# DATA MINING
# LECTURE 5

Similarity and Distance

Recommendation Systems

Sketching, Locality Sensitive Hashing

# SIMILARITY AND DISTANCE

Thanks to:

Tan, Steinbach, and Kumar, “Introduction to Data Mining”

Rajaraman and Ullman, “Mining Massive Datasets”

# Similarity and Distance

- For many different problems we need to quantify how close two objects are.
- Examples:
  - For an item bought by a customer, find other similar items
  - Group together the customers of a site so that similar customers are shown the same ad.
  - Group together web documents so that you can separate the ones that talk about politics and the ones that talk about sports.
  - Find all the near-duplicate mirrored web documents.
  - Find credit card transactions that are very different from previous transactions.
- To solve these problems we need a definition of similarity, or distance.
  - The definition depends on the type of data that we have

# 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], sometimes in [-1,1]

- Desirable properties for similarity
  1. $s(p, q) = 1$ (or maximum 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?

# Similarity: Intersection

- Number of words in common

| | | |
|---|---|---|
| apple releases new ipod | apple releases new ipad | new apple pie recipe |

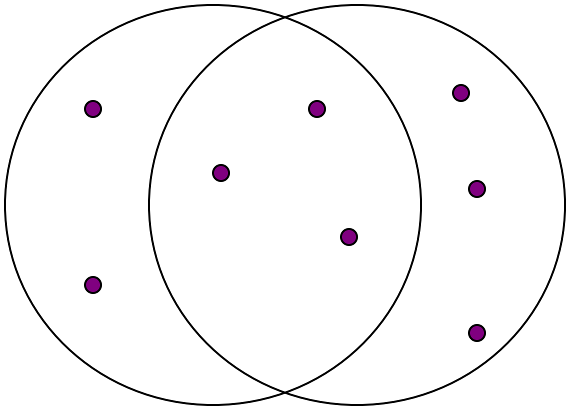
- Sim(D,D) = 3, Sim(D,D) = Sim(D,D) =2
- What about this document?

Vefa rereases new book with apple pie recipes

- Sim(D,D) = Sim(D,D) = 3

# 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
- JSim is symmetric

# Jaccard Similarity between sets

- The distance for the documents

apple releases new ipod

apple releases new ipad

new apple pie recipe

Vefa rereases new book with apple pie recipes

- JSim(D,D) = 3/5
- JSim(D,D) = JSim(D,D) = 2/6
- JSim(D,D) = JSim(D,D) = 3/9

# Similarity between vectors

Documents (and sets in general) can also be represented as vectors

| document | Apple | Microsoft | Obama | Election |
|---|---|---|---|---|
| D1 | 10 | 20 | 0 | 0 |
| D2 | 30 | 60 | 0 | 0 |
| D3 | 60 | 30 | 0 | 0 |
| D4 | 0 | 0 | 10 | 20 |

How do we measure the similarity of two vectors?

- We could view them as sets of words. Jaccard Similarity will show that D4 is different form the rest
- But all pairs of the other three documents are equally similar

