Recommendation Systems

Why Recommender Systems

Recommendation Systems can help match users to items

They help item provider deliver their items to right user.

- Identify products most relevant to the user
- Personalized content
- Eg. Top n offers

Help website improve user engagement & increase online sales

2 Key approaches to RecSys

- Collaborative Filtering
- Content Based
- Popularity based

Market basket analysis

- ❑Discovers co-occurrence relationships among activities performed.
- ❑Market basket analysis can be used to **divide customers into groups.**
- ❑Market basket analysis may provide the retailer with information **to understand the purchase behaviour of a buyer**.
- ❑**“customers who bought book A also bought book B”**

For example:

- ❑When one super market chain discovered in its analysis that male customers that bought diapers often bought beer as well, have put the diapers close to beer coolers, and their sales increased dramatically.

- ❑Might tell a retailer that customers often purchase shampoo and conditioner together, so putting both items on promotion at the same time would not create a significant increase in revenue, while **a promotion involving just one of the items** would likely drive sales of the other.

Support and Confidence

The support s for a particular association rule A => B is the proportion of transactions in Dthat contain both A and B. That is,

$$\text{support} = P(A \cap B) = \frac{\text{number of transactions containing both } A \text{ and } B}{\text{total number of transactions}}.$$

The confidence c of the association rule A => B is a measure of the accuracy of the rule, as determined by the percentage of transactions in D containing A that also contain B. In other words,

$$\begin{aligned} \text{confidence} = P(B|A) &= \frac{P(A \cap B)}{P(A)} \\ &= \frac{\text{number of transactions containing both } A \text{ and } B}{\text{number of transactions containing } A} \end{aligned}$$

Popularity based recommender system

- ❑Recommend items viewed/purchased by most people .
- ❑Recommendations: **Ranked list of items by their purchase count / viewed count "Popular news".**
- ❑It uses:
  - ❑text only
  - ❑Purchase history
  - ❑User and item features
  - ❑Scale

The problems with popularity based recommendation system is that the personalization is not available with this method i.e. even though you know the behaviour of the user, you cannot recommend items accordingly.

For example: for a song recommendation if the model is popularity based recommender, it is not personalised towards any user and will output the same list of recommended songs.

Intuition behind CF

	Finding Nemo	Toy Story	Exorcist	Conjuring	The Terminal	Forrest Gump
User 1	5	4		2		
User 2			4	4		3
User 3		2			5	4
User 4	4	5	3			
User 5			5	5	3	
User 6	4	4		2		

"Finding Nemo" and "Toy Story" seem to be related since users who have rated both these movies have rated them highly

How do we compute Similarity – 1/2

Pearson Similarity

- Similarity is the Pearson coefficient between the two vectors

Cosine Similarity

- Similarity is the cosine of the angle between the 2 item vectors

Cosine Similarity

$$CosSim(x,y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} = \frac{\langle x, y\rangle}{||x||\ ||y||}$$

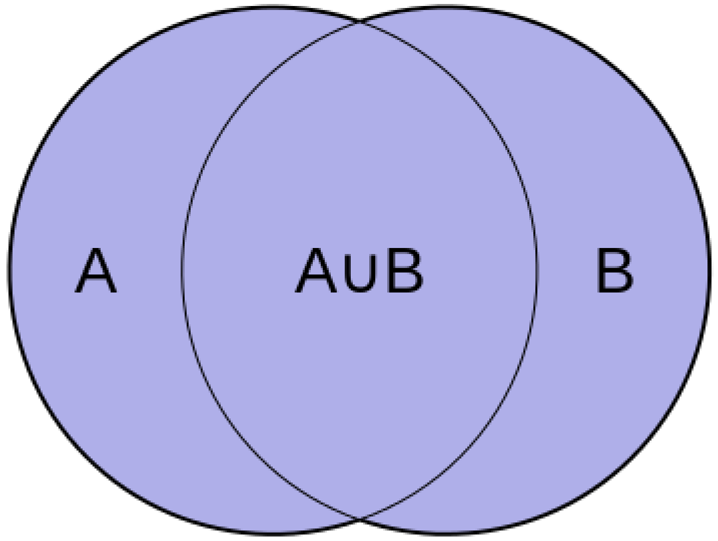
Pearson Similarity

$$\begin{aligned} Corr(x,y) &= \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2}\sqrt{\sum (y_i - \bar{y})^2}} \\ &= \frac{\langle x - \bar{x},\ y - \bar{y}\rangle}{||x - \bar{x}||\ ||y - \bar{y}||} \\ &= CosSim(x - \bar{x}, y - \bar{y}) \end{aligned}$$

Correlation is the cosine similarity between centered versions of x and y

How do we compute Similarity – 2/2

Jaccard Similarity is useful when the User/Item Matrix contain binary values (e.g. did the user watch the movie or not)



