Finding Similar Items:Nearest Neighbor Search

Barna Saha

March 29, 2018

Finding Similar Items

- A fundamental data mining task

Finding Similar Items

- A fundamental data mining task
-
    - May want to find whether two documents are similar to detect plagiarism, mirror websites, multiple versions of the same article.
    - While recommending products we want to find users that have similar buying patterns.
    - In Netflix two movies can be deemed similar if they are rated highly by the same customers.

Jaccard Similarity

- A very popular measure of similarity for “sets”.

Jaccard Similarity

- A very popular measure of similarity for “sets”.
- The Jaccard similarity of sets S and T is $\frac{|S \cap T|}{|S \cup T|}$

Shingling of Documents

- k-Shingles: any substring of length k

Shingling of Documents

  - k-Shingles: any substring of length k

- Example

  Suppose a document D is $abcdabd$, then if $k = 2$, the 2-shingles are $\{ab, bc, cd, da, bd\}$

Shingling of Documents

  - k-Shingles: any substring of length k

- Example

  Suppose a document D is $abcdabd$, then if $k = 2$, the 2-shingles are $\{ab, bc, cd, da, bd\}$

  - Therefore from each document one can get a **set** of k-shingles and then apply Jaccard Similarity.

Shingling of Documents

- Choosing the shingle size.
  - If we use $k = 1$, most Web pages will have most of the common characters, so almost all Web pages will be similar.
  - k should be picked large enough such that the probability of any given shingle appearing in any given document is low.
  - For example, for research articles use $k = 9$.
- Hashing Shingles
  - Often shingles are hashed to a large hash table, and the bucket number is used instead of the actual k-shingle. From $\{ab, bc, cd, da, bd\}$, we may get $\{4, 5, 1, 6, 8\}$

Challenges of Finding Similar Items

- Number of shingles from a document could be large. If we have million documents, it may not be possible to store all the shingle-sets in main memory.
- Comparing pair-wise similarity among documents could be highly time-consuming.

Challenges of Finding Similar Items

- **Number of shingles from a document could be large. If we have million documents, it may not be possible to store all the shingle-sets in main memory.**
- Comparing pair-wise similarity among documents could be highly time-consuming.

Minhash

- When shingles do not fit in the main memory–create a small *signature* of each document from the set of shingles.

Minhash

- When shingles do not fit in the main memory–create a small *signature* of each document from the set of shingles.
- Consider a random permutation of all possible shingles (number of buckets in the hash table), pick the number from the set that appears first in that permutation.

Minhash

- Given two sets of shingles S and T, $Prob(S \text{ and } T \text{ have same minhash }) = Jaccard(S, T)$

Minhash

- Given two sets of shingles S and T,
  $Prob(S \text{ and } T \text{ have same minhash }) = Jaccard(S, T)$
- Take t such permutations to create a signature of length t.

Minhash

- Given two sets of shingles S and T, $Prob(S \text{ and } T \text{ have same minhash }) = Jaccard(S, T)$
- Take t such permutations to create a signature of length t.
- Compute the number of positions among t that are the same for the two documents. If that number is k, then the estimated $Jaccard(S, T)$ is $\frac{k}{t}$.

Minhash

- Given two sets of shingles S and T,
  $Prob(S$ and T have same minhash $) = Jaccard(S, T)$
- Take t such permutations to create a signature of length t.
- Compute the number of positions among t that are the same for the two documents. If that number is k, then the estimated $Jaccard(S, T)$ is $\frac{k}{t}$.
- When is this a good estimate? [Homework 2]

Challenges of Finding Similar Items

- Number of shingles from a document could be large. If we have million documents, it may not be possible to store all the shingle-sets in main memory.
- **Comparing pair-wise similarity among documents could be highly time-consuming.**

Challenges of Finding Similar Items

- Number of shingles from a document could be large. If we have million documents, it may not be possible to store all the shingle-sets in main memory.
- **Comparing pair-wise similarity among documents could be highly time-consuming.**
- If we have a million of documents, then for computing pair-wise similarity, we have to compute over half a trillion pairs of documents.

Locality Sensitive Hashing

- Often we want only the most similar pairs or all pairs that are above some threshold of similarity.

