Min-Hash and Document comparison

Yoav Freund

This lecture is based on:

- "Mining Massive Datasets" by Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman
- Description is in section 11.6 in the lecture notes.

Finding Similar Items

1. Based on chapter 3 of the book “Mining Massive Datasets” by Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman
2. Suppose we recieve a stream of documents.
3. We want to find sets of documents that are very similar
4. Reasons: Plagiarism, Mirror web sites, articles with a common source.

Measuring the distance between sets

1. Suppose we consider the **set** of words in a document. Ignoring order and number of occurances.
2. We will soon extend this assumption.
3. If two documents define two sets S, T, how do we measure the similarity between the two sets?
4. Jaccard similarity: $\frac{|S \cap T|}{|S \cup T|}$

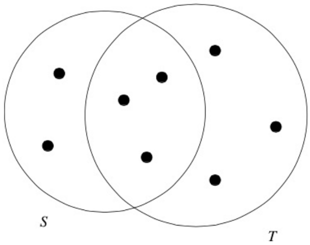


Figure 3.1: Two sets with Jaccard similarity 3/8

Hash Functions

1. Let X be a finite (but large) set
2. Let $N = \{1, 2, \ldots, n\}$ be a (very large) set of numbers.
3. A Hash-Function $h : X -> N$ is a function that "can be seen as" a mapping from each element of X to a an indpendently and uniformly chosen random element of N.
4. In actuality, the functions are pseudo-random and not random. But that distinction is out of scope for this class.

Min-Hash

1. Choose a random hash function h_i
2. Given a set of elements S in the domain X
3. $\text{min-}H_i(S) = \min_{s \in S} h_i(s)$
4. A min-hash **signature** for a document is the vector of numbers $\langle \text{min-}H_1(S), \text{min-}H_2(S), \ldots, \text{min-}H_k(S) \rangle$
5. Signature also called a "sketch": Any length document is represented by k numbers.
6. A lot of information is lost, but enough is retained to approximate the Jaccard similarity.

Visualizing Min-Hash

- We can represent the set of words in each document as a matrix.
- Rows $a, b, c, \ldots$ correspond to words.
- Columns $S_1, S_2, \ldots$ correspond to documents.
- A "1" in row b, column S_i means that document S_i contains the word b
- Hashing corresponds to randomly permuting the rows.
- Min-hashing a document corresponds to identifying the first "1" starting from the top of the column

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

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

Understanding Min-Hash

- For any set S of size $|S|$, the probability that any particular element $s \in S$ is the min-hash is $1/|S|$
- Fix two documents S_i, S_j (columns) and partition the rows that contain at least a single "1" in those columns
- Denote by X rows that contain 1,1 (both documents contain the word.)
- Denote by Y rows that contain 1,0 or 0,1 (only one document contains the word)
- Permuting the rows does not change which rows are X and which are Y
- The min-hash of S_i, S_j agree if and only if first row that is not 0,0 is an X
  - The probability that the min-hash of S_i, S_j agree is exactly $\frac{\#X}{\#X+\#Y}$ which is equal to $JS(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$

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

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

Estimating Jaccard Similarity

1. We can use min-hash to estimate Jaccard similarity (JS): $\frac{|S_i \cap S_j|}{|S_i \cup S_j|}$
2. For each min hash function MH_i we have that

$$P_i\left[\text{min-}H_i(S) = \text{min-}H_i(T)\right] = \frac{|S \cap T|}{|S \cup T|}$$

3. A single comparison yields only true (1) or false (0)
4. Taking the average of k independent hash functions we can get an accurate estimate.

How many hash functions do we need? (1)

1. From a statistics point of view we have k independent binary random variables:

$$X_i = \begin{cases} 1 & \text{if min-}H_i(S) = \text{min-}H_i(T) \\ 0 & \text{otherwise} \end{cases}$$

2. We seek the expected value: $p \doteq E(X_i) = \frac{|S \cap T|}{|S \cup T|}$
3. We have to overcome the large std: $\sigma(X_i) = \sqrt{p(1-p)}$
4. Averaging gives a random variable with the same expected value but a smaller variance.

$$Y = \frac{1}{k} \sum_{i=1}^{k} X_i; \;\; E(Y) = p \;\; \sigma(Y) = \sqrt{\frac{p(1-p)}{k}}$$

5. $$\sigma(Y) \leq \sqrt{1/2(1-1/2)(1/k)} = \frac{1}{2\sqrt{k}}$$

Using a z-Scores to calculate the minimal number of hash functions.

