Topic: Duplicate Detection and Similarity Computing

Some of slides are from text book [CMS] and Rajaraman/Ullman

Table of Content

- Motivation
- Shingling for duplicate comparison
- Minhashing
- LSH

Applications of Duplicate Detection and Similarity Computing

- Duplicate and near-duplicate documents occur in many situations
  - Copies, versions, plagiarism, spam, mirror sites
  - Over 30% of the web pages in a large crawl are exact or near duplicates of pages in the other 70%
- Duplicates consume significant resources during crawling, indexing, and search
  - Little value to most users
- Similar query suggestions
- Advertisement: coalition and spam detection

Duplicate Detection

- Exact duplicate detection is relatively easy
  - Content fingerprints
  - MD5, cyclic redundancy check (CRC)
- Checksum techniques
  - A checksum is a value that is computed based on the content of the document
  - e.g., sum of the bytes in the document file

  | T | r | o | p | i | c | a | l | f | i | s | h | S | u | m |
  |---|---|---|---|---|---|---|---|---|---|---|---|---|
  | 54 | 72 | 6F | 70 | 69 | 63 | 61 | 6C | 30 | 66 | 69 | 73 | 68 | 5B |
Near-Duplicate News Articles

Near-Duplicate Detection

- More challenging task
  - Are web pages with same text context but different advertising or format near-duplicates?
- Near-Duplication: Approximate match
  - Compute syntactic similarity with an edit-distance measure
  - Use similarity threshold to detect near-duplicates
    - E.g., Similarity > 80% => Documents are “near duplicates”
    - Not transitive though sometimes used transitively

Near-Duplicate Detection

- Search:
  - Find near-duplicates of a document \( D \)
  - \( O(N) \) comparisons required
- Discovery:
  - Find all pairs of near-duplicate documents in the collection
  - \( O(N^2) \) comparisons
- IR techniques are effective for search scenario
- For discovery, other techniques used to generate compact representations

Two Techniques for Computing Similarity

1. **Shingling**: convert documents, emails, etc., to fingerprint sets.
2. **Minhashing**: convert large sets to short signatures, while preserving similarity.
Fingerprint Generation Process for Web Documents

1. The document is parsed into words. Non-word content, such as punctuation, HTML tags, and additional whitespace, is removed.
2. The words are grouped into contiguous n-grams for some n. These are usually overlapping sequences of words, although some techniques use non-overlapping sequences.
3. Some of the n-grams are selected to represent the document.
4. The selected n-grams are hashed to improve retrieval efficiency and further reduce the size of the representation.
5. The hash values are stored, typically in an inverted index.
6. Documents are compared using overlap of fingerprints.

Computing Similarity with Shingles

- Shingles (Word k-Grams) [Brin95, Brod98]
  “a rose is a rose is a rose” =>
  a_rose_is_a
  rose_is_a_rose
  is_a_rose_is

- Similarity Measure between two docs (= sets of shingles)
  - Size_of_Intersection / Size_of_Union

Example: Jaccard Similarity

- The Jaccard similarity of two sets is the size of their intersection divided by the size of their union.
  \[ \text{Sim}(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}. \]

Example:

- 3 in intersection.
- 8 in union.
- Jaccard similarity = \(3/8\)

Fingerprint Example for Web Documents

- Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.
  - Original text:
    tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species
  - 3-grams:
    938 664 463 822 492 798 78 969 143 136 913 908 694 553 870 779
  - Hash values:
Approximated Representation with Sketching

Computing exact set intersection of shingles between all pairs of documents is expensive

- Approximate using a subset of shingles (called sketch vectors)
  - For doc \( d \), sketch\(_i\) is computed as follows:
    - Let \( f \) map all shingles in the universe to 0..2\(^m\)
    - Let \( \pi_i \) be a specific random permutation on 0..2\(^m\)
    - Pick \( \text{MIN}_i \pi_i(f(s)) \) over all shingles \( s \) in this document \( d \)

Documents which share more than \( t \) (say 80\%) in sketch vector's elements are similar

Computing Sketch\(_i\) for Doc1

Document 1

- Start with 64 bit shingles
- Permute on the number line with \( \pi_i \)
- Pick the min value

Test if Doc1.Sketch\(_i\) = Doc2.Sketch\(_i\)

Test for 200 random permutations: \( \pi_1, \pi_2, \ldots, \pi_{200} \)

Shingling with minhashing

- Given two documents \( d1, d2 \)
- Let \( S1 \) and \( S2 \) be their shingle sets
- Resemblance = \( \frac{|\text{Intersection of } S1 \text{ and } S2|}{|\text{Union of } S1 \text{ and } S2|} \)
- Let \( \alpha = \text{min}(\pi(S1)) \)
- Let \( \beta = \text{min}(\pi(S2)) \)
  - Probability (\( \alpha = \beta \)) = Resemblance
  - Computing this by sampling (e.g. 200 times).
Proof with Boolean Matrices