We want to capture how well the two vectors are aligned

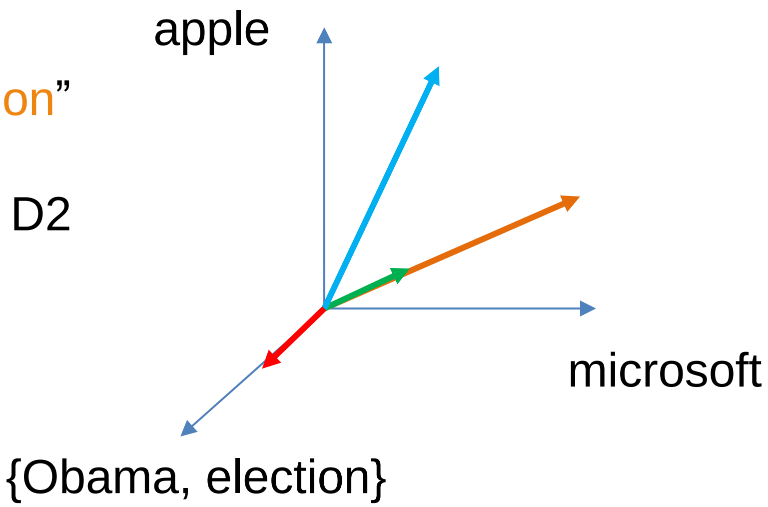
# Example

| document | Apple | Microsoft | Obama | Election |
|---|---|---|---|---|
| D1 | 10 | 20 | 0 | 0 |
| D2 | 30 | 60 | 0 | 0 |
| D3 | 60 | 30 | 0 | 0 |
| D4 | 0 | 0 | 10 | 20 |

Documents D1, D2 are in the “same direction”

Document D3 is on the same plane as D1, D2

Document D3 is orthogonal to the rest



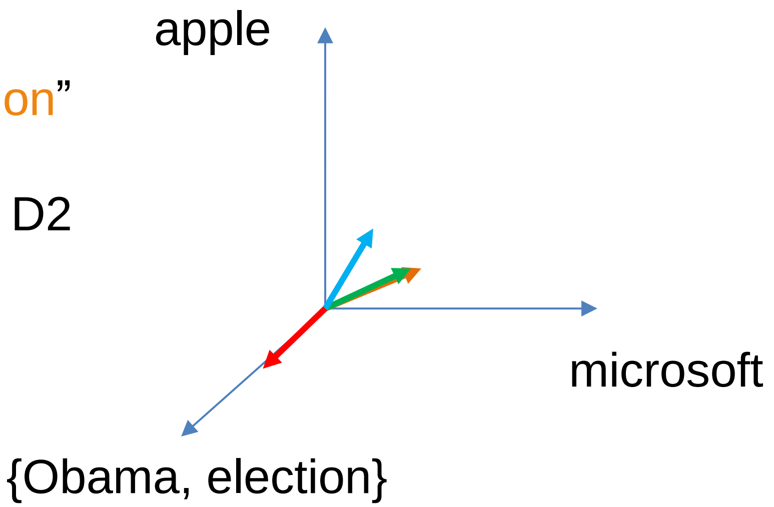
# Example

| document | Apple | Microsoft | Obama | Election |
|---|---|---|---|---|
| D1 | 1/3 | 2/3 | 0 | 0 |
| D2 | 1/3 | 2/3 | 0 | 0 |
| D3 | 2/3 | 1/3 | 0 | 0 |
| D4 | 0 | 0 | 1/3 | 2/3 |

Documents D1, D2 are in the "same direction"

Document D3 is on the same plane as D1, D2

Document D3 is orthogonal to the rest



# Cosine Similarity

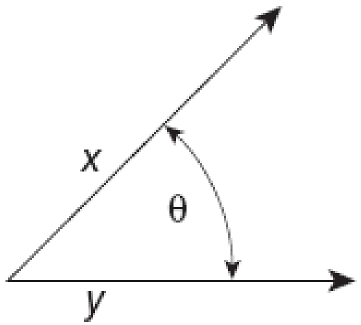



**Figure 2.16.** Geometric illustration of the cosine measure.

- Sim(X,Y) = cos(X,Y)
  - The cosine of the angle between X and Y

- If the vectors are aligned (correlated) angle is zero degrees and cos(X,Y)=1
- If the vectors are orthogonal (no common coordinates) angle is 90 degrees and cos(X,Y) = 0

- Cosine is commonly used for comparing documents, where we assume that the vectors are normalized by the document length.

# Cosine Similarity - math

- If $d_1$ and $d_2$ are two vectors, then

  $\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1||\ ||d_2||$ ,

  where $\bullet$ indicates vector dot product and $||\ d\ ||$ is the length of vector $d$.