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

Types of CF

Item based CF

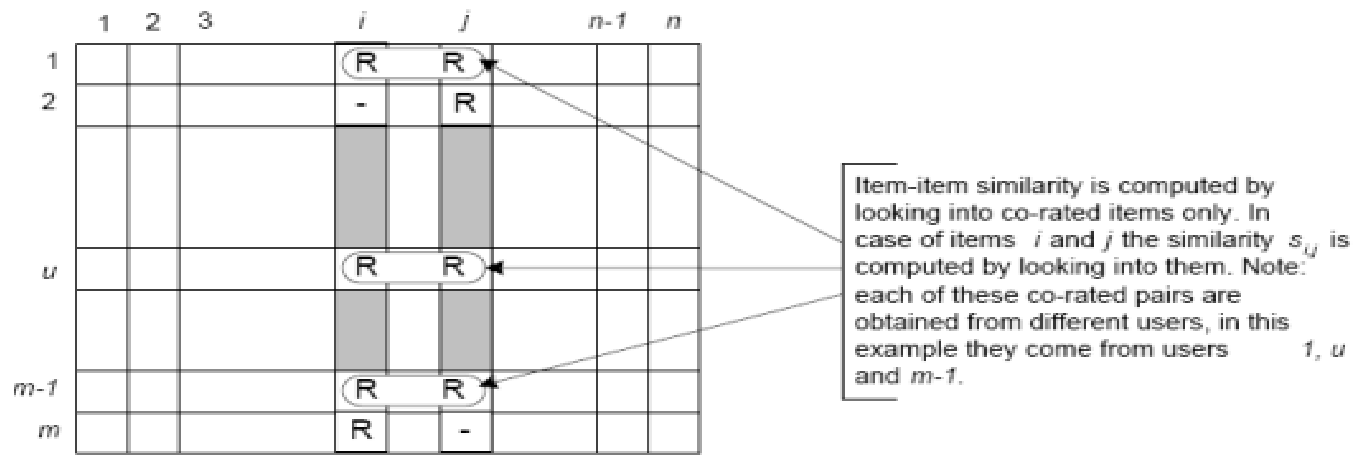
- Compute Similarity between Items

User Based CF

- Compute Similarity between Users

Item based CF

Recommend items that are similar to the items the user has already purchased (or viewed/interacted with)


1
2
3
i
j
n-1
n
1
2
u
m-1
m
R
-
Item-item similarity is computed by looking into co-rated items only. In case of items i and j the similarity $s_{i,j}$ is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users 1, u and m-1.

Similarity is computed between any 2 pairs of items by looking into co-rated items only (using any of the similarity measures described before)

Item based CF

How to generate recommendations using IBCF

IBCF Formula

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(j)} \text{sim}(i,j)}$$

Explanation: To determine the preference of a given user, say u_1, to a given item say i_12, we perform the following steps:

- From the Item-Item Similarity Matrix, identify the Top K most similar items to item, i_12
- Let's assume the items are (with K=5) : i_4, i_13, i_21, i_32, i_43
- From the list of similar items, find the items already rated by the user u_1 – Say, the user has rated i_4 and i_32 as 4 and 5 respectively
- Similarity between i_1 and i_4 and i_1 and i_32 is 0.7 and 0.9 respectively
- The estimated rating that u_1 will give to i_12 is then

  (4*0.7 + 5*0.9)/(0.7 + 0.9) = 4.56

User based CF

Instead of finding similarity between items (columns, in the User-Item matrix), similarity between Users (rows) is considered

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

Explanation: To determine the preference of a given user, say u_1, to a given item say i_12, we perform the following steps:

- From the User-User Similarity Matrix, identify the Top K most similar items to User, u_1
- Let's assume the users are (with K=5) : u_10, u_22, u_67, u_72, u_89
- From the list of similar users, find the users who have rated the item i_12 – Say, the users u_10 & u_72 have rated i_12 (rating 4 & 3 respectively). Similarity between u_1 and u_10 and u_1 and u_72 is 0.7 and 0.9 respectively
- The estimated rating that u_1 will give to i_12 is then
  (4*0.7 + 3*0.9)/(0.7 + 0.9) = 3.43

IBCF vs UBCF

IBCF is more efficient than UBCF

- Typical applications involve far more Users than Items. Hence Similarity matrix for IBCF is more compact than UBCF
- Similarity estimates between items is also more likely to converge over time than similarity between users. Hence the similarities can be pre-computed and cached unlike similarity between users that need to be dynamically computed
- **However**, the IBCF recommendations tend to be more *conservative* than UBCF

