Movie Recommendation System

SHRI RAMSWAROOP MEMORIAL UNIVERSITY

Department of Computer science

Submitted to:
Anupam Singh

Submitted by:
Anchal Gupta
Siddharth Pandey

Why Recommender?

"We are leaving the age of information and entering the age of recommendation."

—— Chris Anderson in "The Long Tail"

Picture from: https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcTON_H3eMI4LSCvtrzSWK8beYhtXs_z70NwAYYA2sQCsjMssEsD

The Age of Recommendation

<u>Search:</u>

User → Items

<u>Recommend:</u>

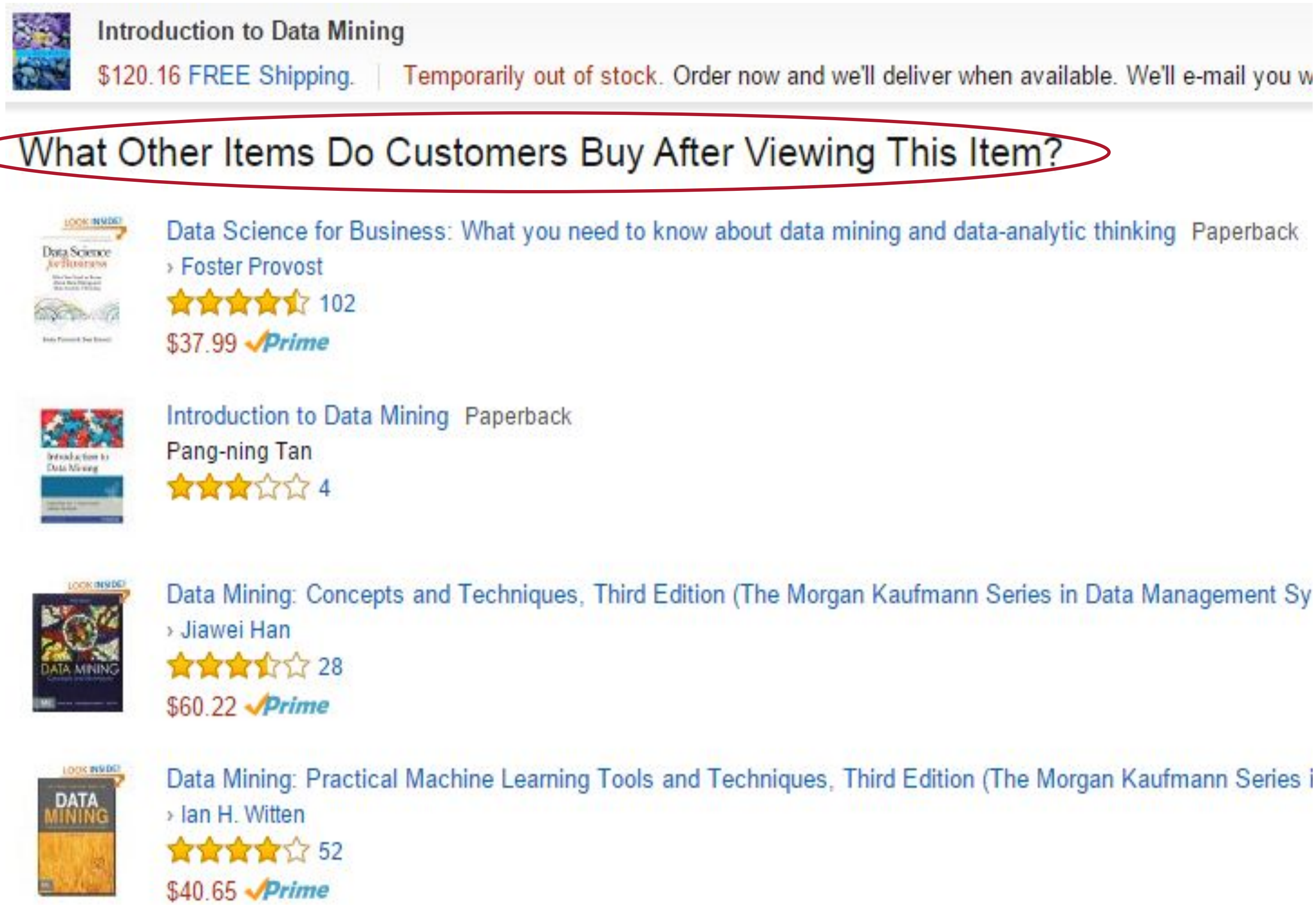
Items → User

Picture from: http://insidebigdata.com/wp-content/uploads/2014/06/Humor_recommender.jpg

Amazon: A personalized online store

Picture from: amazon.com

Amazon: A personalized online store



Picture from: amazon.com

Recommender Problem

A good recommender

- Show programming titles to a software engineer and baby toys to a new mother.
- Don’t recommend items user already knows or would find anyway.
- Expand user’s taste without offending or annoying him/her…

Challenges

- Huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Results are required to be returned in real time.
- New customers have limited information.
- Old customers can have a glut of information.
- Customer data is volatile.

