Mining Big Data

Finding Similar Items

(Chapter 3)

Motivation

- Finding similar items in a set of documents is a fundamental problem in data mining.

Example:

- Given a collection of web pages, the goal is to find near-duplicate pages.
- Such pages could be plagiarisms or mirrors

An important problem that arises is that there may be far too many pairs of items to test for similarity

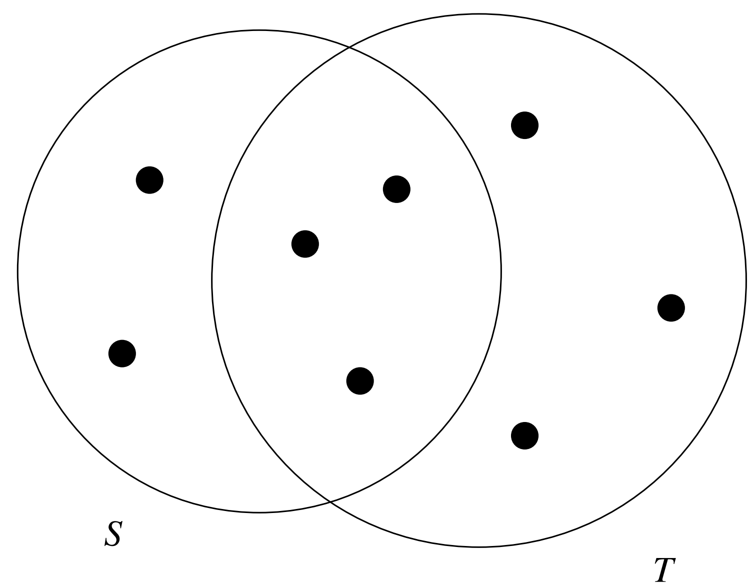
Applications of Near-Neighbor Search

- We need a notion of similarity.
- We measure the similarity of sets by the relative size of their intersection
- This is called “Jaccard similarity”.
- Applicable to
  - finding textually similar documents
  - collaborative filtering by finding similar customers and similar products

Jaccard Similarity of Sets

- Given two sets S and T, the Jaccard similarity is defined as $|S \cap T|/|S \cup T|$ (size of the intersection divided by size of the union)

Example with Jaccard similarity 3/8.
Figure 3.1 in Rajaramam/Ullman



Similarity of Documents

- Jaccard similarity works well for finding textually similar documents in a large set of documents (for example the Web or collection of news articles)
- We are currently looking at “character level” similarity not “similar meaning”
- Testing whether two documents are the same is easy (compare documents by character).

Examples

For many documents large portions of the text are identical.

Examples:

- Plagiarism
- Mirror Pages
- Articles from the same source

Collaborative Filtering

- Collaborative filtering is another application of similarity of sets.
- Goal is to recommend to users items that were liked by other users who have exhibited similar tastes.

Examples:

- Online Purchases
- Movie Ratings

Example: Online Purchases

- Amazon has millions of customers and sells millions of items
- Its database records which items have been bought by which customers.
- We can say that two customers are similar if their sets of purchased items have high Jaccard similarity
- Likewise, two items are similar if the sets of purchasers have high Jaccard similarity.
- We might expect mirror sites to have a Jaccard similarity of 90%
- For customers with similar tastes this will be much lower (20% might be unusual).
- Collaborative filtering is often combined with clustering (see Chapter 7) in this case.

Movie Ratings

- NetFlix records which movies each of its customers rented and ratings of movies by customers.
- We can regard movies as similar if rented or rated highly by many of the same customers.
- We can regard customers as similar if they rented or rated highly many of the same movies.
- Similarities don't need to be high to be significant (same as for Amazon)
- Clustering movies by genre will make things easier.

Movie Ratings

How to deal with ratings.

- Ignore low-rated customer/movie pairs and treat them as if the customer never rented the movie.
- Two elements for each movie: liked or hated. Use Jaccard similarities for these different categories.
- If ratings are 1-to-5 stars: Put a movie in a customers set i times if they rated it i stars. Use Jaccard similarity for bags B and C and count an element j times in the intersection if j is the minimum of the number of times the element appears in B and C.

