COLLABORATIVE FILTERING ON MOVIE RATING DATASET

Presentation prepared for MDLE
Pedro Cruz, nº89997

LOCALITY SENSITIVE HASHING (LSH)

- What is it?
- How does it work?
- Properties of LSH
- What is it used for?

WHAT IS IT AND HOW DOES IT WORK?

Important Concepts:
MinHashing
Banding
Similarity

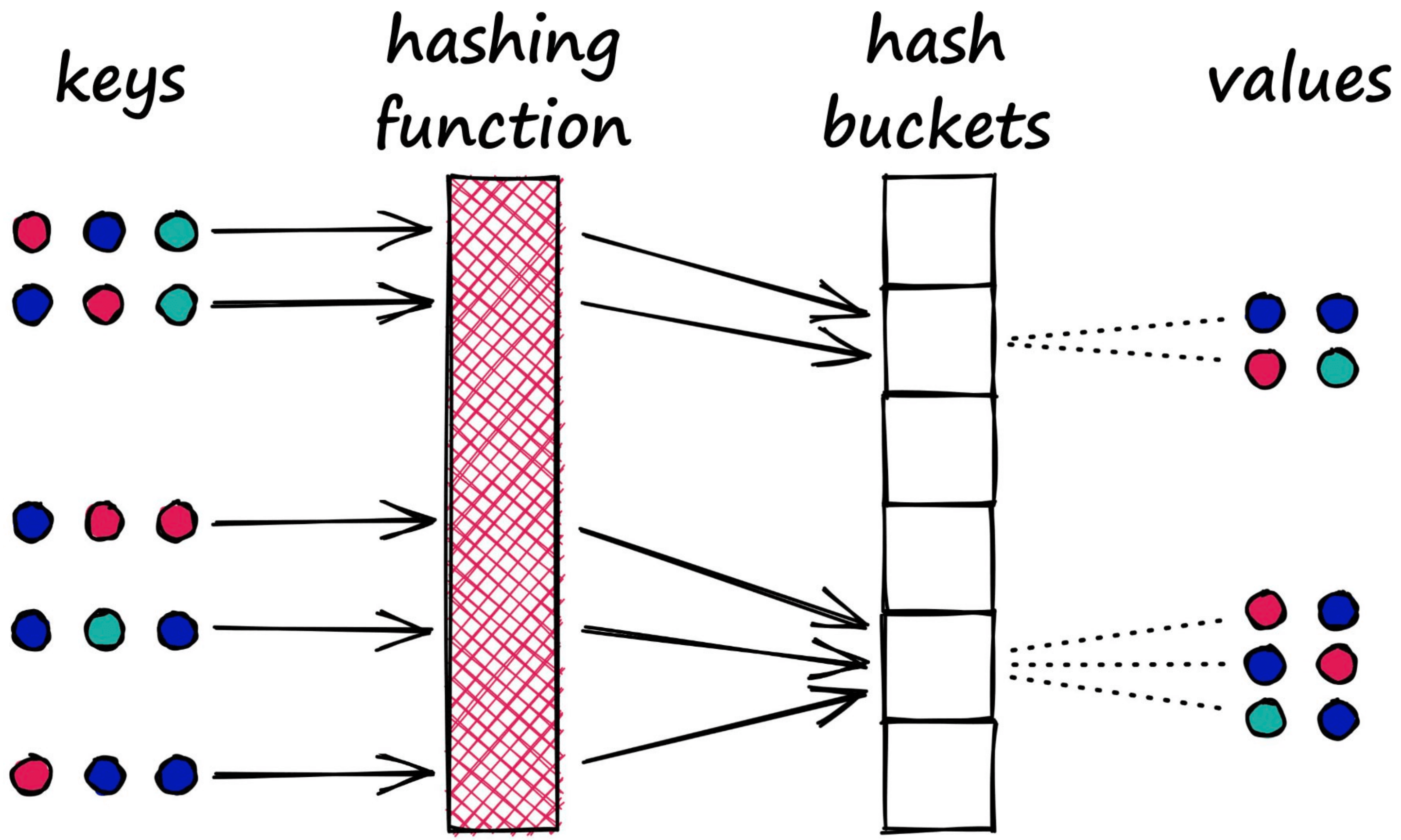

Image 1 – LSH diagram taken from https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing/

PROPRIETIES OF LSH

Reduces Complexity.

