Data Mining

SIMILARITY AND DISTANCE MEASURES

Prof. Dr. Hikmat Ullah Khan
Department of Information Technology
UNIVERSITY OF SARGODHA, SARGODHA

Lesson from Holy Quran

Topics

- Distance
- Similarity
- Jaccard Coefficient
- Dice coefficient
- Cosine Similarity
  - TF
  - DF
  - IDF
- Applications
- Algorithms
- Task

Distance Measures

- Common Distance Metrics:
  - Euclidean distance(continuos distribution)

$$d(p,q) = \sqrt{\sum (p_i - q_i)^2}$$

  - Manhatton Distance

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

  - Hamming distance (overlap metric)

bat (distance = 1)	**toned (distance = 3)**
cat	**roses**

  - Discrete Metric(boolean metric)

if $x = y$ then $d(x,y) = 0$. Otherwise, $d(x,y) = 1$

Detailed Example for Distances

Name	Deptt	Age	CGPA
Umar	CS	23	3.1
Umair	CS	21	2.7

Detailed Example for Distances

Name	Deptt	Age	CGPA
Umar	CS	23	3.1
Umair	CS	21	2.7

1. Hamming Distance (Umar and Umair) = 1
2. Discrete Distance (CS and CS) = 0
3. Euclidean Distance (23 and 21) = sqrt($((23-21)_2)$) = 2
4. Manhattan Distance (3.1 and 2.7) = 0.4

Similarity

- Numerical measure of how alike two data objects are.
  - A function that maps pairs of objects to real values
  - Higher when objects are more alike.
- Often falls in the range [0,1]
- Properties for similarity
  1. s(p, q) = 1 (or max similarity) only if p = q. (Identity)
  2. s(p, q) = s(q, p) for all p and q. (Symmetry)

Similarity between sets

- Consider the following documents

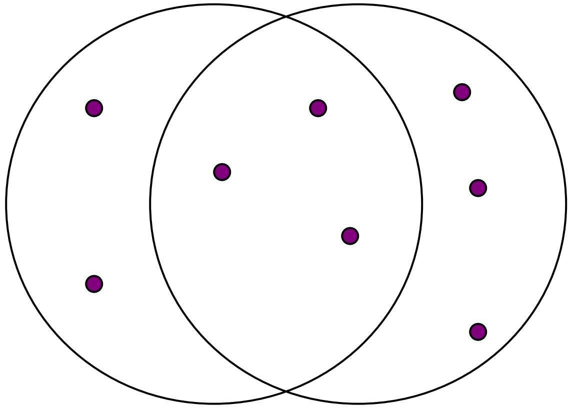
apple releases new ipod	apple releases new ipad	new apple pie recipe

- Which ones are more similar?

- How would you quantify their similarity?

Jaccard Similarity

- The Jaccard similarity (Jaccard coefficient) of two sets S_1, S_2 is the size of their intersection divided by the size of their union.
  - JSim $(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$.



3 in intersection.
8 in union.
Jaccard similarity
= 3/8

  - Extreme behavior:
    - JSim(X,Y) = 1, iff X = Y
    - JSim(X,Y) = 0 iff X,Y have no elements in common

Jaccard Coefficient –(another way too)

- Comparing the similarity and diversity of sample sets
- Jaccard Co-efficient is calculated as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Dice Coefficient

- Also known as **Sørensen–Dice index**,
- Used for comparing the similarity and diversity of sample sets
- Dice Co-efficient is calculated as follows:

$$= \frac{2|X \cap Y|}{|X| + |Y|}$$

Dice and Jaccard Coefficient

- Take any two Sets and then compute the
  - Jaccard Similarity
  - Dice Similarity

Vector Space Model(Cosine Similarity)

- Model for representing text documents
- It is used in [Applications]
  - information retrieval.
  - relevancy rankings.
  - Plagiarism detection
  - Topic based search
  - Expert/Advisor Search
- Model for searching query-based results
- Documents and queries are represented as vectors.