- Example:

  $d_1$ **= 3 2 0 5 0 0 0 2 0 0**
  $d_2$ **= 1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

$||d_1|| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

$||d_2|| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.245$

$\cos(d_1, d_2) = .3150$

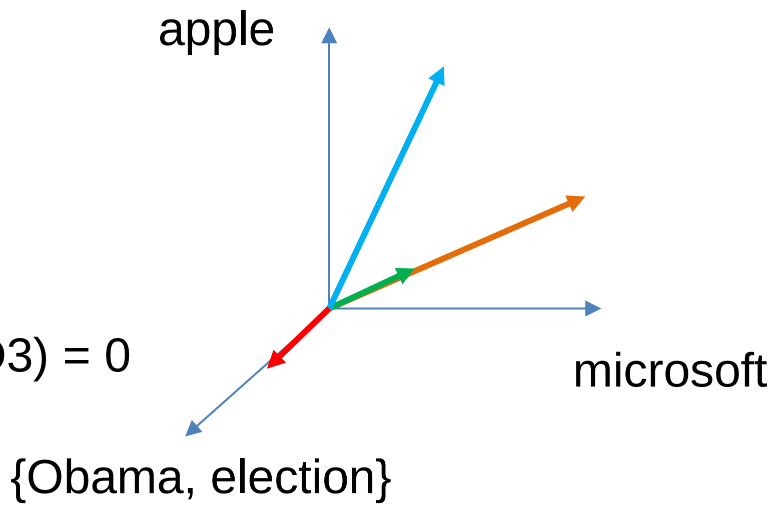
# Example

| document | Apple | Microsoft | Obama | Election |
|---|---|---|---|---|
| D1 | 10 | 20 | 0 | 0 |
| D2 | 30 | 60 | 0 | 0 |
| D3 | 60 | 30 | 0 | 0 |
| D4 | 0 | 0 | 10 | 20 |

Cos(D1,D2) = 1

Cos (D3,D1) = Cos(D3,D2) = 4/5

Cos(D4,D1) = Cos(D4,D2) = Cos(D4,D3) = 0



# Distance

- Numerical measure of how different two data objects are
  - A function that maps pairs of objects to real values
  - Lower when objects are more alike
  - Higher when two objects are different
- Minimum distance is 0, when comparing an object with itself.
- Upper limit varies

# Distance Metric

- A distance function d is a distance metric if it is a function from pairs of objects to real numbers such that:
    1. $d(x,y) \geq 0$. (non-negativity)
    2. $d(x,y) = 0$ iff $x = y$. (identity)
    3. $d(x,y) = d(y,x)$. (symmetry)
    4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality ).

# Triangle Inequality

- Triangle inequality guarantees that the distance function is well-behaved.
  - The direct connection is the shortest distance

- It is useful also for proving useful properties about the data.

# Distances for real vectors

- Vectors $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$
- $\mathbf{L_p}$ norms or Minkowski distance:

$$L_p(x, y) = \left[|x_1 - y_1|^p + \cdots + |x_d - y_d|^p\right]^{1/p}$$

- $\mathbf{L_2}$ norm: Euclidean distance:

$$L_2(x, y) = \sqrt{|x_1 - y_1|^2 + \cdots + |x_d - y_d|^2}$$

- $\mathbf{L_1}$ norm: Manhattan distance:

$$L_1(x, y) = |x_1 - y_1| + \cdots + |x_d - y_d|$$

- $\mathbf{L_\infty}$ norm:

$$L_\infty(x, y) = \max\{|x_1 - y_1|, \dots, |x_d - y_d|\}$$

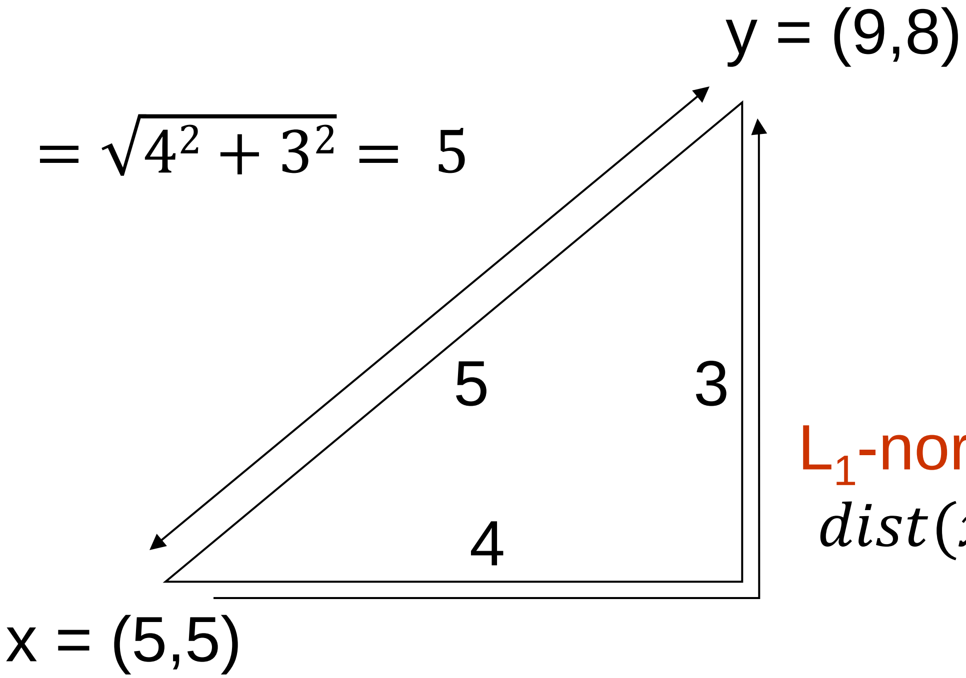
  - The limit of $\mathbf{L_p}$ as $p$ goes to infinity.

$L_p$ norms are known to be distance metrics

# Example of Distances

$L_2$-norm:

$$dist(x, y) = \sqrt{4^2 + 3^2} = 5$$



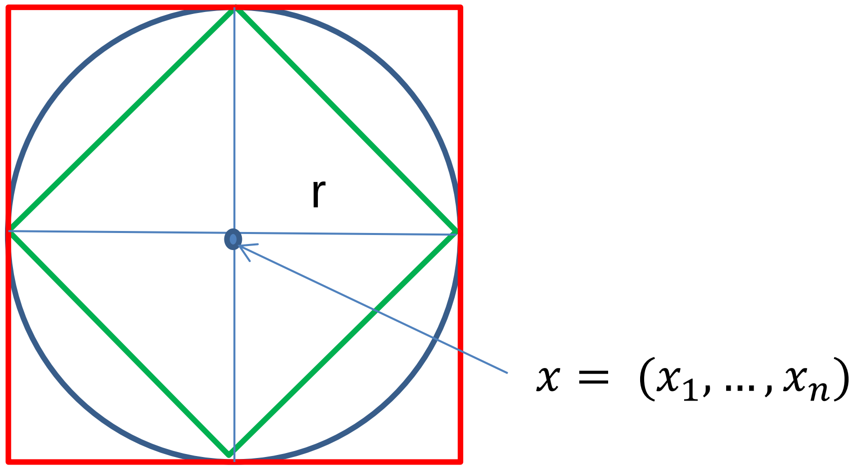

$L_1$-norm:

$$dist(x, y) = 4 + 3 = 7$$

$L_\infty$-norm:

$$dist(x, y) = \max\{3,4\} = 4$$

# Example




Green: All points y at distance $L_1(x,y) = r$ from point x

Blue: All points y at distance $L_2(x,y) = r$ from point x

Red: All points y at distance $L_\infty(x,y) = r$ from point x

# $L_p$ distances for sets

- We can apply all the $L_p$ distances to the cases of sets of attributes, with or without counts, if we represent the sets as vectors
  - E.g., a transaction is a 0/1 vector
  - E.g., a document is a vector of counts.