Shingling of Documents

- Represent documents as sets by constructing the set of short strings that appear in a document.
- Documents that share pieces (sentences, phrases) will have many common elements in their sets.

K-Shingles

- A document is a string of characters.
- A k-shingle for a document is any substring of length k in that document
- We can associate with each document its set of k-shingles (appear at least once in the document).

Example:

- Document D is the string abcdabd
- Set of 2-shingles for D is {ab,bc,cd,da,bd}
- Note that ab appear twice in D (but not in 2-shingles)

White Spaces

Several ways to treat white spaces (blank, tab, newline, etc).

Makes sense to replace any sequence of white spaces by a single blank.

Example:

- If we use k=9 and eliminate all white spaces then "The plane was ready for touch down" and
- "The quarterback scored at touchdown" look similar.
- Doesn't hold if we keep a blank.

Shingle Size

- We can pick k as any constant we like.
- If k is too small, sequences of k characters appear in most documents.
- We could have documents whose shingle-sets have high Jaccard similarity although the documents don’t share phrases or sentences.
- Good choice of k depends on how long typical documents are and how large the set of typical characters is.
- k should be large enough such that the probability of any given shingle in any given document is low.

Shingle Size

- If corpus of documents is emails, k=5 should be fine.
- Suppose that only letters and general white-spaces appear.
- This implies 27^5=14,348,907 possible shingles.
- Typical email is much smaller than 14 million characters and we expect k=5 to work well (and it does).
- Calculation is more subtle as characters don't appear with equal probability (for example letter "z" doesn't appear to often)
- Good rule of thumb is to imagine that there are only 20 characters and estimate number of k-shingles as 20^k.
- For large documents, such as research articles, k=9 is considered safe.

Hashing Shingles

- Instead of using substrings directly, we can pick a hash function and map a string of length k to some number of buckets
- Treat the resulting bucket as shingle.
- Set representing a document is the set of integers that are bucket numbers of one or more k-shingles that appear in the document.

Example:

- Set of 9-shingles for a document
- Map each 9-shingle to a bucket number in the range 0 to 2^{32}-1.
- Each shingle is represented by 4 bytes (instead of nine).

Data is compacted and we can manipulate (hashed) shingles by single-word machine operations.

Shingle from Words

- We want to identify similar news articles.
- News articles are most prose and have a lot of stop words such as “and”, “you”, “to”.

- Often we ignore stop words since they are not useful.
- For finding similar news articles, it has been shown that using shingles defined by a stop word plus the next two words is useful.

Example

- An ad might have the simple text “Buy Sudzo”
- A news article with the same idea might read as “A spokesperson for the Sudzo Corporation revealed today that studies have shown it is good for people to buy Sudzo products” (stop words in red)
- First three shingles made from stop word + 2 following words are:
  - A spokesperson for
  - For the Sudzo
  - the Sudzo Corporation
- In total nine such shingles from the sentence, but none from the ad.

Similarity-Preserving Summaries

- Sets of shingles are large (even holds if we hash them, 4 times the size of the document)
- If we have millions of documents, it might not be possible to store all the shingle-sets in main memory.
- We want to replace large sets by much smaller representations called "signatures".
- Import property: We can compare the signatures of two sets and estimate the Jaccard similarity of the underlying sets.
- Signatures can't give exact similarity.
- But they provide good estimates and get more accurate for larger signatures.

Matrix Representation of Sets

- Goal is to get good signatures, let’s start with representations for sets.
- We now visualize a collection of sets by their characteristic matrix.
- Columns correspond to the sets, rows correspond to the elements.
- (r,c)=1 if element for row r is member of set for column c.
- Otherwise the value in position (r,c)=0.
- If rows are products and columns are customers, the matrix represents the products bought by the different customers.

Example

Matrix representing four sets.