Amazon’s solution

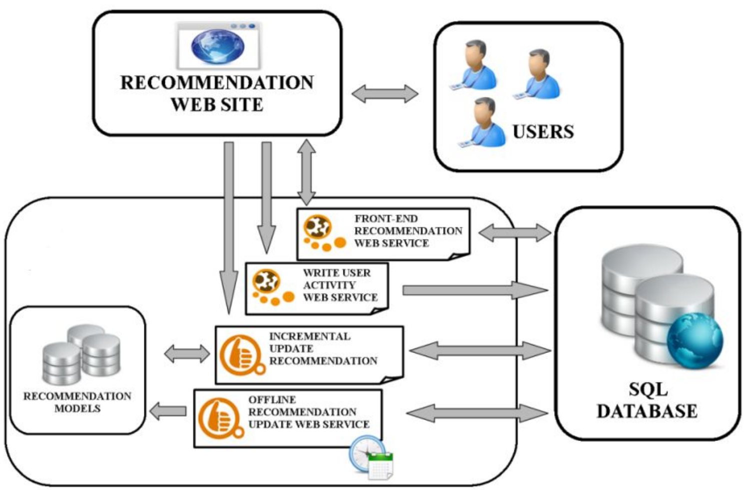
1. Amazon Recommendation Engine

- Amazon’s model that implements recommendation algorithm.
- Recommendation algorithm is designed to personalize the online store for each customer.

2. Algorithm feature

- Most recommendation algorithms start by finding a set of similar customers whose purchased and rated items overlap the user’s purchased and rated items.
- The Amazon’s item-to-item collaborative filtering is focusing on finding similar items instead of similar customers.

3. Recommendation Engine Workflow



Picture from:

Traditional Recommendation Algorithms

Two mostly used traditional algorithms:

1. User Based Collaborative Filtering

2. Cluster Models

User Based Collaborative Filtering

Approach

- Represents a customer as an N-dimensional vector of items
- Vector is positive for purchased or positively rated items and negative for negatively rated items
- Based on cosine similarity: finds similar customers/users

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|}$$

- Generates recommendations based on a few customers who are most similar to the user
- Rank each item according to how many similar customers purchased it

Problems

- **computationally expensive**, ***O(MN)*** in the worst case, where
  - M is the number of customers and
  - N is the number of items
- **dimensionality reduction** can increase the performance, BUT, also **reduce the quality of the recommendation**
- For very large data sets, such as **10 million customers and 1 million items**, the algorithm encounters **severe performance and scaling issues**

Cluster Models

Approach

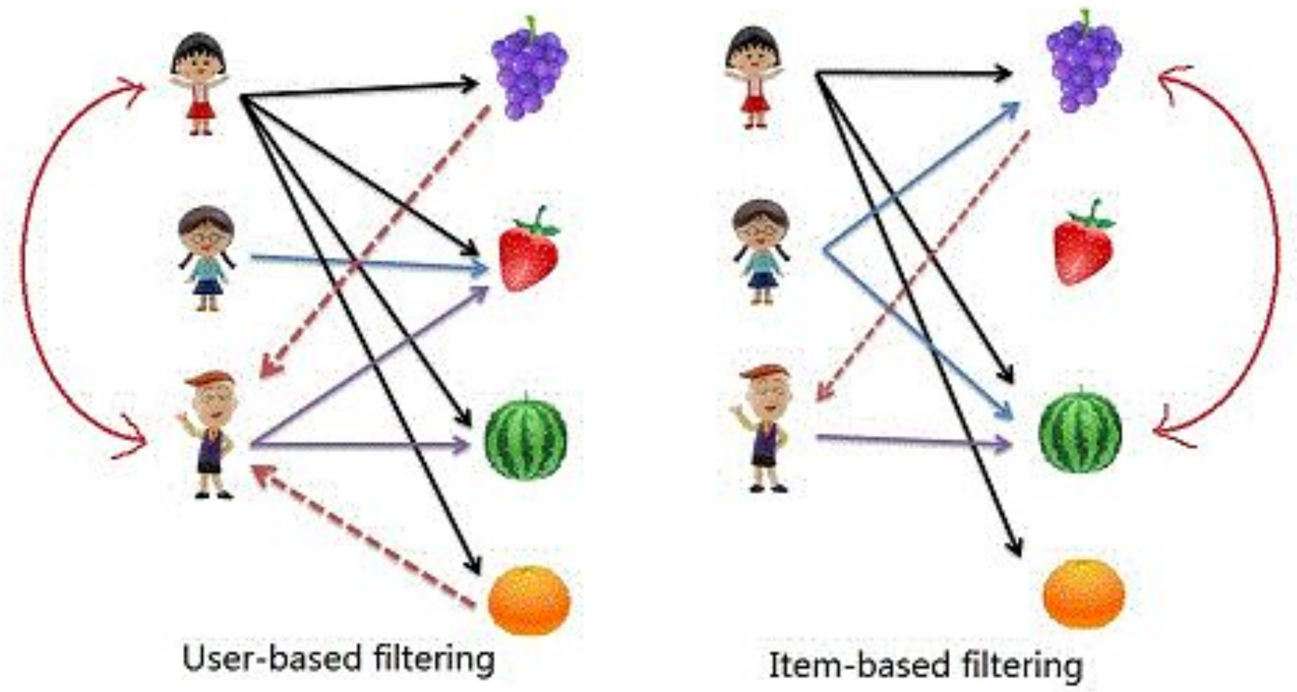
- Divide the customer base into many segments and treat the task as a classification problem
- Assign the user to the segment containing the most similar customers
- Uses the purchases and ratings of the customers in the segment to generate recommendations
- Cluster models have **better online scalability and performance** than collaborative filtering because they compare the user to a controlled number of segments rather than the entire customer base.

