Finding near-duplicate documents

Finding duplicate or near duplicate documents

A general paradigm:

1. Define function *f* capturing contents of each document in one number
   - "Hash function", "signature", "fingerprint"
2. Create <*f*(doc_i), ID of doc_i> pairs
3. Sort the pairs
4. Recognize duplicate or near-duplicate documents as having the same *f* value or *f* values within a small threshold

Compare: computing a similarity score on pairs of documents

Finding duplicate or near duplicate documents

A general paradigm:

1. Define function *f* capturing contents of each document in one number
   "Hash function", "signature", "sketch", "fingerprint"
2. Create <*f*(doc_i), ID of doc_i> pairs
3. Sort the pairs
4. Recognize duplicate or near-duplicate documents as having the same *f* value or *f* values within a small threshold

Problem with "small threshold" ?

Finding duplicate or near duplicate documents

A general paradigm:

1. Define function *f* capturing contents of each document in one number
   "Hash function", "signature", "sketch", "fingerprint"
2. Create <*f*(doc_i), ID of doc_i> pairs
3. Sort the pairs
4. Recognize duplicate or near-duplicate documents as having the same *f* value or *f* values within a small threshold

Problem with "small threshold" ?

How deal with <1, D_1> <1.01, D_2> <1.02, D_3> …..<1.99, D_{100}> and threshold .01 (using ≤ threshold) ?

"Syntactic clustering"

We will look at this one example:

Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig, Syntactic Clustering of the Web *Sixth International WWW Conference*, 1997.

- "syntactic similarity" versus semantic
  - Sequences of words
- Finding near duplicates
- Doc = sequence of words
  - Word = Token
- Uses **sampling**
- Similarity based on **shingles**
- Does compare documents

Shingles

- A *w*-shingle is a contiguous subsequence of *w* words
- The *w*-shingling of doc D, S(D, *w*) is the set of unique w-shingles of D

Similarity of docs with shingles

For **fixed *w*,** resemblance of docs A and B :

$r(A, B) = |S(A) \cap S(B)| \quad / \quad |S(A) \cup S(B)|$

For **fixed *w*,** containment of doc A in doc B :

$C(A, B) = |S(A) \cap S(B)| \quad / \quad |S(A)|$

For **fixed *w*,** resemblance distance between docs A and B :

$D(A, B) = 1 - r(A, B)$

Is a metric (triangle inequality)

Note we are now comparing documents!

Sketch of shingles

Want to *estimate* r and/or c

Do this by calculating approximation on a sample of shingles

- Fix *w;* use some ordering to totally order all possible w-shingles (e.g. convert *w*-shingle to integer)
- For any **W** a set of w-shingles define:

  $MIN_s(\mathbf{W})$ = Set of s smallest shingles in **W** if $|\mathbf{W}| \geq s$
  W if $|\mathbf{W}| < s$

  And if actually interpret each shingle as an integer:
  $MOD_m(\mathbf{W})$ = set of shingles *x* in W with ($x \equiv 0 \bmod m$)

Sketch of shingles continued

Let Π be a permutation of all *w*-shingles (reorder),
For any doc A and chosen integer parameters s and m

We define sketches :

$F(A) = MIN_s(\Pi(S(A)))$ s random shingles of A

$V(A) = MOD_m(\Pi(S(A)))$

random sample of shingles of A with size depending on length of A

Sketch is a ***sampling***

Mathematics

Theorem:

If Π is chosen uniformly at random then for fixed *w*, s and m and any two docs A and B:

r(A, B) has the unbiased estimates:

1. $|MIN_s(F(A) \cup F(B)) \cap F(A) \cap F(B)| \;/\; |MIN_s(F(A) \cup F(B))|$
2. $|V(A) \cap V(B)| \;/\; |V(A) \cup V(B)|$

c(A, B) has the unbiased estimate:

1. $|V(A) \cap V(B)| \;/\; |V(A)|$

$r(A, B) = |S(A) \cap S(B)| \;/\; |S(A) \cup S(B)|$

$C(A, B) = |S(A) \cap S(B)| \;/\; |S(A)|$

Algorithm used

1. Calculate *sketch* $V(D_i)$ for every doc D_i
   Denote $| V(D_i) |$ by ct_i
2. Calculate $|V(D_i) \cap V(D_j)| = ct_{ij}$ for each non-empty intersection:
   i. Produce list of <shingle value, docID> pairs for all shingles in the sketch for each document
   ii. Sort the list by shingle value
   iii. Produce all triples $<ID(D_i), ID(D_j), ct_{i,j}>$ for which $ct_{i,j}>0$
   Note this is not linear-time for the list of docs for one shingle
3. Build clusters of similar/almost identical docs
   Degree of similarity depends on threshold …

Clustering

1. Define docs to be similar if approximate resemblance greater than a predetermined threshold *t*:
   $ct_{ij} / (ct_i + ct_j - ct_{ij}) > t$
2. Build graph of docs: edge between each pair of similar docs
3. The clusters of similar docs are the connected components in the graph

Paradigm?

- Does compare docs, so not same as paradigm we started with, but uses ideas
- Contents of doc captured by sketch – a set of shingles (numbers)
- Similarity of docs scored by count of common shingles for docs
- Don't look at all doc pairs, look at all doc pairs that share a shingle
- Uses clustering by similarity threshold

More efficient : super-shingles

"meta-sketch"

1. Sort shingles of a sketch
2. Compute the shingling of the sequence of shingles
   Each original shingle now a token
   Gives "supershingles"
3. "meta-sketch" = set of supershingles

One supershingle in common =>
sequences of shingles in common
Documents with ≥1 supershingle in common considered similar

- Each supershingle for a doc. characterizes the doc
- Sort <supershingle, docID> pairs: docs sharing a supershingle are similar => our first paradigm

Pros and Cons of Super-shingles

+ Faster
- Problems with small documents – not enough shingles
- Can't do containment

  Shingles of superset that are not in subset break up sequence of shingles

Experiments (1996)

- 30 million HTML and text docs (150GB) from crawl of Web
- 10-word shingles
- 40-bit shingle "fingerprints" for random permutation
- Sample shingles using mod m=25
- 600 million shingles (3GB)
- Used count of shingles for similarity
- Max space used: 20GB when counting shared shingles
- Using threshold t = 50%, found
  - 3.6 million clusters of 12.3 million docs
  - 2.1 million clusters of identical docs – 5.3 million docs