Fast compared to other similar methods.

Allows for the processing of huge amounts of data with "limited" resources.

It's not perfect, it has trade-offs as well.

- Efficiency
- Speed
- Scalability
- Probabilistic Nature

WHAT IS IT USED FOR?

- Collaborative Filtering
- Content Based Filtering
- Plagiarism Detection
- Plenty more uses.

COLLABORATIVE FILTERING

- User-User
- User-Item
- Item-Item

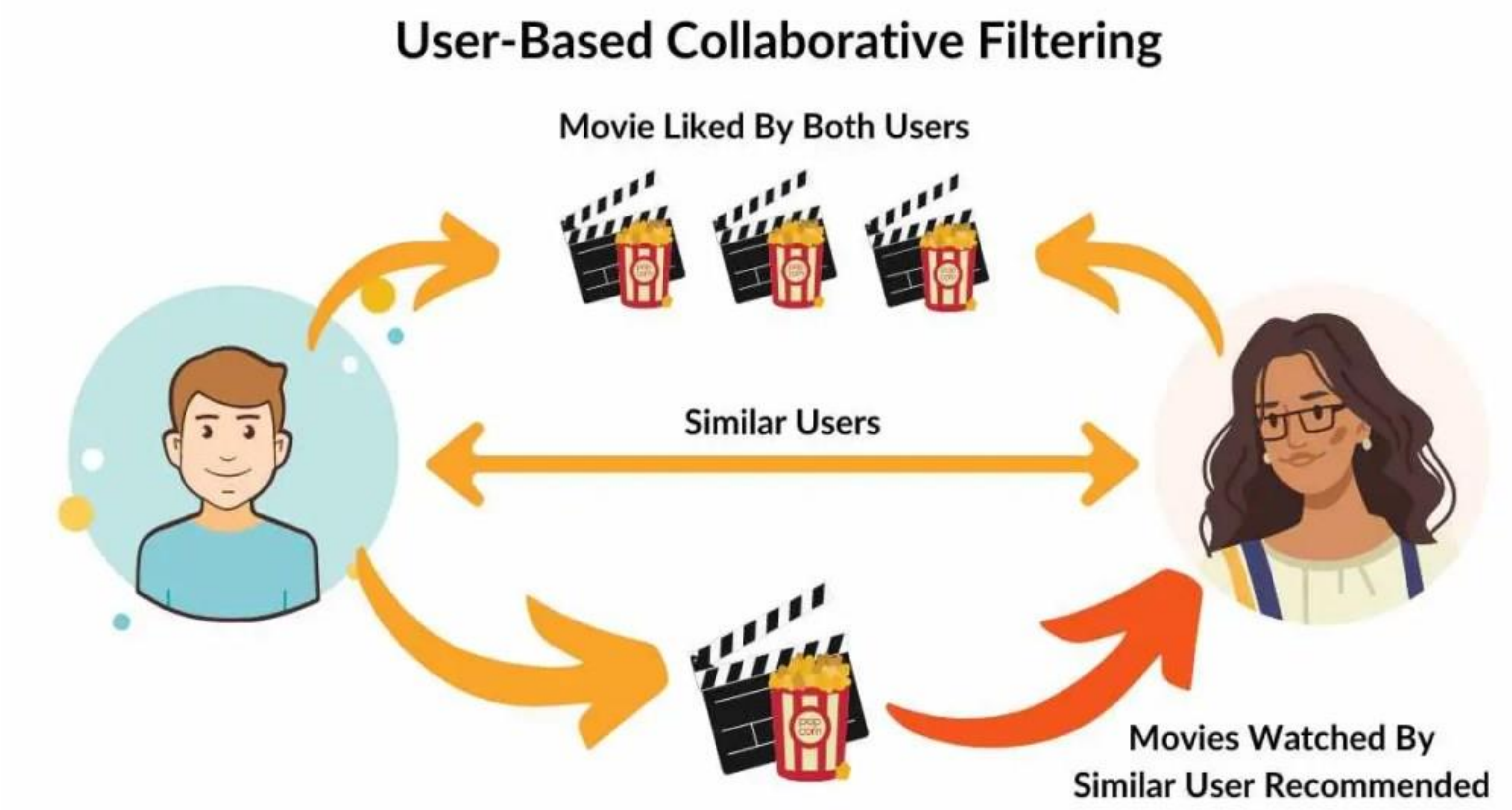

Image 2 – Collaborative Filtering diagram taken from https://spotintelligence.com/2024/04/25/collaborative-filtering/

