Fast Detection of Near Duplicates

Sambuddha Roy, LinkedIn

July 2, 2016

Motivation.

and some more...

and more...

The duplicate detection problem.

- Why are duplicates problematic?

The duplicate detection problem.

- Why are duplicates problematic?
- Imagine if...

The duplicate detection problem.

- Why are duplicates problematic?
- Imagine if...
- Youtube had multiple copies of each video!
  - Someone copies the cat video that *you* uploaded!

The duplicate detection problem.

- Why are duplicates problematic?
- Imagine if...
- Youtube had multiple copies of each video!
  - Someone copies the cat video that *you* uploaded!
- Duplicate content confuses search engines.
  - What happens to the pageRank of the page?

How do you catch duplicates?

- Some form of hashing...
- Different items go to different hashes (collision resistant).
- Group by hashes - each group corresponds to a distinct element.

How about near-duplicates?

- Now imagine if...

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures,

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures, connected to the same friends,

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures, connected to the same friends, indicated same interests

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures, connected to the same friends, indicated same interests etc.

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures, connected to the same friends, indicated same interests etc.
- Identity theft!

How about near-duplicates?

- Now imagine if...
- Someone made a profile on Facebook, very similar to yours.
- copied profile pictures, connected to the same friends, indicated same interests etc.
- Identity theft!
- Other use cases: plagiarism, etc.

Essential Problem: Nearest Neighbor Search

- There is a database of items $\mathcal{D}$.
- ...and a query q arrives.

Essential Problem: Nearest Neighbor Search

- There is a database of items $\mathcal{D}$.
- ...and a query q arrives.
- Given the query q, find the nearest neighbors of q in the database $\mathcal{D}$.

Essential Problem: Nearest Neighbor Search

- There is a database of items $\mathcal{D}$.
- ...and a query q arrives.
- Given the query q, find the nearest neighbors of q in the database $\mathcal{D}$.
- Often, we just want the k nearest neighbors (k-NN problem).

Nearness?

- Nearness only makes sense in the presence of a *distance measure*.

Nearness?

- Nearness only makes sense in the presence of a *distance measure*.
- So, similarity measures between objects/items...
- i.e. *featurize* items

Nearness?

- Nearness only makes sense in the presence of a *distance measure*.
- So, similarity measures between objects/items...
- i.e. *featurize* items as vectors $\in \mathbb{R}^n$ or in $\{0, 1\}^n$.

Nearness?

- Nearness only makes sense in the presence of a *distance measure*.
- So, similarity measures between objects/items...
- i.e. *featurize* items as vectors $\in \mathbb{R}^n$ or in $\{0, 1\}^n$.
- And consider some distance/similarity measure between these vectors.

Typical Distance/Similarity Measures

- Jaccard similarity: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Typical Distance/Similarity Measures

- Jaccard similarity: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- Cosine Similarity etc.: $\mathrm{sim}(a, b) = \frac{a \cdot b}{|a||b|}$

Typical Distance/Similarity Measures

- Jaccard similarity: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- Cosine Similarity etc.: $\mathrm{sim}(a, b) = \frac{a \cdot b}{|a||b|}$
- Hamming Distance
- ℓ_p for $p \in (0, 2]$.
- Tanimoto distance, Mahalanobis distance, etc.

Typical Distance/Similarity Measures

- Jaccard similarity: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- Cosine Similarity etc.: $\mathrm{sim}(a, b) = \frac{a \cdot b}{|a||b|}$
- Hamming Distance
- ℓ_p for $p \in (0, 2]$.
- Tanimoto distance, Mahalanobis distance, etc.
- It's typical to relate the distance and similarity measures as $\mathrm{dist}(a, b) = 1 - \mathrm{sim}(a, b)$.

Back to what we want...

- Given a *query* object q, we would want to retrieve all the *database* items that are "similar" to q.

Back to what we want...

- Given a *query* object q, we would want to retrieve all the *database* items that are "similar" to q.
- A search by pairwise comparisons between q and all the items in the database may become too costly.

Another use-case...

- If we want to “cluster” the items in the database according to their similarities/distances.

Another use-case...

- If we want to "cluster" the items in the database according to their similarities/distances.
- (Aside: what does this even mean? It may be for a threshold τ and points a, b, c that $\mathrm{dist}(a, b) \leqslant \kappa$ and $\mathrm{dist}(b, c) \leqslant \kappa$ but $\mathrm{dist}(a, c) > \kappa$ i.e. similarity is not "transitive").

Similarity is not transitive

- A more meaningful formulation can be: we want all *pairs* of items a, b such that $\mathrm{sim}(a, b) \geqslant \tau$ for some $\tau \in [0, 1]$ (sufficiently similar items).