Advantage

- Simple model based on linear algebra
- Term weights not binary
  - Frequency based
- Provides similarity between queries and documents
- Allows partial matching

Note

1. do not take into account WHERE the terms occur in documents.
2. use all terms, including very common terms and **stop-words**.
3. No need to reduce terms to root terms (**stemming**).

Example

- D1: "Shipment of gold damaged in a fire"
  D2: "Delivery of silver arrived in a silver truck"
  D3: "Shipment of gold arrived in a truck”

- query : "gold silver truck“

Terms

- Term Frequency (tf)
  - No of times a term occurred in a document
- Document Frequency (df)
  - No of documents in which a term occurred.
- Inverse Document Frequency
  - IDF = $\log(D/d_i)$

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$

Query, Q: "gold silver truck"

D_1: "Shipment of gold damaged in a fire"

D_2: "Delivery of silver arrived in a silver truck"

D_3: "Shipment of gold arrived in a truck"

D = 3; IDF = $\log(D/df_i)$

	Counts, tf_i							Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

$$|D_1| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.5173} = 0.7192$$

$$|D_2| = \sqrt{0.1761^2 + 0.4771^2 + 0.9542^2 + 0.1761^2} = \sqrt{1.2001} = 1.0955$$

$$|D_3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.1240} = 0.3522$$

$$\therefore \ |D_i| = \sqrt{\sum_i w_{i,j}^2}$$

$$|Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.1761^2} = \sqrt{0.2896} = 0.5382$$

$$\therefore \ |Q| = \sqrt{\sum_i w_{Q,j}^2}$$

$$Q \bullet D_1 = 0.1761 * 0.1761 = 0.0310$$

$$Q \bullet D_2 = 0.4771 * 0.9542 + 0.1761 * 0.1761 = 0.4862$$

$$Q \bullet D_3 = 0.1761 * 0.1761 + 0.1761 * 0.1761 = 0.0620$$

$$\therefore \; Q \bullet D_i = \sum_i w_{Q,j} w_{i,j}$$

$$\text{Cosine}\ \theta_{D_1} = \frac{Q \bullet D_1}{|Q| * |D_1|} = \frac{0.0310}{0.5382 * 0.7192} = 0.0801$$

$$\text{Cosine}\ \theta_{D_2} = \frac{Q \bullet D_2}{|Q| * |D_2|} = \frac{0.4862}{0.5382 * 1.0955} = 0.8246$$

$$\text{Cosine}\ \theta_{D_3} = \frac{Q \bullet D_3}{|Q| * |D_3|} = \frac{0.0620}{0.5382 * 0.3522} = 0.3271$$

$$\therefore\ \text{Cosine}\ \theta_{D_i} = \text{Sim}(Q, D_i)$$

$$\therefore\ \text{Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2}\ \sqrt{\sum_i w_{i,j}^2}}$$

Ranking

- Rank 1: Doc 2 = 0.8246
  Rank 2: Doc 3 = 0.3271
  Rank 3: Doc 1 = 0.0801

Algo (Optional)

CosineScore(q)

float $Scores[N] = 0$
Initialize $Length[N]$
for each query term t
do calculate $w_{t,q}$ and fetch postings list for t
    for each pair$(d, \mathrm{tf}_{t,d})$ in postings list
    do $Scores[d] \mathrel{+}= \mathrm{wf}_{t,d} \times w_{t,q}$
Read the array $Length[d]$
for each d
do $Scores[d] = Scores[d] / Length[d]$
return Top K components of $Scores[]$

Figure 6.14: The basic algorithm for computing vector space scores.

Task

- Think to create your own Document and Query

- Take one example and Solve it

  - solve taking an example
  - Use built in any language or Implement yourself
  - C#
  - Python
  - R
  - Or any other language

Task


Every successful person has a painful story.
Every painful story has a successful ending.
Accept the pain and get ready for success.
MumbaiHangout.Org