# Similarities into distances

- Jaccard distance:

$$JDist(X,Y) = 1 - JSim(X,Y)$$

- Jaccard Distance is a metric

- Cosine distance:

$$Dist(X,Y) = 1 - \cos(X,Y)$$

- Cosine distance is a metric

# Hamming Distance

- Hamming distance is the number of positions in which bit-vectors differ.
  - Example: $p_1 = 10101$
    $p_2 = 10011$.
    - $d(p_1, p_2) = 2$ because the bit-vectors differ in the 3rd and 4th positions.
    - The $L_1$ norm for the binary vectors
- Hamming distance between two vectors of categorical attributes is the number of positions in which they differ.
  - Example: x = (married, low income, cheat),
    y = (single, low income, not cheat)
  - d(x,y) = 2

# Why Hamming Distance Is a Distance Metric

- d(x,x) = 0 since no positions differ.
- d(x,y) = d(y,x) by symmetry of "different from."
- d(x,y) $\geq$ 0 since strings cannot differ in a negative number of positions.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.
- For binary vectors if follows from the fact that $L_1$ norm is a metric

# Distance between strings

- How do we define similarity between strings?

| | |
|---|---|
| weird | wierd |
| intelligent | unintelligent |
| Athena | Athina |

- Important for recognizing and correcting typing errors and analyzing DNA sequences.

# Edit Distance for strings

- The edit distance of two strings is the number of inserts and deletes of characters needed to turn one into the other.
- Example: x = abcde ; y = bcduve.
  - Turn *x* into *y* by deleting a, then inserting u and v after d.
  - Edit distance = 3.
- Minimum number of operations can be computed using dynamic programming
- Common distance measure for comparing DNA sequences

# Why Edit Distance Is a Distance Metric

- d(x,x) = 0 because 0 edits suffice.
- d(x,y) = d(y,x) because insert/delete are inverses of each other.
- d(x,y) $\geq$ 0: no notion of negative edits.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*. The minimum is no more than that

# Variant Edit Distances

- Allow insert, delete, and mutate.
  - Change one character into another.
- Minimum number of inserts, deletes, and mutates also forms a distance measure.

- Same for any set of operations on strings.
  - Example: substring reversal or block transposition OK for DNA sequences
  - Example: character transposition is used for spelling

# Distances between distributions

- We can view a document as a distribution over the words

| document | Apple | Microsoft | Obama | Election |
|---|---|---|---|---|
| D1 | 0.35 | 0.5 | 0.1 | 0.05 |
| D2 | 0.4 | 0.4 | 0.1 | 0.1 |
| D2 | 0.05 | 0.05 | 0.6 | 0.3 |

- KL-divergence (Kullback-Leibler) for distributions P,Q

$$D_{KL}(P \| Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- KL-divergence is asymmetric. We can make it symmetric by taking the average of both sides

$$\frac{1}{2} D_{KL}(P \| Q) + \frac{1}{2} D_{KL}(Q \| P)$$

- JS-divergence (Jensen-Shannon)

$$JS(P, Q) = \frac{1}{2} D_{KL}(P \| M) + \frac{1}{2} D_{KL}(Q \| M)$$

$$M = \frac{1}{2}(P + Q)$$

M: Average distribution

# Why is similarity important?

- We saw many definitions of similarity and distance
- How do we make use of similarity in practice?
- What issues do we have to deal with?

# APPLICATIONS OF SIMILARITY: RECOMMENDATION SYSTEMS

# An important problem

- Recommendation systems
  - When a user buys an item (initially books) we want to recommend other items that the user may like
  - When a user rates a movie, we want to recommend movies that the user may like
  - When a user likes a song, we want to recommend other songs that they may like

- A big success of data mining
- Exploits the long tail
  - How Into Thin Air made Touching the Void popular

# Utility (Preference) Matrix

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

How can we fill the empty entries of the matrix?