Similarity is not transitive

- A more meaningful formulation can be: we want all *pairs* of items a, b such that $\mathrm{sim}(a, b) \geqslant \tau$ for some $\tau \in [0, 1]$ (sufficiently similar items).
- If we make $\tau = 0$ we are asking for all of the $\binom{n}{2}$ pairs of items (for n items in the database).

Similarity is not transitive

- A more meaningful formulation can be: we want all *pairs* of items a, b such that $\mathrm{sim}(a, b) \geqslant \tau$ for some $\tau \in [0, 1]$ (sufficiently similar items).
- If we make $\tau = 0$ we are asking for all of the $\binom{n}{2}$ pairs of items (for n items in the database).
- Imagine if the number of items n were

Similarity is not transitive

- A more meaningful formulation can be: we want all *pairs* of items a, b such that $\mathrm{sim}(a, b) \geqslant \tau$ for some $\tau \in [0, 1]$ (sufficiently similar items).
- If we make $\tau = 0$ we are asking for all of the $\binom{n}{2}$ pairs of items (for n items in the database).
- Imagine if the number of items n were 1 million...

Similarity is not transitive

- A more meaningful formulation can be: we want all *pairs* of items a, b such that $\mathrm{sim}(a, b) \geqslant \tau$ for some $\tau \in [0, 1]$ (sufficiently similar items).
- If we make $\tau = 0$ we are asking for all of the $\binom{n}{2}$ pairs of items (for n items in the database).
- Imagine if the number of items n were 1 million... 1 billion...!

Alternate Formulation/Relaxations

- We really do not want all pairs for the similarity threshold, τ really small; in fact, typical use-cases will consider $\tau > 0.8$ or so (sufficiently similar).
- Let's also relax the all in the above too; replace that by most.

Refining what we want

- Find near-duplicates of query items.
- Some mistakes will be allowed (both false positives, false negatives).
- Time! Querying should be fast. The clustering variant should take $O(n)$ time instead of $O(n^2)$.

Hashing to the rescue

- Can we hash items so that “nearby” items are in same hash-buckets?
- Note that this is counter to the usual notion of hashing, where collisions are taboo.
- Here, we would like collisions - but only between *nearby* items.

Enter LSH

- Locality Sensitive Hashing - introduced by Indyk & Motwani, in 1998.
  - Introduced concept
  - exhibited LSH for Hamming Distance.

Enter LSH

- Locality Sensitive Hashing - introduced by Indyk & Motwani, in 1998.
  - Introduced concept
  - exhibited LSH for Hamming Distance.
- MinHash - by Broder, Charikar, Frieze, Mitzenmacher, 1998.
  - Introduced min-wise permutations.
  - LSH for Jaccard similarity.

Enter LSH

- Locality Sensitive Hashing - introduced by Indyk & Motwani, in 1998.
  - Introduced concept
  - exhibited LSH for Hamming Distance.
- MinHash - by Broder, Charikar, Frieze, Mitzenmacher, 1998.
  - Introduced min-wise permutations.
  - LSH for Jaccard similarity.
- SimHash - by Charikar, 2002.
  - Demonstrated connections between randomized rounding and LSH
  - LSH for cosine similarity (angular distance).

Formal Definition

- A family of hash functions $\mathcal{F}$ is a LSH if for *any* x, y the following holds:
$$\Pr_{h \in \mathcal{F}}[h(x) = h(y)] = \mathrm{sim}(x, y)$$
where $\mathrm{sim}(x, y)$ is a similarity measure.
- Read as: items that are “highly” similar land in the same hash-bucket with “high” probability, and items that are dissimilar land in the same hash bucket with “low” probability.

LSH *for angular distance*

- Given two vectors a, b, construct a hash function that maps these to the same bucket if the angle θ between them is *small*.
- Idea: use a random hyperplane (random projection).

LSH *for angular distance*

- Given two vectors a, b, construct a hash function that maps these to the same bucket if the angle θ between them is *small*.
- Idea: use a random hyperplane (random projection).

LSH *for angular distance*

- Given two vectors a, b, construct a hash function that maps these to the same bucket if the angle θ between them is *small*.
- Idea: use a random hyperplane (random projection).

LSH *for angular distance*

- This random hyperplane is a hash function $h : \mathbb{R}^n \rightarrow \{0, 1\}$.
- It maps a vector v to $\{0, 1\}$, depending on whether the vector v is to the top/bottom of the hyperplane (essentially, checking for the sign of the inner product).

Probability calculation

- Probability that vectors a, b map to the same output, i.e. $\Pr[h(a) = h(b)]$?

Probability calculation

- Probability that vectors a, b map to the same output, i.e. $\Pr[h(a) = h(b)]$?
- If θ be the angle between a, b, then this equals $\left[1 - \frac{\theta}{\pi}\right]$.

Precision? Recall?

- With a single hash function *h*, the precision may be quite low; the recall quite high.
- Just one hash may not be able to detect vectors that are indeed close by. So...