- **Rows**: elements of the universal set.
- **Columns**: sets.
- 1 in row $e$ and column $S$ if and only if $e$ is a member of $S$.
- Column similarity is the Jaccard similarity of the sets of their rows with 1.
- Typical matrix is sparse.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{sim}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}
\]

Key Observation

- For columns $C_i$, $C_j$, four types of rows:
  
<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$C_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Overload notation: $A = \#$ of rows of type $A$
- Claim

\[
\text{sim}(C_i, C_j) = \frac{A}{A + B + C}
\]

Minhashing

- Imagine the rows permuted randomly.
- “Hash” function $h(C) = \text{index of the first (in the permuted order) row with 1 in column } C$.
- Use several (e.g., 100) independent hash functions to create a signature.
- The **similarity of signatures** is the fraction of the hash functions in which they agree.

Property

- The probability (over all permutations of the rows) that $h(C_1) = h(C_2)$ is the same as $\text{Sim}(C_1, C_2)$.

\[
P[h(C_1) = h(C_2)] = \text{sim}(C_1, C_2)
\]

- Both are $A / (A + B + C)!$
- Why?
  - Look down the permuted columns $C_1$ and $C_2$ until we see a 1.
  - If it’s a type-a row, then $h(C_1) = h(C_2)$. If a type-b or type-c row, then not.
Locality-Sensitive Hashing

All-pair comparison is expensive

- We want to compare objects, finding those pairs that are sufficiently similar.
- comparing the signatures of all pairs of objects is quadratic in the number of objects
- Example: 10^6 objects implies 5*10^{11} comparisons.
  - At 1 microsecond/comparison: 6 days.

The Big Picture

Locality-Sensitive Hashing

- General idea: Use a function f(x,y) that tells whether or not x and y is a candidate pair: a pair of elements whose similarity must be evaluated.
- Map a document to many buckets
  - Make elements of the same bucket candidate pairs.
Another view of LSH: Produce signature with bands

One short signature

b bands

r rows per band

Signature

Signature agreement of each pair at each band

Agreement? Mapped into the same bucket?

b bands

r rows per band

Signature generation and bucket comparison

- Create \( b \) bands for each document
  - Signature of doc X and Y in the same band agrees \( \rightarrow \) a candidate pair
  - Use \( r \) minhash values (\( r \) rows) for each band
- Tune \( b \) and \( r \) to catch most similar pairs, but few nonsimilar pairs.

Docs 2 and 6 are probably identical.

Docs 6 and 7 are surely different.

b bands

r rows

Matrix M

Buckets
Analysis of LSH

- Probability the minhash signatures of $C_1$, $C_2$ agree in one row: $s$
  - Threshold of two similar documents
- Probability $C_1$, $C_2$ identical in one band: $s'$
- Probability $C_1$, $C_2$ do not agree at least one row of a band: $1 - s'$
- Probability $C_1$, $C_2$ do not agree in all bands: $(1 - s')^b$
  - False negative probability
- Probability $C_1$, $C_2$ agree one of these bands: $1 - (1 - s')^b$
  - Probability that we find such a pair.

Example

- Suppose $C_1$, $C_2$ are 80% Similar
- Choose 20 bands of 5 integers/band.
- Probability $C_1$, $C_2$ identical in one particular band: $(0.8)^5 = 0.328$.
- Probability $C_1$, $C_2$ are not similar in any of the 20 bands: $(1 - 0.328)^{20} = 0.00035$.
  - i.e., about 1/3000th of the 80%-similar column pairs are false negatives.

Analysis of LSH – What We Want

![Diagram showing the probability of sharing a bucket given similarity $s$.]

What One Band Gives You

![Diagram showing the probability of sharing a bucket given similarity $s$.]
**What b Bands of r Rows Gives You**

![Diagram](Image)

**Example: b = 20; r = 5**

<table>
<thead>
<tr>
<th>s</th>
<th>1-(1-s^r)^b</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.006</td>
</tr>
<tr>
<td>.3</td>
<td>.047</td>
</tr>
<tr>
<td>.4</td>
<td>.186</td>
</tr>
<tr>
<td>.5</td>
<td>.470</td>
</tr>
<tr>
<td>.6</td>
<td>.802</td>
</tr>
<tr>
<td>.7</td>
<td>.975</td>
</tr>
<tr>
<td>.8</td>
<td>.9996</td>
</tr>
</tbody>
</table>

**Probability of a similar pair to share a bucket**

**LSH Summary**

- **Get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.**
  - Check that candidate pairs really do have similar signatures.
- **LSH involves tradeoff**
  - Pick the number of minhashes, the number of bands, and the number of rows per band to balance false positives/negatives.
  - **Example:** if we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up.