# Recommendation Systems

- Content-based:
  - Represent the items into a feature space and recommend items to customer C similar to previous items rated highly by C
    - Movie recommendations: recommend movies with same actor(s), director, genre, …
    - Websites, blogs, news: recommend other sites with "similar" content

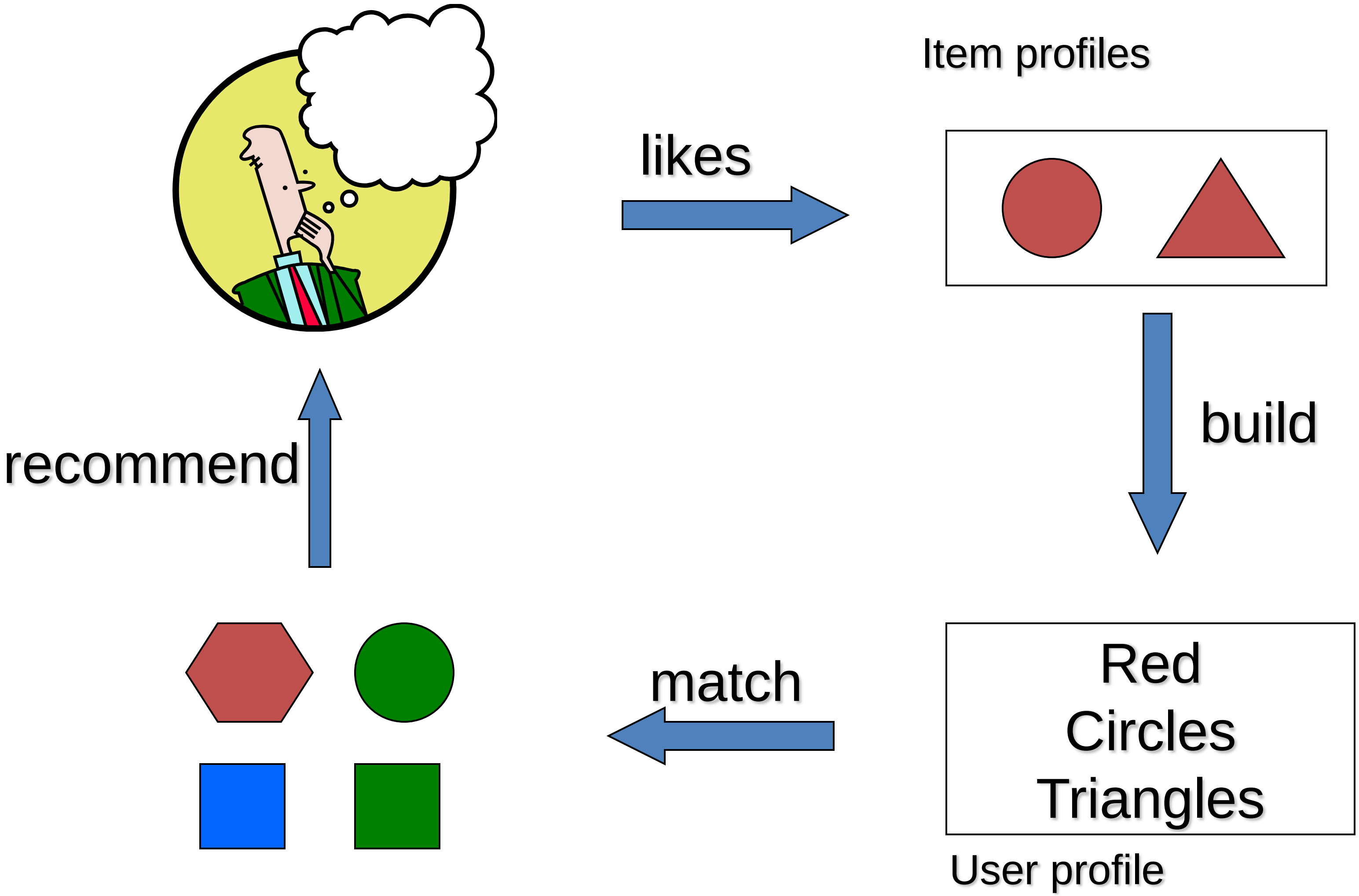
# Content-based prediction

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

Someone who likes one of the Harry Potter (or Star Wars) movies is likely to like the rest