$Element$	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Figure 3.2 in Rajaramam/Ullman

Minhashing

- Signatures that we want to construct for sets are composed of results of a large number of calculations (say several hundreds).
- Each calculation is a “minhash” of the characteristic matrix.
- We want to see how minhash is computed in principle.
- To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.
- The minhash value of any column is the number of the first row (in permuted order) in which the column has a 1.

Example

- Suppose we pick order of rows beadc.
- This permutation defines a minhash function h.
- We have
  - $h(S_1)=a$
  - $h(S_2)=c$
  - $h(S_3)=b$
  - $h(S_4)=a$

$Element$	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Figure 3.3 in Rajaramam/Ullman

MinHashing and Jaccard Similarity

Connection between minhashing and Jaccard similarity:

- The probability that the minhash function for a random permutation of the rows produces the same value for two sets equals the Jaccard similarity of those sets.

MinHashing and Jaccard Similarity

Reason: Consider the columns for those to sets, let’s say S1 and S2.

- Rows can be divided into three classes:
  - Type X rows have 1 in both columns
  - Type Y rows have 1 in one of the columns and 0 in the other
  - Type Z rows have 0 in both columns
- Consider similarity SIM(S1,S2) and probability that h(S1)=h(s2)
- Let there be x rows of typ X and y rows of type Y.
- We have SIM(S1,S2)= x/(x+y).
- Consider rows that are permuted randomly.
- We have h(S1)=h(S2) if we meet a X row before a Y row.
- When moving from top to botton the probability of meeting a type X row before a type Y row is x/(x+y).

Minhash Signatures

- Think of collection of sets by characteristic matrix M.
- To represent sets, we pick at random some number n of permutations of rows of M.
- 100 permutations or several hundred permutations will do.
- We call the minhash functions by these permutations
  h1, h2, …, hn
- From the column representing set S, construct the minhash signature for S given by the vector [h1(S), h2(S), …., hn(s)].
- We can form from M a signature matrix in which the ith column is replaced by the minhash signature.
- Resulting matrix has same number of columns as M, but only n rows (much smaller than M).

Computing Minhash Signatures

- It's not feasible to permute a large characteristic matrix explicitly.
- It's possible to simulate the effect of a random permutation by a random hash function that maps row numbers to as many buckets as there are rows.
- Hash function that maps integers 0, 1, …, k-1 to bucket number 0,1,…,k-1 will typically lead to collisions and leave other buckets unfilled.
- However, difference is unimportant as long as k is large and there are not too many collisions.

Computing Minhash Signatures

- Instead of picking n random permutations of rows, we pick n randomly chosen hash functions h1, h2, …, hn on the rows.
- We construct signature matrix by considering each row r in given order

1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\textsc{sig}(i, c)$ to the smaller of the current value of $\textsc{sig}(i, c)$ and $h_i(r)$.

Figure 3.4 in Rajaramam/Ullman

Example (3.8 in Rajaramam/Ullman)

- Consider the hash functions h1(x) = x+1 mod 5 and h2(x) = 3x + 1 mod 5.
- Letters naming the rows are replaced by numbers 0 through 4 and hash are applied to them.

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Example

Initial state:Matrix consist of all ∞'s.

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Row 0: 1s in S1 and S4 and h1(0)=h2(0)=1.

This leads to

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Example

Row 1: 1 in S3 and h1(1)=2, h2(1)=4.

- We set SIG(1,3)=2, SIG(2,3)=4

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Row2: 1s in S2 and S4, h1(2)=3, h2(2)=2.

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

Example

Row 3: 1s in S1, S3, S4. h1(3)=4, h2(3)=0.

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Row 4: 1 in S3, h1(4)=0, h2(4)=3.

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Example

- We can estimate the Jaccard similarities of the underlying sets from the signature matrix.

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

We can estimate the Jaccard similarity by the fraction of rows where given two sets agree.