Problems

- **Quality of the recommendation is low**
- The recommendations are less relevant because the similar customers that the cluster models find are not the most similar customers
- **To improve quality**, it **needs online segmentation**, which is almost as **expensive as** finding similar customers using **collaborative filtering**

Amazon's Item-to-Item CF

- **Difference with User-to-User CF**



Picture from: http://paxcel.net/blog/wp-content/uploads/2013/08/recommendation_types.gif

Amazon’s Item-to-Item CF

Similarity of item i with item 17

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
,1	,3	,6	,1	,3	,4	,3	,3	,2	,6	,2	,5	,4	,5	,5	,3	1	,3	,5	,4	,2	,4	,4	,5

Items

Users

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3					2			4		2				
b			4							3							5	1		3					
c		5		4			4						3		5					4		5			
d								3				5				3			4		2			3	
e		3					5			4	5				5					1			5	4	
f			4				1		3	5		4	1		5	4	4		4				3		
g	2	4				4		2			5		1	4	5		4	2	4		5			4	
h			2		1		4		3	5		4	2		5	4	5						5		
i		1					3			5				5		4	4		5			4		3	
j			4			4				5			1		5		4		4				4		
k		5				4			2		5		1	5		4		2		4				2	
l					3			3				4	1		4		4	2	4					3	
m	5		3					5	3		5	4		5	5	3			4	4	5	4		4	
n			1		4	5				4	5		1	5		4		3		4		4	3		
o			4			4				5		4		5			4	2		5		5		3	
p				4			5								5	4		2	4	4	5	4		2	
q					3			3					1	5		4	4		4			4		3	
r		4			1	4		2					2		5		4				5	4		4	
s			2		4		4			5			1			4		2	4		4		5		
t		1		4			3					4		5	5		4			4				3	
u			2		1		4		3				1		5	4		2	4		5	4			
v					4	5				4	3		5			2					2			5	
w				2			2		3			5			4	5		4	2		3	4			
x	4			5				3		3				4	5					1					
y			1			3				2	3						3	3		5		4			

Picture from: Francesco Ricci, Slides [5]

Amazon's Item-to-Item CF

How It Works

- Matches each of the user's purchased and rated items to similar items
- Combines those similar items into a recommendation list

An iterative algorithm:

- **Builds a similar-items table** by finding items that customers tend to purchase together
- Provides a better approach by **calculating** the **similarity between a single product and all related products**:

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by customer C
      Record that a customer purchased I1 and I2
  For each item I2
    Compute the similarity between I1 and I2
```

- The **similarity** between two items uses the **cosine** measure
- Each **vector** corresponds to an **item** rather than a customer and
- Vector's **M dimensions correspond to customers** who have purchased that item

Offline computation : Online Recommendation

Offline Computation:

- **builds a similar-items table** which is extremely time intensive, $O(N^2M)$
- In practice, it's closer to O(NM), as most customers have very few purchases
- Sampling customers can also reduce runtime even further with little reduction in quality.

Online Recommendation:

- Given a similar-items table, the algorithm
  - **finds items** similar to each of the user's purchases and ratings,
  - **aggregates** those items, and then
  - **recommends the most popular or correlated items**.

Scalability and Quality: Comparison

User Based collaborative filtering:

- little or no offline computation
- impractical on large data sets, unless it uses dimensionality reduction, sampling, or partitioning
- dimensionality reduction, sampling, or partitioning reduces recommendation quality

Cluster models:

- can perform much of the computation offline,
- but recommendation quality is relatively poor

Item-to-Item collaborative filtering:

- scalability and performance are achieved by creating the expensive similar-items table offline
- online component "looking up similar items“ scales independently of the catalog size or the number of customers
- fast for extremely large data sets
- recommendation quality is excellent since it recommends highly correlated similar items
- unlike traditional collaborative filtering,
  - the algorithm performs well with limited user data,
  - producing high-quality recommendations based on as few as two or three items

Results:

- The MovieLens dataset contains 1 million ratings from 6,040 users on 3,900 movies.
- The best overall results are reached by the item-by-item based approach. It needs 170 seconds to construct the model and 3 seconds to predict 100,021 ratings.

	User Based	Model Based	Item Based
model construction time (sec.)	730	254	170
prediction time (sec.)	31	1	3
MAE	0.6688	0.6736	**0.6382**

Table from: Candillier, L., Meyer, F., & Boull'e Marc. (2007). Comparing state-of-the-art collaborative filtering systems [3]

Some Related Applications

- Pandora
- Netflix
- Google YouTube

Pandora music recommendation service

How It Works:

- Base its recommendation on data from Music Genome Project
- Assigns 400 attributes for each song, done by musicians, takes half an hour per song
- Use this method to find songs which is similar to user's favorite songs

Benefits:

- Accurate method, don't need lots of users information, needs little to get started

Drawback:

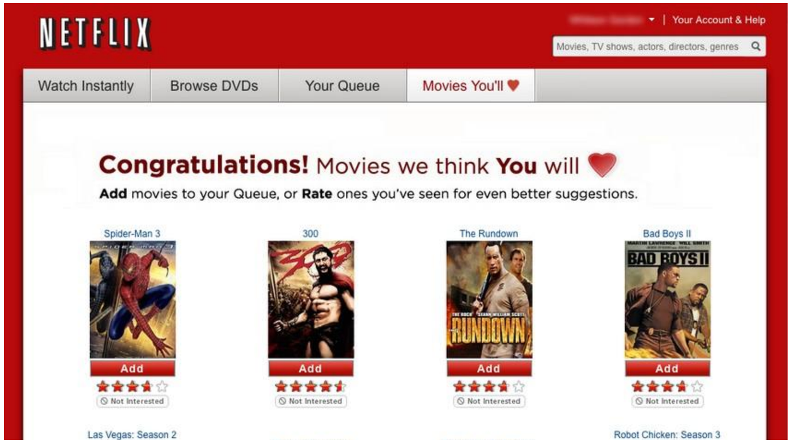
- Doesn't scale very well and often feels that Pandora's library is somewhat limited

Picture from: pandora.com

Netflix movie recommendation system

What's it

- Make recommendations by comparing the watching and the searching habits of similar users as well as by offering movies that share characteristics with films that a user has rated highly
- Collaborative, content-based, knowledge-based, and demographic techniques serves as the basis of its recommendation system. An ensemble method of 107 different algorithmic approaches, blended into a single prediction



Benefit:

- Each of these techniques has known shortcomings, using multiple techniques together achieves some synergy between them.

Picture from: netflix.com

Google YouTube recommendation system

Why:

- Focus on videos, bring videos to users which they believe users will be interest in
- Increase the numbers of videos, increase the length of time, and maximize the enjoyment
- Ultimately google can increase revenue by showing more ads

Interesting things:

- Give up its old recommendation system based on random walk, changed to a new one based on Amazon's item-to-item collaborative filtering in 2010
- **Amazon's item-to-item collaborative filtering appears to be the best for video recommendation**

Picture from: youtube.com

References:

1. Linden, G.; Smith, B.; York, J.; , "Amazon.com recommendations: item-to-item collaborative filtering,". Internet Computing, IEEE , vol.7, no.1, pp. 76- 80, Jan/Feb 2003
2. Takács, G.; Pilászy, I.; Németh, B.; Tikk, D. (March 2009). "Scalable Collaborative Filtering Approaches for Large Recommender Systems". Journal of Machine Learning Research 10: 623-656
3. Candillier, L., Meyer, F., & Boull'e Marc. (2007). Comparing state-of-the-art collaborative filtering systems. Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany. 548-562. doi: 10.1007/978-3-540-73499-4_41
4. Ala Alluhaidan, Ala, "Recommender System Using Collaborative Filtering Algorithm" (2013). Technical Library. Paper 155. http://scholarworks.gvsu.edu/cistechlib/155
5. Francesco Ricci, Slides on "Item-to-Item Collaborative Filtering and Matrix Factorization". http://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/presentations/course_Ricci/13-Item-to-Item-Matrix-CF.pdf
6. Xavier Amatriain, Bamshad Mobasher; KDD 2014 Tutorial - the recommender problem revisited. http://www.kdd.org/kdd2014/tutorials.html

Thank You