- Same actors, similar story, same genre

# Intuition


Item profiles
likes
build
Red
Circles
Triangles
User profile
match
recommend

# Approach

- Map items into a feature space:
  - For mocvies:
    - Actors, directors, genre, rating, year,…
    - Challenge: make all features compatible.
  - For documents?

- To compare items with users we need to map users to the same feature space. How?
  - Take all the movies that the user has seen and take the average vector
    - Other aggregation functions are also possible.

- Recommend to user C the most similar item I computing similarity in the common feature space
  - Distributional distance measures also work well.

# Limitations of content-based approach

- Finding the appropriate features
  - e.g., images, movies, music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
- Recommendations for new users
  - How to build a profile?

# Collaborative filtering

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

Two users are similar if they rate the same items in a similar way

Recommend to user C, the items
liked by many of the most similar users.

# User Similarity

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

Which pair of users do you consider as the most similar?

What is the right definition of similarity?

# User Similarity

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 1 | | | 1 | 1 | | |
| B | 1 | 1 | 1 | | | | |
| C | | | | 1 | 1 | 1 | |
| D | | 1 | | | | | 1 |

Jaccard Similarity: users are sets of movies

Disregards the ratings.
Jsim(A,B) = 1/5
Jsim(A,C) = Jsim(B,D) = 1/2

# User Similarity

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

## Cosine Similarity:

Assumes zero entries are negatives:
Cos(A,B) = 0.38
Cos(A,C) = 0.32

# User Similarity

| | Harry Potter 1 | Harry Potter 2 | Harry Potter 3 | Twilight | Star Wars 1 | Star Wars 2 | Star Wars 3 |
|---|---|---|---|---|---|---|---|
| A | 2/3 | | | 5/3 | -7/3 | | |
| B | 1/3 | 1/3 | -2/3 | | | | |
| C | | | | -5/3 | 1/3 | 4/3 | |
| D | | 0 | | | | | 0 |

Normalized Cosine Similarity:

- Subtract the mean and then compute Cosine (correlation coefficient)

Corr(A,B) = 0.092
Cos(A,C) = -0.559

# User-User Collaborative Filtering

- Consider user c
- Find set D of other users whose ratings are most "similar" to c's ratings
- Estimate user's ratings based on ratings of users in D using some aggregation function

- Advantage: for each user we have small amount of computation.

# Item-Item Collaborative Filtering

- We can transpose (flip) the matrix and perform the same computation as before to define similarity between items
  - Intuition: Two items are similar if they are rated in the same way by many users.
  - Better defined similarity since it captures the notion of genre of an item
    - Users may have multiple interests.
- Algorithm: For each user c and item i
  - Find the set D of most similar items to item i that have been rated by user c.
  - Aggregate their ratings to predict the rating for item i.
- Disadvantage: we need to consider each user-item pair separately

# Pros and cons of collaborative filtering

- Works for any kind of item
  - No feature selection needed
- New user problem
- New item problem
- Sparsity of rating matrix
  - Cluster-based smoothing?

# SKETCHING AND LOCALITY SENSITIVE HASHING

Thanks to:

Rajaraman and Ullman, “Mining Massive Datasets”

Evimaria Terzi, slides for Data Mining Course.

# Another important problem

- Find duplicate and near-duplicate documents from a web crawl.
- Why is it important:
  - Identify mirrored web pages, and avoid indexing them, or serving them multiple times
  - Find replicated news stories and cluster them under a single story.
  - Identify plagiarism
- What if we wanted exact duplicates?