- SIM(S1,S4)=1.0, but the true Jaccard similarity of S1 and S4 is 2/3.
- SIM(S1,S3)=1/2 (true ¼)
- SIM(S1,S2)=0 (correct)

Locality-Sensitive Hashing

- Minhashing compresses large documents into small signatures and preserves expected similarity of any pair of documents.
- It still might be impossible to find the pairs with greatest similarity efficiently.

Reason:

- Number of pairs of documents may be too large even if there are not too many documents.

Example

- Assume we have a million documents and use signatures of length 250.
- We use 1000 bytes per document for signatures and entire data fits in a gigabyte (and therefore in main memory)
- However there are roughly $1{,}000{,}000^2/2$ (half a trillion) pairs of documents.
- If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all similarities (on a laptop)

Local-sensitivity Hashing

- If we want to compute the similarity of every pairs, there is nothing we can do (except parallelizing the computation of similarities)
- Often we want the most similar pairs or all pairs above some lower bound in similarity.
- Local-sensitivity hashing (LSH) allows to do this without investigating every pair.

Procedure for Finding Similar Documents:

- Pick value of k and construct from each document the set of k-shingles.
- Sort document-shingle pairs by shingle
- Pick length n of minhash signatures and compute minhash signatures for all documents.
- Choose a threshold t that defines similarity.
- Pick number of bands b and number of rows r such that br=n and threshold for S-curve is lower than t (limits false negatives).
- Construct candidate pairs by LSH
- Examine each candidate pair and determine whether similarity is at least t.
- Optional: If signatures are sufficient similar, check the documents directly.

LSH for Minhash Signatures

- One approach to LSH is to hash items several times.
- Similar items are more likely to be hashed to the same bucket than dissimilar ones.
- We consider any pair that hashed to the same bucket for any of the hashings as a candidate pair.
- Check only candidate pairs for similarity
- Dissimilar pairs that are hashed to the same bucket are false-positive (we hope that there are not too many)
- Truly similar pairs are missed (false-negative) if they are not hashed to the same bucket for a least one of the hash function (hope that there are just a few)

- If we have minhash signatures for the items, we can divide the signature matrix into b bands consisting of r rows each.
- For each band, there is a hash function that takes the vectors of r integers and hashes them to some large number of buckets.
- We can use the same hash function for each band, but have to use separate buckets for each band.

Bands

- Dividing signature matrix into bands

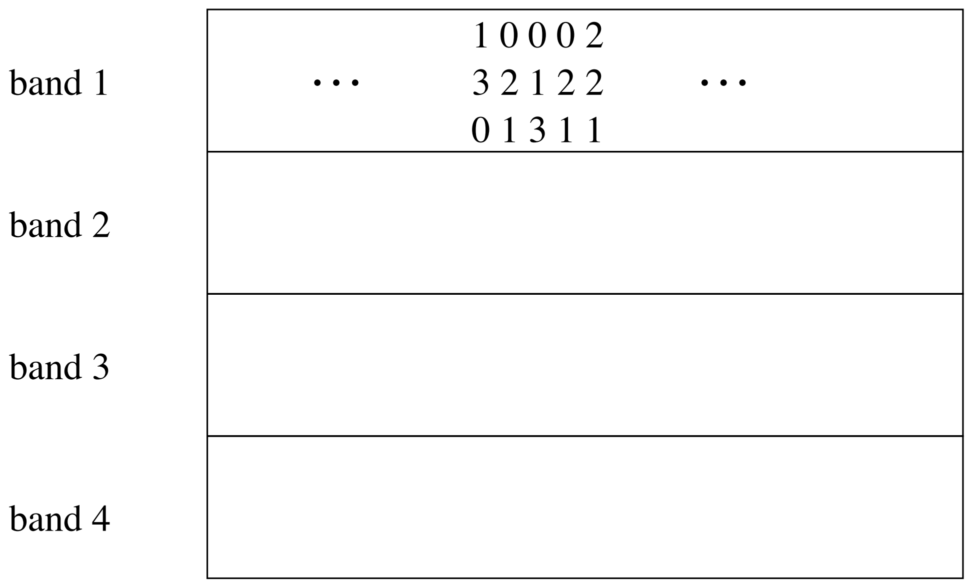


Figure 3.6 in Rajaramam/Ullman

Analysis of Banding

- Suppose that we use b bands of r rows each and that a particular pair of documents has Jaccard similarity s.
- The probability that the minhash signatures for these documents agree in any particular row is s.