IMPLEMENTATION

- Dataset limited to 100K, 1M and 3M samples.
- Implemented on Jupyter Notebook.
- Implemented with pyspark.
- Jaccard Distance = 0.3
- Default rating system.
- Predict Ratings
- Validate results.

DATASET

userId	movieId	rating	timestamp
1	1	4.0	1225734739
1	110	4.0	1225865086
1	158	4.0	1225733503
1	260	4.5	1225735204
1	356	5.0	1225735119
1	381	3.5	1225734105
1	596	4.0	1225733524
1	1036	5.0	1225735626
1	1049	3.0	1225734079
1	1066	4.0	1225736961
1	1196	3.5	1225735441
1	1200	3.5	1225735861
1	1210	4.5	1225735210
1	1214	4.0	1225736426
1	1291	5.0	1225734809
1	1293	2.0	1225733842
1	1376	3.0	1225733539
1	1396	3.0	1225733534
1	1537	4.0	1225736687
1	1909	3.0	1225733717

1. AVAILABLE AT HTTPS://GROUPLENS.ORG/DATASETS/MOVIELENS/

2. CONSISTS OF 33 MILLION SAMPLES.

3. EACH SAMPLE HAS A USER PAIRED TO A MOVIE AND A RATING.

DATA PREPROCESSING

userId	movieId	rating	timestamp
1	1	4.0	1225734739
1	110	4.0	1225865086
1	158	4.0	1225733503
1	260	4.5	1225735204
1	356	5.0	1225735119
1	381	3.5	1225734105
1	596	4.0	1225733524
1	1036	5.0	1225735626
1	1049	3.0	1225734079
1	1066	4.0	1225736961
1	1196	3.5	1225735441
1	1200	3.5	1225735861
1	1210	4.5	1225735210
1	1214	4.0	1225736426
1	1291	5.0	1225734809
1	1293	2.0	1225733842
1	1376	3.0	1225733539
1	1396	3.0	1225733534
1	1537	4.0	1225736687
1	1909	3.0	1225733717

userId	movieId	rating	userId-rating
1	1	4.0	1-4.0
1	1036	5.0	1-5.0
1	1049	3.0	1-3.0
1	1066	4.0	1-4.0
1	110	4.0	1-4.0
1	1196	3.5	1-3.5
1	1210	4.5	1-4.5
1	1291	5.0	1-5.0
1	1293	2.0	1-2.0
1	1376	3.0	1-3.0
1	1396	3.0	1-3.0
1	1537	4.0	1-4.0
1	158	4.0	1-4.0
1	1909	3.0	1-3.0
1	1959	4.0	1-4.0
1	1960	4.0	1-4.0
1	2028	5.0	1-5.0
1	2085	3.5	1-3.5
1	2116	4.0	1-4.0
1	2336	3.5	1-3.5

- Load dataset.
- Limit dataset size.
- Filter unnecessary features.
- Splitt into Train(90%) and Test(10%).
- Prepare the Item-Item filtering.

MINHASHING

Grouping by item (movieId)

movieId	Id-Rates
1	[1000-3.5, 10000-...
10	[10007-5.0, 10009...
100	[10023-4.0, 10099...
1000	[10132-4.0, 10410...
100001	[24480-4.0]
100008	[4074-3.0, 4880-3...
100017	[10060-3.5, 10092...
100032	[9401-4.5]
100034	[2651-2.5]
100036	[11436-3.0, 23925...
100038	[18931-3.0]
100044	[10109-5.0, 10609...
100046	[15661-2.5, 3469-...
100048	[14404-3.5, 16466...
100054	[8833-4.0]
100058	[13085-4.5, 15528...
100060	[15370-3.5]
100062	[14521-4.5, 15493...
100070	[11329-3.0, 16475...
100072	[24160-2.0, 7716-...