Mean centering the ratings

Ratings can be centered around the mean to address individual biases in rating

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

MATRIX FACTORIZATION BASED APPROACH

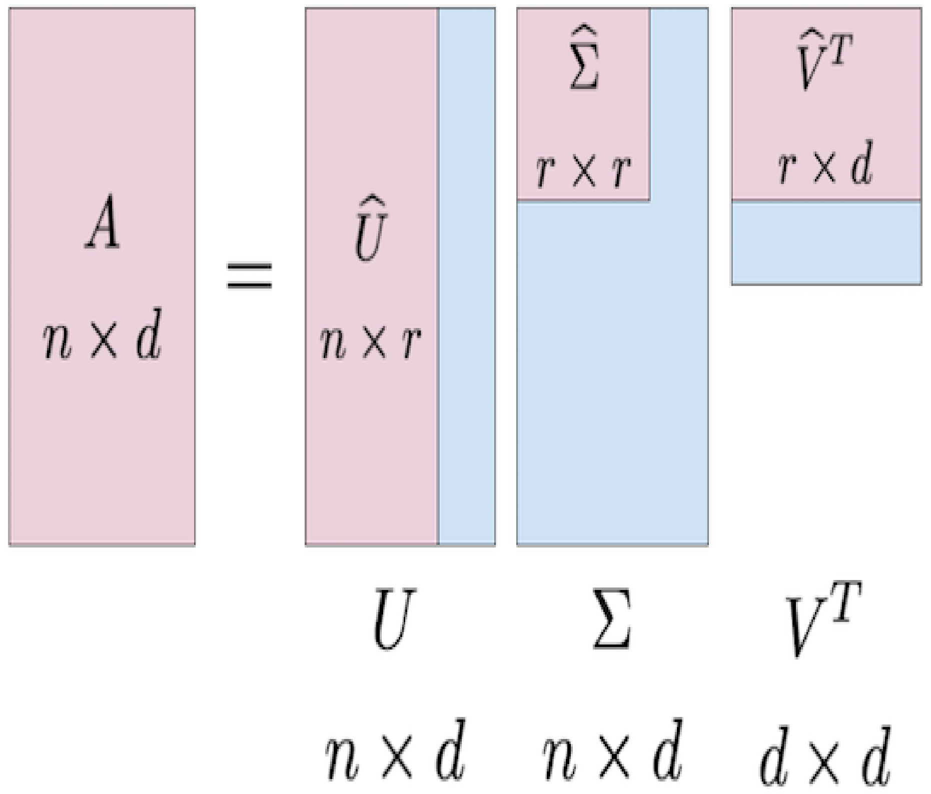
What is Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) can be used to decompose any given Matrix, M into a product of 3 matrices as follows:

$$M = U \times \Sigma \times V^T$$

- where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*

The matrix can be approximated to a lower rank by considering only the largest singular values



SVD for User Item Matrix

SVD on the User Item Matrix can help generate 2 lower rank matrices – A *User-Feature Matrix* and an *Item Feature Matrix*

Intuitively, the features may represent some underlying themes/latent factors. Eg in the case of movies the features may broadly represent different genres, high budget/low budget movies, commercial/art films etc

Funk SVD approach

User Item matrix is highly sparse, hence traditional SVD algorithms cannot be used to factorize the matrix

Hence, User features and Item features are determined by optimizing the following cost function:

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

Where,

- p_u and q_i are the features vectors for User u and item i respectively
- lambda is the regularization parameter

The cost function can be optimized using Gradient Descent

Generating Predictions using SVD

The vector pu represents the affinity of user u for each of the latent factors

Similarly the vector qi represents affinity of the item i for the latent factors

Hence, to estimate the preference of user u towards rating I we can simply multiply the feature vectors together

$$r_{ui} = p_u \cdot q_i = \sum_{f \in \text{latent factors}} \text{affinity of } u \text{ for } f \times \text{affinity of } i \text{ for } f$$

To estimate the rating for each pair of User/Item we can simply take the dot product of *User-Feature* and *Feature-Item* Matrix

Content based Recommendation

NLP Basics

Bag of Word (BoW) approach

Common text processing techniques

- Stop word removal
- Lemmatization/Stemming
- N gram tokenization
- TF-IDF Weighing

Word Embedding models

Bag of Words (BoW) vectorization

Text data cannot be directly used for Modelling

One approach to extract features involves the Bag of Words (BoW) approach

The BoW approach generates a Document-Term Matrix (DTM) where columns represents all the unique words in the corpus and the cell value indicates the frequency of that word in a given document (represented by rows)

Raw Text	movie	great	loved	amazing	action	scenes	boring	hated	bored	really	acting	plot
This movie is great	1	1										
loved the movie	1		1									
Amazing action scenes. Great movie				1	1	1						
movie is boring	1						1					
boring movie. Hated it	1						1	1				
loved the acting and the plot			1								1	1
got really bored with the movie	1								1	1		

