



Building a Recommendation Engine

agenda

1. an introduction to recommendation engines & how they work (three types!)
2. how can we code a recommendation engine?

// part 1: an introduction to recommendation engines

types of recommendation engines, and **why**???

- why do we need recommendation engines + what are some examples?

types of recommendation engines, and why???

- why do we need recommendation engines + what are some examples?
- three main types of recommendation engines:
 - a. non-personalized
 - b. content-based
 - c. collaborative filtering

non-personalized recommendation engines

The screenshot displays the YouTube homepage with a left sidebar and a main content area. The sidebar includes navigation links for Home, Trending, Subscriptions, Library, and History, along with a 'SIGN IN' button and a 'BEST OF YOUTUBE' section listing categories like Music, Sports, Gaming, Movies, TV Shows, News, Live, Spotlight, and 360° Video. The main content area features a 'YouTube Music' banner with the text 'Our new music streaming service is here.' and a 'LET'S GO' button. Below this is a 'Trending' section with five video thumbnails: 'SPIDER-MAN: FAR FROM HOME - Official Trailer', 'HIGHLIGHTS | Canelo Alvarez vs. Daniel Jacobs', 'Ping Pong Trick Shots 5 | Dude Perfect', 'Family Feud Cold Open - SNL', and '\$800 KITCHEN KNIFE'. The bottom section is 'YouTubeTV', featuring 'Live TV from 70+ networks' with a 'TRY YOUTUBE TV' button and three featured shows: 'The Voice', 'GREY'S ANATOMY', and 'SPORTSCENTER'.

YouTube

Search

YouTube Music

Our new music streaming service is here.

LET'S GO

Trending

SPIDER-MAN: FAR FROM HOME - Official Trailer
Sony Pictures Entertainment
31M views • 1 day ago

HIGHLIGHTS | Canelo Alvarez vs. Daniel Jacobs
DAZN USA
3.4M views • 2 days ago

Ping Pong Trick Shots 5 | Dude Perfect
Dude Perfect
8.4M views • 22 hours ago

Family Feud Cold Open - SNL
Saturday Night Live
3M views • 2 days ago

\$800 KITCHEN KNIFE
BuzzFeedVideo
1.5M views • 2 days ago

YouTubeTV Featured

Live TV from 70+ networks

\$49.99 per month
Cancel anytime

TRY YOUTUBE TV

The Voice
Mondays and Tuesdays 8/7c

GREY'S ANATOMY
Thursdays 8/7c

SPORTSCENTER
Tune in Daily 9/8c

content-based recommendation engines

- makes recommendations based on an item's **features**

movies	Genre	Actor	Director	Year	IMDB	Rotten Tomatoes	...
1							
2							
3							
4							
5							
...							

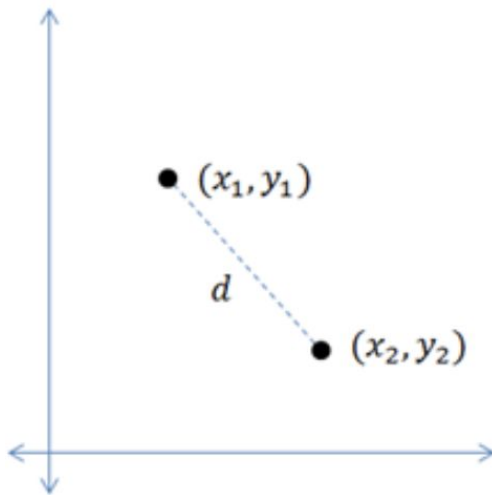
similarity metrics

- euclidean distance
- jaccard index
- cosine similarity

movies	Genre	Actor	Director	Year	IMDB	Rotten Tomatoes	...
1							
2							
3							
4							
5							
...							

similarity metrics

- **euclidean distance**
- jaccard index
- cosine similarity

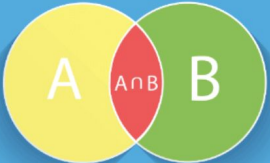


$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

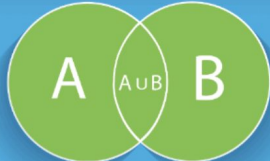
similarity metrics

- euclidean distance
- **jaccard index**
- cosine similarity

Jaccard Index



A Venn diagram illustrating the Jaccard Index. It consists of two overlapping circles, A and B. Circle A is yellow and circle B is green. The intersection of A and B is shaded red and labeled $A \cap B$.