# Finding similar items

- Both the problems we described have a common component
  - We need a quick way to find highly similar items to a query item
  - OR, we need a method for finding all pairs of items that are highly similar.
- Also known as the Nearest Neighbor problem, or the All Nearest Neighbors problem

- We will examine it for the case of near-duplicate web documents.

# Main issues

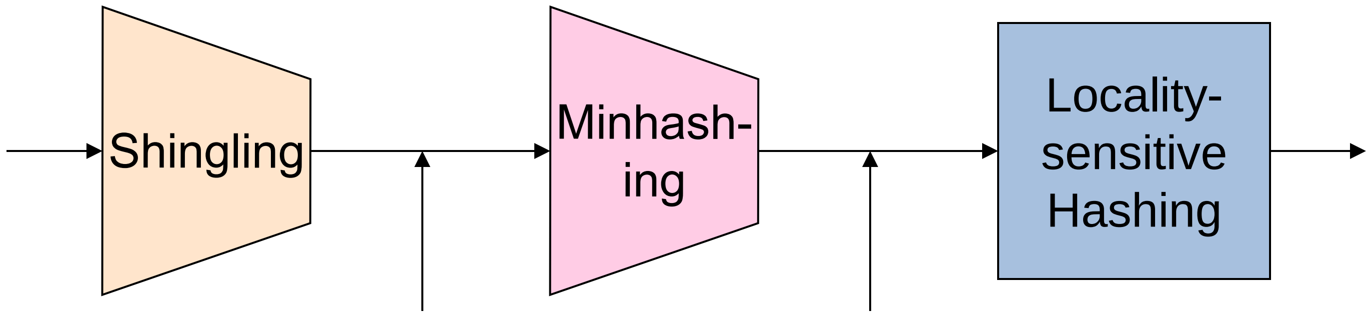
- What is the right representation of the document when we check for similarity?
  - E.g., representing a document as a set of characters will not do (why?)
- When we have billions of documents, keeping the full text in memory is not an option.
  - We need to find a shorter representation
- How do we do pairwise comparisons of billions of documents?
  - If we wanted exact match it would be ok, can we replicate this idea?

# Three Essential Techniques for Similar Documents

1. Shingling : convert documents, emails, etc., to sets.
2. Minhashing : convert large sets to short signatures, while preserving similarity.
3. Locality-Sensitive Hashing (LSH): focus on pairs of signatures likely to be similar.

# The Big Picture

Docu-
ment




Candidate pairs : those pairs of signatures that we need to test for similarity.

The set of strings of length *k* that appear in the doc-ument

Signatures : short integer vectors that represent the sets, and reflect their similarity

# Shingles

- A k -shingle (or k -gram) for a document is a sequence of k characters that appears in the document.
- Example: document = abcab. k=2
  - Set of 2-shingles = {ab, bc, ca}.
  - Option: regard shingles as a bag, and count ab twice.

- Represent a document by its set of k-shingles.

# Shingling

- Shingle: a sequence of k contiguous characters

```
a rose is a rose is a rose
a rose is 
 rose is a
  rose is a 
   ose is a r
    se is a ro
     e is a ros
      is a rose
       is a rose 
        s a rose i
         a rose is
          a rose is 
```

# Working Assumption

- Documents that have lots of shingles in common have similar text, even if the text appears in different order.
- Careful: you must pick $k$ large enough, or most documents will have most shingles.
  - Extreme case $k = 1$: all documents are the same
  - $k = 5$ is OK for short documents; $k = 10$ is better for long documents.
- Alternative ways to define shingles:
  - Use words instead of characters
  - Anchor on stop words (to avoid templates)

# Shingles: Compression Option

- To compress long shingles, we can hash them to (say) 4 bytes.
- Represent a doc by the set of hash values of its *k*-shingles.
- From now on we will assume that shingles are integers
  - Collisions are possible, but very rare

# Fingerprinting

- Hash shingles to 64-bit integers