Precision? Recall?

- With a single hash function h, the precision may be quite low; the recall quite high.
- Just one hash may not be able to detect vectors that are indeed close by. So...
- Several (independent) copies of the hash function h; call these $h_1, h_2, \cdots, h_k$.

Precision? Recall?

- With k hash functions, we have a k-bit binary string, corresponding to a vector v.
- Call this the *hashcode* corresponding to the vector v (denoted as (v)).
- For two vectors a, b: if $(a) = (b)$ bit-by-bit, then surely a and b are close.

Precision? Recall?

- With k hash functions, we have a k-bit binary string, corresponding to a vector v.
- Call this the *hashcode* corresponding to the vector v (denoted as (v)).
- For two vectors a, b: if $(a) = (b)$ bit-by-bit, then surely a and b are close.
- This would help in improving precision. But recall may suffer.

Precision? Recall? Two extremes

- Given the hashcodes of the vectors, we can ask for a full bit-by-bit match to declare near duplicates. High precision, low recall.
- Given the hashcodes, we can ask for a *single* bit match to declare near duplicates. High recall, low precision.

Precision? Recall? Two extremes

- Given the hashcodes of the vectors, we can ask for a full bit-by-bit match to declare near duplicates. High precision, low recall.
- Given the hashcodes, we can ask for a *single* bit match to declare near duplicates. High recall, low precision.
- Mix the two: Banding. In the parlance of complexity theory, gap amplification.

Banding

- An example with $k = 15$ hashes, where $\mathrm{rows} = 3$ and $\mathrm{bands} = 5$.

Banding

- An example with $k = 15$ hashes, where $\mathrm{rows} = 3$ and $\mathrm{bands} = 5$.

Banding

- An example with $k = 15$ hashes, where $\mathrm{rows} = 3$ and $\mathrm{bands} = 5$.

Are a and b near-duplicates?

- We show the ’s for *a* and *b*:

Are a and b near-duplicates?

- We show the ’s for *a* and *b*:

Are a and b near-duplicates?

- We show the ’s for *a* and *b*:

Are a and b near-duplicates?

- We declare *a* and *b* as near duplicates, if there is a *band* in which they match bit-by-bit.

Are a and b near-duplicates?

- We declare *a* and *b* as near duplicates, if there is a *band* in which they match bit-by-bit.

- To increase precision, increase the number of rows. To increase recall, increase number of bands.

Still a flourishing/active area of research

- Various considerations:
  - Engineering aspects: hash table construction time, query times, etc.
  - Engineering aspects: Maintain hashbuckets, update hashes. Bloom filters for hash buckets, etc.
  - How much randomness do I need?
  - How do I improve recall while maintaining precision.
  - Deep Hashing techniques?
  - Which similarity measures work best for different content: text, images, video.

Still a flourishing/active area of research

- Improve *recall* of LSH
  - covering LSH (only for Hamming space), (Pham-Pagh16)
  - other similarity measures wide open.
- Develop LSH's for other similarity measures.
  - Inner product (Neyshabur-Srebro15, Li-Shrivastava14)
  - Also gave rise to Assymetric LSH.
- Improve training and query times based on data:
  - Data Dependent Hashing (Andoni-Razenshteyn15)
  - Learning to hash

LinkedIn's interest in finding near-duplicates

- Do we like memes on our LinkedIn page? Puzzles?

- Spam Filtering: spammers often use the same text repeatedly to *spam* members.

Thank You!

Thank You!

Questions?

Formal definition of LSH

- A hash is said to be a (S, cS, p_1, p_2)-LSH for a similarity function sim over the space $\mathcal{X}$ if for any $x, y \in \mathcal{X}$:
  - if $\mathrm{sim}(x, y) \geqslant S$ then $\Pr[h(x) = h(y)] \geqslant p_1$.
  - if $\mathrm{sim}(x, y) \leqslant cS$ then $\Pr[h(x) = h(y)] \leqslant p_2$.

  Here, $c \in (0, 1)$

Formal definition of LSH

- A hash is said to be a (S, cS, p_1, p_2)-LSH for a similarity function sim over the space $\mathcal{X}$ if for any $x, y \in \mathcal{X}$:
  - if $\mathrm{sim}(x, y) \geqslant S$ then $\Pr[h(x) = h(y)] \geqslant p_1$.
  - if $\mathrm{sim}(x, y) \leqslant cS$ then $\Pr[h(x) = h(y)] \leqslant p_2$.

  Here, $c \in (0, 1)$
- Read as: items that are "highly" similar land in the same hash-bucket with "high" probability, and items that are dissimilar land in the same hash bucket with "low" probability.
- Ideally we want: $p_1 = 1$, and $p_2 = 0$. These probabilities "mirror" the similarity function $\mathrm{sim}(\cdot, \cdot)$.