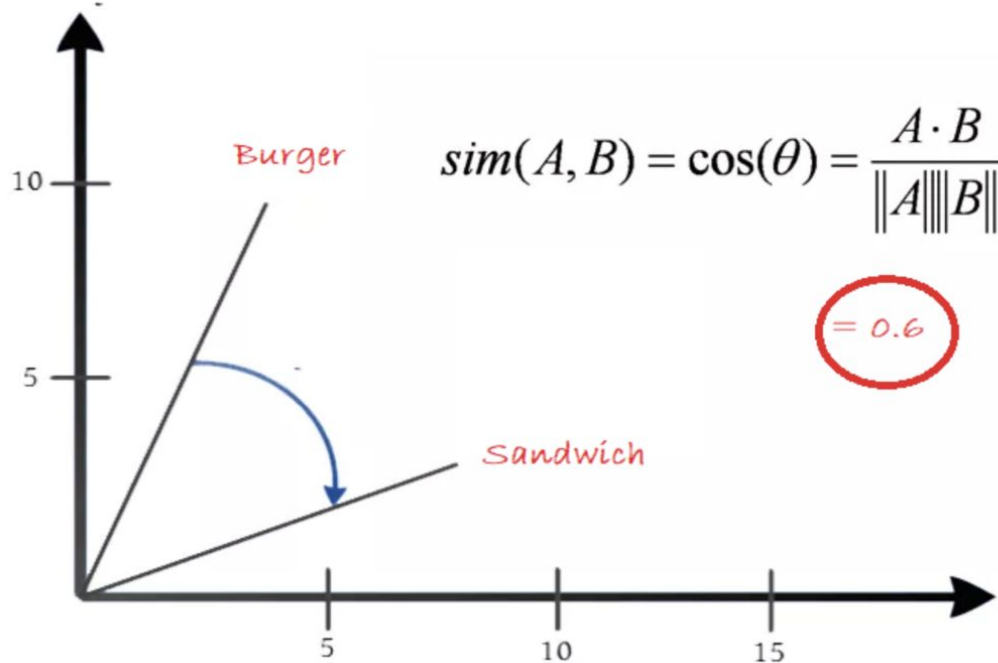
$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$


A second Venn diagram, identical in structure to the first, showing two overlapping circles A and B. In this diagram, both circles are green. The intersection is labeled $A \cup B$, which represents the union of the two sets.

similarity metrics

- euclidean distance
- jaccard index
- **cosine similarity**

Cosine Similarity

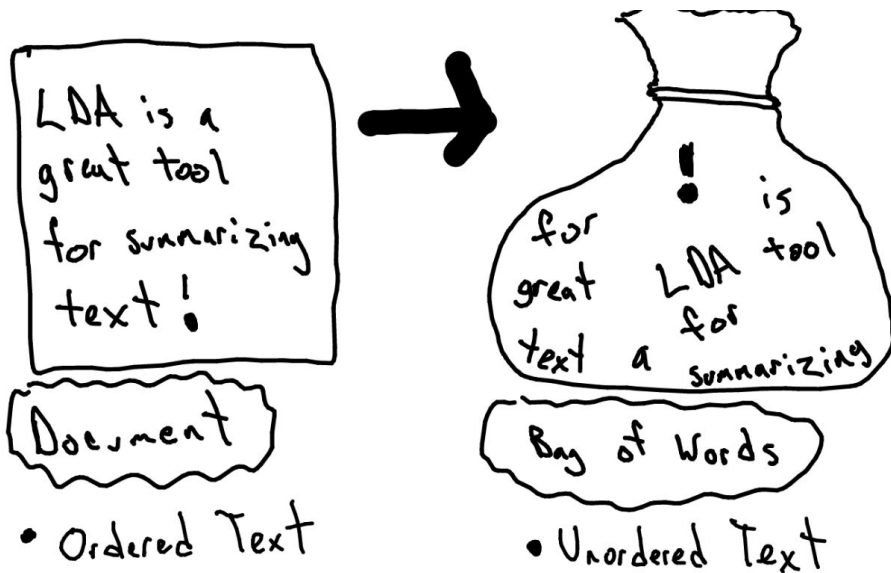


similarity metrics

- Now that we have learned how to calculate distance metrics, how do we calculate that for documents?
- Two options:
 - Bag of words
 - tf-idf

similarity metrics

- Now that we have learned how to calculate distance metrics, how do we calculate that for documents?
- Two options:
 - **Bag of words**
 - tf-idf



similarity metrics

Example:

1. "I love dogs"
2. "Dogs are awesome pets"
3. "I have four pets, two dogs, a guinea pig, and a cat, and i love them" →

i	love	dogs	are	awes ome	pets	have	four	two	a	guine a	pig	cat
1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0	0
2	1	1	0	0	1	1	1	1	2	1	1	1

similarity metrics

- Now that we have learned how to calculate distance metrics, how do we calculate that for documents?
- Two options:
 - Bag of words
 - **Tf-idf**
 - **TF: Term Frequency:**
 $TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$.
 - **IDF: Inverse Document Frequency:**
 $IDF(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it})$
 - $TF\text{-}IDF = TF * IDF$

similarity metrics

- Now that we have learned how to calculate distance metrics, how do we calculate that for documents?
- Documents will then be turned into tf-idf matrix, where each document is an observation, and each column represent a token
- Each document can then be represented as a vector in an n dimensional space, where n stands for number of tokens
- We can then calculate a cosine similarity matrix of $p \times p$, where p is the number of documents

content-based recommendation engines

- what are some pros and some pitfalls of content-based recommendations?



collaborative filtering

- recommends items **based on ratings of other users**
- different ways to do collaborative filtering:
 - Memory-based(aka neighborhood based):
 - Item-item similarity
 - User-user similarity
 - model-based: Singular Vector Decomposition (SVD)



collaborative filtering -- the utility matrix

- a utility matrix shows user ratings of different items
- the idea is to fill in the blanks

	Movie 1	Movie 2	Movie ...	Movie N
User 1	1	BLANK	BLANK	3
User 2	BLANK	5	BLANK	3
User 3	BLANK	BLANK	1	BLANK
User 4	2	3	BLANK	BLANK
User 5	BLANK	BLANK	1	BLANK
User 6	4	BLANK	5	BLANK
User 7	BLANK	4	BLANK	BLANK
User ...	BLANK	3	BLANK	BLANK
User m	BLANK	BLANK	BLANK	4



	Movie 1	Movie 2	Movie ...	Movie N
User 1	1	4	2	3
User 2	1	5	3	3
User 3	2.5	2.8	1	3.5
User 4	2	3	2	3.5
User 5	2.5	2.8	1	3.1
User 6	4	1.2	5	1.4
User 7	1	4	2.5	3
User ...	2	3	2	3
User m	1	4	2	4

user-user collaborative filtering

filling up the utility matrix based on **user** similarities

to get recommendations for user X:

1. get user similarity values
2. the predicted rating for item A is a **weighted average** of others' ratings

	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

cos	U1
U2	0.65
U3	0.76
U4	0.83

sum = 2.24

$$\begin{aligned} ? &= (0.65*4 + 0.76*5 + 0.83*4)/2.24 \\ &= \mathbf{4.34} \end{aligned}$$

item-item collaborative filtering

filling up the utility matrix based on **item** similarities

to get recommendations for user X:

1. get ITEM similarity values
2. the predicted rating for item A is a **weighted average** of other **items**

	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

cos	I3
I1	0.78
I2	0.83
I4	0.87

sum = 2.48

$$\begin{aligned} ? &= (0.78*4 + 0.83*3 + 0.87*3)/2.48 \\ &= \mathbf{3.31} \end{aligned}$$

user-user vs item-item

which is better?????

- in general, item-item has proven to be more effective
- it's hard to predict users' *unique* tastes

time complexity (for m users and n items)

- user-user: $O(m^2n)$
- item-item: $O(mn^2)$
- which would be faster if $m > n$ (more users than items)?

model-based collaborative filtering

singular value decomposition:

- another way to fill in the utility matrix via **matrix factorization**

modified SVD in recommendation engines:

- breaks down the utility matrix into a **user matrix** and an **item matrix**
- the other dimensions are **latent features**
- **gradient descent** using Alternating Least Squares to preserve the relationship between items and between users (parallelizable)

very math, but **best-in-class** models use some form of SVD

SVD for filling in utility matrices

	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

\approx

I1	x	x
I2	x	x
I3	x	x
I4	x	x

\cdot

U1	U2	U3	U4
x	x	x	x
x	x	x	x

collaborative filtering -- pros and cons

- personalized for each user!
- computationally heavy
- popularity bias
- the **cold start** problem

stuff we've learned

recommendation engines!!!

1. non-personalized		
2. content-based		
collaborative filtering	memory-based	3. user-user
		4. item-item
	model-based	5. SVD

// part 2: coding our own
recommendation engine!

stuff we've learned

recommendation engines!!!

1. non-personalized
2. content-based
3. model-based collaborative filtering with SVD

code-along: <https://github.com/jessicafangfanglee/hotel-recommendation>