1. Suppose we want our estimate of JS to be within ± 0.05 of the Jaccard distance with probability at least 95%
2. The fraction of min-has matches is the average of k independent binary random variables.
3. Lets assume k is large enough so that central limit theorem holds.
4. We want a confidence of 95% that the estimate is within ± 0.05 of the true value. In other words, we want

$$2\sigma(Y) \leq 0.05$$

5. Using the bound

$$\sigma(Y) \leq \frac{1}{2\sqrt{k}}$$

we find that it is enough if $\frac{1}{k} \leq 0.05$ or if $k \geq 20$

Introducing Order

1. So far, we represented each document by the set of words it contains
2. This removes the order in which the words appear: "Sampras beat Nadal" is the same as "Nadal beat Sampras"
3. We can add order information to the set representation using **Shingles**

Shingles

1. Consider the sentence:"the little dog loughed to see such craft"
2. Word set representation: {"the","little","dog", "loughed","to","see","such","craft"}
3. 2-shingle representation: {"the little","little dog", "dog loughed","loughed to","to see","see such","such craft"}
4. 3-shingle representation: {"the little dog","little dog loughed",...}
5. And so on
6. The number of shingles of length k from a document of length n is?
7. $n+1-k$ - largest for single words!
8. On the other hand, there is a much larger number of **different items.**
9. k too small - documents judged similar too often.
10. k too large - documents judged dissimilar too often

The problem with computing all pairwise distances

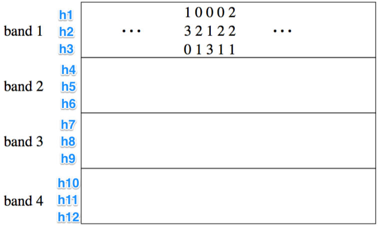
1. Suppose we have $1,000,000$ documents, each containing about $10,000$ words
2. Loading all documents to memory is infeasible
3. Using a min-hash signature for each document reduces storage to about 50 numbers per document. We can load into memory
4. Still, there are about $C(1,000,000,2) \approx 10^{12} \textit{pairs}$
5. Computing *all* pairwise distances is infeasible
6. How about finding just the most similar pairs?

Finding only the similar pairs

1. Idea 1: If the distance between similar pairs is much smaller than the average distance between random pairs then we can eliminate most pairs even with a crude approximation of the distance.
2. Idea 2: Make a single pass over elements and map that to a **hash table** in such a way that distant pairs tend to fall in different bins.
3. Pairs that fall in the same bin are "candidates" - only candidates are compared to see whether they are actually similar pairs.
4. The process is repeated several times to increase the chance that all similar pairs become candidates.

Algorithm for efficiently finding the most similar pairs

1. Suppose we are using m functions for performing min-hash
2. Partition the functions to b bands, each containing r rows. $m = br$. In the figure $m = 12, b = 4, r = 3$



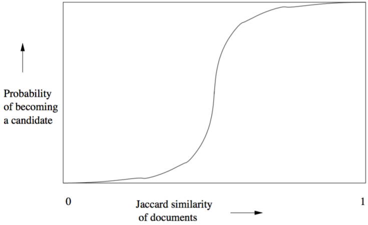
1. Use hash function to map the r-vector in each band to a hash table with K bins.
2. Cadidate pairs are document that fall in the same bin in at least one of the bands.
3. Compute the similarity of the candidate pairs exactly
4. Accept those pairs whose similarity is higher than some threshold θ

The effect of b and r (1)

1. Fix two documents (=two columns in the table). Suppose that the Jaccard similarity between them is s.
2. The probability that the r-vectors are identical, is s^r, in this case the two r-vectors will map to the same bin in the hash table.
3. The probability that the the two r-vectors differ in at least on place is $1 - s^r$
4. The probability that all b signatures differ between the two documents is $(1 - s^r)^b$

The effect of b and r (2)

1. The probability that all b signatures differ between the two documents is $(1 - s^r)^b$
2. For $b = 20$, $r = 5$ we get the table on the right.



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

- For high similarity $s \geq 0.8$ the document pair is very likely to be a candidate.
- For low similarity $s \leq 0.2$ the document pair is very likely not to be a candidate.
- If 99.9% of the documents have low similarity then the number of false positives is about $10^{12} \times (.006 + 0.001) = 7 \times 10^9$. Still large, but not prohibitively so.


happy
thanks
giving