"Creating" Sparse matrix for our signatures using CountVectorizer or HashingTF.

$$
\begin{array}{c|ccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 3 & 0 & 0 & 7 & 0 \\
4 & 2 & 0 & 0 & 9 & 0 & 0 & 0 \\
5 & 0 & 0 & 8 & 0 & 0 & 0 & 0
\end{array}
$$

Minhashing Results

We are looking for movies with similar ratings by similar users.

movieId	Id-Rates	vectorized_rates	hashed_rates
1	[1000-3.5, 1001-5...	(65536,[65,71,78,...	[[431617.0], [606...
10	[1001-4.0, 1003-3...	(65536,[99,308,35...	[[1380108.0], [18...
100	[1109-5.0, 1324-3...	(65536,[681,717,2...	[[1.3880476E7], [...
1000	[2401-4.0, 3078-4...	(65536,[24495,302...	[[1.214088191E9],...
100008	[4074-3.0, 4880-3...	(65536,[3865,3283...	[[7.62221094E8], ...
100017	[6001-3.5, 7629-3.5]	(65536,[26398,405...	[[1.136325078E9],...
100032	[9401-4.5]	(65536,[34386],[1...	[[1.69828445E9], ...
100034	[2651-2.5]	(65536,[53851],[1...	[[5.5908619E8], [...
100038	[9068-3.5]	(65536,[31590],[1...	[[8.99163005E8], ...
100044	[2589-4.0, 305-4....	(65536,[2549,4009...	[[5.204915E8], [2...
100046	[3469-3.0, 3917-3...	(65536,[25537,315...	[[5.606449E8], [8...
100054	[8833-4.0]	(65536,[61557],[1...	[[4.21679155E8], ...
100058	[305-2.0]	(65536,[36741],[1...	[[9.9304489E7], [...
100062	[8204-3.0]	(65536,[23240],[1...	[[1.254875851E9],...
100081	[8833-2.0]	(65536,[58195],[1...	[[1.415767196E9],...
100083	[1117-4.0, 1195-2...	(65536,[259,857,4...	[[2.00304006E8], ...
100087	[389-3.0]	(65536,[34060],[1...	[[4.98602386E8], ...
100091	[8833-2.5]	(65536,[59593],[1...	[[7.96290547E8], ...
100106	[4202-4.5, 4249-5...	(65536,[378,1164,...	[[1.32787764E8], ...
100108	[2004-3.5, 2172-1...	(65536,[551,1470,...	[[1.12185413E8], ...

datasetA	datasetB	JaccardDistance
{262259, [7341-4....	{262413, [7341-4....	0.0
{171129, [1312-2....	{82110, [1312-2.5...	0.0
{193439, [6008-4....	{212365, [6008-4....	0.0
{127230, [6008-4....	{193439, [6008-4....	0.0
{127230, [6008-4....	{212365, [6008-4....	0.0
{127230, [6008-4....	{127232, [6008-4....	0.0
{127230, [6008-4....	{127238, [6008-4....	0.0
{127232, [6008-4....	{193439, [6008-4....	0.0
{127232, [6008-4....	{212365, [6008-4....	0.0
{127232, [6008-4....	{127238, [6008-4....	0.0
{127238, [6008-4....	{193439, [6008-4....	0.0
{127238, [6008-4....	{212365, [6008-4....	0.0
{185423, [26257-3...	{189495, [26257-3...	0.0
{185423, [26257-3...	{189503, [26257-3...	0.0
{185423, [26257-3...	{203649, [1811-4....	0.0
{189495, [26257-3...	{189503, [26257-3...	0.0
{189495, [26257-3...	{203649, [1811-4....	0.0
{189503, [26257-3...	{203649, [1811-4....	0.0
{270754, [2172-3....	{271016, [2172-3....	0.0
{270754, [2172-3....	{271034, [2172-3....	0.0

movieA	movieB	Similarity
262259	262413	1.0
171129	82110	1.0
193439	212365	1.0
127230	193439	1.0
127230	212365	1.0
127230	127232	1.0
127230	127238	1.0
127232	193439	1.0
127232	212365	1.0
127232	127238	1.0
127238	193439	1.0
127238	212365	1.0
185423	189495	1.0
185423	189503	1.0
185423	203649	1.0
189495	189503	1.0
189495	203649	1.0
189503	203649	1.0
270754	271016	1.0
270754	271034	1.0