Enhancements to simple BOW approach

The Document Term Matrix can be further enhanced by:

- Removing stop words (such as “the”, “and” etc..)
- Removing very sparse or very common words from the corpus
- Reducing variations in word forms using Lemmatization or Stemming (e.g. *boring* and *bored* show up as 2 separate columns in the example before)
- Including n-grams to capture common phrases (e.g *Machine Learning* is a single phrase)
- *TF-IDF* to capture relative importance of words in a document

TF-IDF

TF-IDF is composed of 2 terms:

- Term Frequency (TF) measures how frequently a term appears in the document
- Inverse Document Frequency (IDF) measures the relative importance of words in a corpus. Hence commonly appearing words would have a low IDF score whereas rarer words (which are potentially more informative) would have a higher IDF score

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

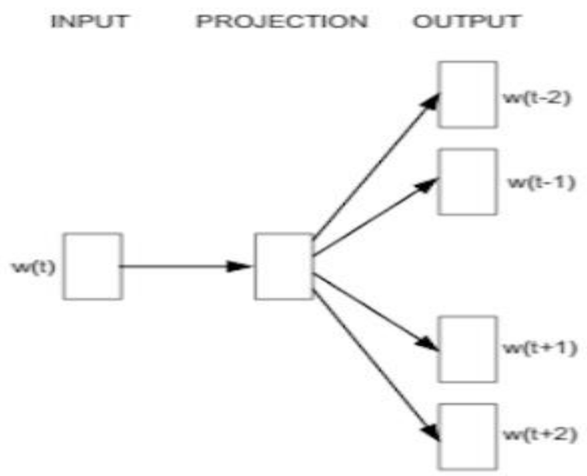
N = total number of documents

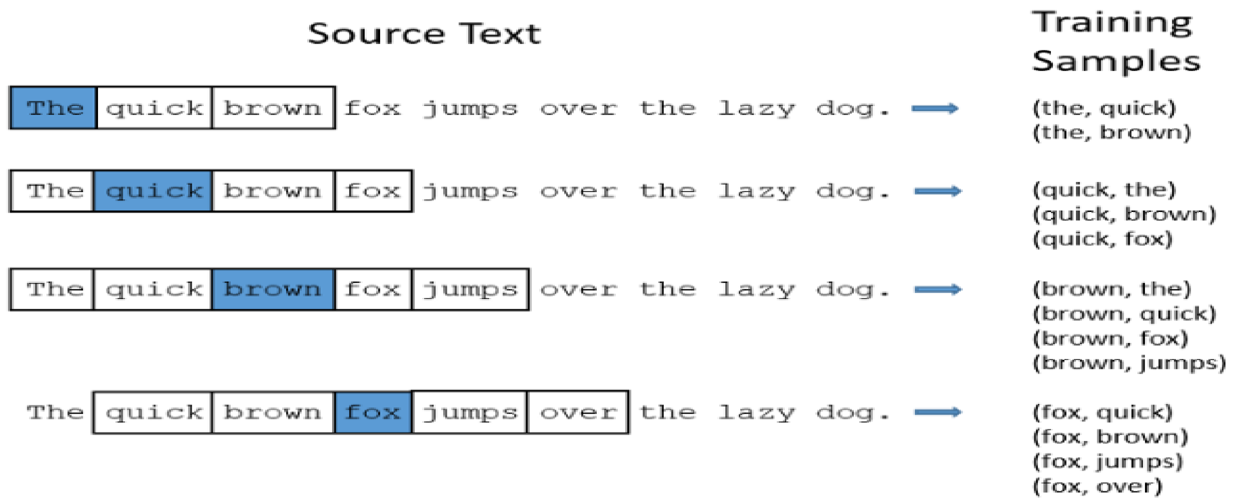
Word Embedding Models (Word2Vec)

Word vectors are trained by setting up a *fake* training task

For e.g. in the Skip gram architecture, the model tried to predict the neighboring/surrounding words given the target word

The weights of the hidden layer learnt during the training process provides the feature vector for each word – Words that appear in similar context ends up having similar feature vectors





Methods of Hybridization

Some of the typical methods of Hybridization include

- **Weighted**–Recommendations from each system is weighted to calculate final recommendation
- **Switching**–System switches between different recommendation model
- **Mixed**-Recommendations from different recommenders are presented together.

A common approach is to use **Latent Factor models for high level recommendation** and then **improving them using content based systems** by using information on users or items

Hybrid recommender systems

Multiple recommender systems are combined to improve recommendations.

❑Although any type of recommender systems can be combined a common approach in industry is to combine **content based approaches and collaborative filtering approaches**.

❑Content based models can be used to solve the **Cold start and Grey Sheep problems in Collaborative filtering.**

THANKS!