Locality Sensitive Hashing

- Often we want only the most similar pairs or all pairs that are above some threshold of similarity.
- We need to focus our attention only on pairs that are likely to be similar without investigating every pair.

Locality Sensitive Hashing (LSH)

- A hashing mechanism such that items with higher similarity have higher probability of colliding into the same bucket than others.

Locality Sensitive Hashing (LSH)

- A hashing mechanism such that items with higher similarity have higher probability of colliding into the same bucket than others.
- Use multiple such hash functions, and only compare items that are hashed in the same bucket.

Locality Sensitive Hashing (LSH)

- A hashing mechanism such that items with higher similarity have higher probability of colliding into the same bucket than others.
- Use multiple such hash functions, and only compare items that are hashed in the same bucket.
- **False positive**: When two “non-similar” items hash to the same bucket.

Locality Sensitive Hashing (LSH)

- A hashing mechanism such that items with higher similarity have higher probability of colliding into the same bucket than others.
- Use multiple such hash functions, and only compare items that are hashed in the same bucket.
- **False positive**: When two “non-similar” items hash to the same bucket.
- **False negative**: When two “similar” items do not hash to the same bucket under any of the chosen hash functions from the family.

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.
- If s is the Jaccard Similarity between two documents then

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.
- If s is the Jaccard Similarity between two documents then
  - Probability that the signature agrees completely in a particular band/bucket$=s^K$

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.
- If s is the Jaccard Similarity between two documents then
  - Probability that the signature agrees completely in a particular band/bucket$=s^K$
  - Probability that the signature does not agree in at least one position in a band/bucket$=1 - s^K$

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.
- If s is the Jaccard Similarity between two documents then
  - Probability that the signature agrees completely in a particular band/bucket$=s^K$
  - Probability that the signature does not agree in at least one position in a band/bucket$=1 - s^K$
  - Probability that the signature does not agree in at least one position in all of the L buckets is $(1 - s^K)^L$.

Locality Sensitive Hashing for MinHash Signatures

- Signature size n is divided into L buckets of size K each. $n = K * L$.
- Use L different hash functions (hence hash tables) each operating on a single band of size K.
- If s is the Jaccard Similarity between two documents then
  - Probability that the signature agrees completely in a particular band/bucket$=s^K$
  - Probability that the signature does not agree in at least one position in a band/bucket$=1 - s^K$
  - Probability that the signature does not agree in at least one position in all of the L buckets is $(1 - s^K)^L$.
  - Probability that there exists at least one hash function which will hash the two documents in the same bucket $1 - (1 - s^K)^L$.

Locality Sensitive Hashing for MinHash Signatures

- How do we select K and L given s?

Locality Sensitive Hashing for MinHash Signatures

- How do we select K and L given s?
- Suppose $s = (\frac{1000}{L})^{\frac{1}{K}}$, then the probability of becoming a candidate for comparison is $1 - (1 - \frac{1000}{L})^{L} \approx 1 - \frac{1}{e^{1000}}$

Applications:MinHash

- Source: Wikipedia
  A large scale evaluation has been conducted by Google in 2006 to compare the performance of Minhash and Simhash algorithms. In 2007 Google reported using Simhash for duplicate detection for web crawling and using Minhash and LSH for Google News personalization.
- Description from blogs:
  - `http://matthewcasperson.blogspot.com/2013/11/minhash-for-dummies.html`
  - `http://robertheaton.com/2014/05/02/jaccard-similarity-and-minhash-for-winners/`: matching twitter users
  - `http://blog.jakemdrew.com/2014/05/08/practical-applications-of-locality-sensitive-hashing-fo`
- Implementation: `https://github.com/rahularora/MinHash` –may have bugs.

Applications:LSH

- Near-duplicate detection
- Hierarchical clustering
- Genome-wide association study
- Image similarity identification
  - VisualRank
- Gene expression similarity identification[citation needed]
- Audio similarity identification
- Nearest neighbor search
- Audio fingerprint
- Digital video fingerprinting
- Anti-spam detection
- Security and digital forensic applications

Check out: `http://www.mit.edu/~andoni/LSH/` and `https://github.com/triplecheck/TLSH`