People ask whether the minhash value should be
  - The original number of the row, or
  - The number in the permuted order (as we did in our example)
- **Answer:** It doesn't matter
  - You only need to be consistent
  - Assure that two columns get the same value if and only if
    - Their first 1's in the permuted order are in the same row

Minhashing and Jaccard Similarity

?!

- **Surprising property:**
  - Probability that minhash values are same = Jaccard similarity of sets
- **Proof**
  - Given columns S_1 and S_2, three classes of rows:
    - Type X: 1 in both columns
    - Type Y: 1, 0 (or 0, 1) in columns
    - Type Z: 0 in both columns
  - If there are x rows of Type X and y rows of Type Y,
    - $\mathrm{sim}(S_1, S_2) = x/(x+y)$
    - $\Pr\big(h(S_1) = h(S_2)\big) = \Pr(\text{meeting type X before type Y}) = x/(x+y)$

?

Similarity of Signatures

- The **similarity of signatures** is the fraction of the same rows
  - The **expected similarity of two signatures** equals
  - The **Jaccard similarity** of the sets that the signatures represent
- The longer the signatures, the smaller will be the expected error

Example: Similarity

Columns 1 & 2:
Jaccard similarity 1/4.
Signature similarity 1/3

Columns 2 & 3:
Jaccard similarity 1/5.
Signature similarity 1/3

Columns 3 & 4:
Jaccard similarity 1/5.
Signature similarity 0

0	1	1	0
0	0	1	1
1	0	0	0
0	1	0	1
0	0	0	1
1	1	0	0
0	0	1	0

Input Matrix

3	1	1	2
2	2	1	3
1	5	3	2

Signature Matrix

Source: Stanford CS246 (2022)

Implementation of Minhashing

- Explicit permutation is infeasible in a large characteristic matrix
  - Suppose 1 billion rows
- **Idea to approximate permutation:** each random permutation
  - Apply a random **hash function** h to row indices $r = 1, \dots, n$
  - Sort the rows in order of their hash values $h(r)$
  - The result is considered as random permutation
- **Idea to implement:**
  - Create a "slot" matrix, and initialize all elements to ∞
  - Update each element as we encounter smaller values

Example: Implementation

characteristic matrix

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5$
$g(x) = (2x+1) \bmod 5$

	Sig1	Sig2
$h(1) = 1$	1	∞
$g(1) = 3$	3	∞
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0

Source: Stanford CS246 (2022)

Algorithm

1. let T be a characteristic matrix
2. initialize a slot matrix M
3. for each hash function h_i and column c
4. $\quad M(i, c) \leftarrow \infty$
5. for each row r and column c
6. $\quad$ if $T(r, c) = 1$ then
7. $\quad\quad$ for each hash function h_i
8. $\quad\quad\quad$ if $h_i(r) < M(i, c)$ then
9. $\quad\quad\quad\quad M(i, c) \leftarrow h_i(r)$

Pop Quiz

- **Quiz:** Let's compute the signature matrix in this case

Characteristic matrix

Row	S_1	S_2	S_3	g_1: $x + 1 \bmod 4$	g_2: $3x + 1 \bmod 4$
0	1	0	0	1	1
1	0	0	1	2	0
2	0	1	0	3	3
3	1	0	1	0	2

row 0 .

2	∞	∞
0	∞	∞
0.		0

Signature matrix

Hash fn	S_1	S_2	S_3
g_1	~~∞~~ ~~1~~ 0	~~∞~~ 3	~~∞~~ ~~1~~ ~~2~~ 0
g_2	~~∞~~ 1	~~∞~~ 3	~~∞~~ 0

Summary

??.

0 3 2

1 3 0

2 2

1. Finding Similar Items
2. Shingling
   - Jaccard similarity
   - Characteristic matrix
3. Minhashing
   - Signature matrix
   - Property of Minhashing
   - Simulating permutation