MEASURING DISTANCES

Distance chosen = 0.3 (Jaccard) (Using approxSimilarityJoin)

INEXISTENT NEIGHBORS PROBLEM APPROACH

The developed Item-Item Collaborative Filtering has a problem:

- What happens when we try to predict the rating of a movie not rated by a user and not similar to any movie the user has seen?
- By this approach said movie wouldn't get a rating by said user.
- "Developed" a system that scores a movie by default.

userId	movieId	default_rating
6823	110173	3.5
13085	139783	4.0
11436	34226	3.0
11436	126975	3.0
7644	238108	5.0
13085	173015	2.5
15227	126562	0.5
11969	165505	2.0
1011	26613	4.0
13085	174125	3.5
11436	200006	2.0
11436	167600	3.0
25084	144322	3.5
8833	39305	2.5
18286	262099	1.0
26408	162032	0.5
13630	194985	2.0
8833	198919	2.5
8833	151741	3.0
8833	6716	3.0

RATING PREDICTION

Movies unrated by user

userId	movieId	default_rating
10088	100591	3.0
10108	100591	3.0
10134	100591	3.0
10186	100591	3.0
10189	100591	3.0
10281	100591	3.0
10294	100591	3.0
10350	100591	3.0
10387	100591	3.0
10617	100591	3.0
10637	100591	3.0
1088	100591	3.0
11254	100591	3.0
11303	100591	3.0
1148	100591	3.0
11509	100591	3.0
11568	100591	3.0
11742	100591	3.0
11753	100591	3.0
11910	100591	3.0

Movies unrated by user paired with Similar movies

userId	movieId	movieB	Similarity	default_rating
10088	100591	114900	1.0	3.0
10088	100591	124709	1.0	3.0
10088	100591	186961	1.0	3.0
10088	100591	95214	1.0	3.0
5656	100591	113223	1.0	3.0
5656	100591	124765	1.0	3.0
10108	100591	41714	1.0	3.0
10108	100591	92399	1.0	3.0
5816	100591	200700	1.0	3.0
5816	100591	272673	1.0	3.0
5849	100591	112261	1.0	3.0
10186	100591	159714	1.0	3.0
6129	100591	116919	1.0	3.0
6129	100591	198505	1.0	3.0
10189	100591	133227	1.0	3.0
10189	100591	145909	1.0	3.0
10281	100591	101485	1.0	3.0
10294	100591	160762	1.0	3.0
6445	100591	114900	1.0	3.0
6445	100591	191679	1.0	3.0

Rating prediction by averaging the ratings of similar movies by the user

userId	unrated_movie	Rate_prediction(with_default)
1	103576	3.5
1	105165	2.5
1	109812	4.0
1	110439	4.0
1	111624	2.5
1	112925	3.5
1	116945	3.5
1	123282	2.5
1	124507	0.5
1	131386	3.5
1	135785	2.0
1	140729	2.0
1	141137	1.5
1	144596	3.5
1	146190	0.875
1	146190	1.1428571428571428
1	146477	3.0
1	146698	3.5
1	148056	1.5
1	148140	3.5

RESULTS AND COMPARISON

user	movie	actual_rating	Rate_prediction(with_default)
10129	279246	4.0	1.0
15216	159853	3.0	3.5
3083	121475	2.0	4.0
6888	163995	3.0	4.0
12428	216408	4.0	4.0
18150	139966	3.5	3.5
19703	182829	3.5	3.0
20210	115711	4.0	4.87037037037037
20210	115711	4.0	5.0
23044	286737	4.5	4.0
24918	142490	2.0	2.5
6116	33261	4.0	2.75
6116	90104	2.0	3.0
7959	83962	3.0	5.0
9290	180039	2.5	2.5
9401	202057	4.5	4.0
14160	104563	3.0	0.5
18802	6117	3.0	2.5
19584	216408	3.0	4.0
27889	128097	4.0	4.0

RMSE = 1.326080173877682

% Correct Predictions = 0.2608695652173913

MAE = 1.0942028985507246