Probability that these documents become candidate pair:

- Probability that the signatures agree in all rows of one particular bands is s^r.
- Probability that they do not agree in at least one row is $1-s^r$.
- Probability that they do not agree in all rows of any of the bands is $(1-s^r)^b$.
- Probability that the signatures agree in all rows of at least one band (and mapped to same bucket) is $1-(1-s^r)^b$.

S-Curve

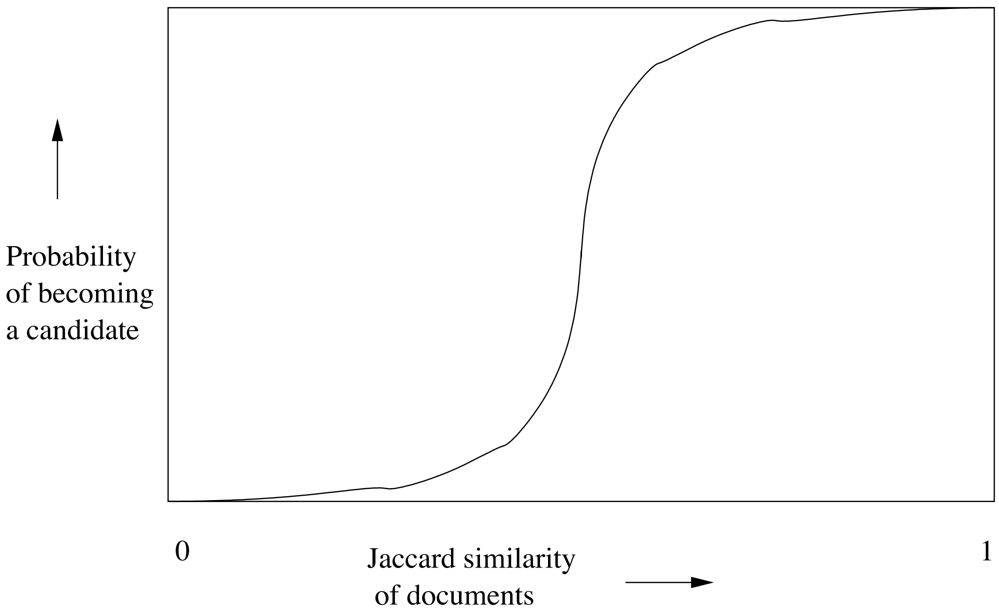


Figure 3.7 in Rajaramam/Ullman

- Regardless of chosen constants b and r, the probability function has the form of an S-curve. (steepest rise is determined by b and r)
- $(1/b)^{(1/r)}$ is approximation of threshold.
- Example b=16, r=4, threshold is approximately ½.

Example

- Consider b=20, r=5 (signatures of length 100)
- Functions values for $1-(1-s^r)^b = 1-(1-s^5)^{20}$

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

Figure 3.8 in Rajaramam/Ullman

Combining Techniques

- We can combine the examined techniques to find similar documents.
- The approach can produce false negatives (pairs of similar documents that are not identified).
- It also contains false positives (pairs that are evaluated but not found to be sufficiently similar)

Procedure for Finding Similar Documents:

- Pick value of k and construct from each document the set of k-shingles.
- Sort document-shingle pairs by shingle
- Pick length n of minhash signatures and compute minhash signatures for all documents.
- Choose a threshold t that defines similarity.
- Pick number of bands b and number of rows r such that br=n and threshold for S-curve is lower than t (limits false negatives).
- Construct candidate pairs by LSH
- Examine each candidate pair and determine whether similarity is at least t.
- Optional: If signatures are sufficient similar, check the documents directly.