DATA MINING

Recommender Systems

Ashish Pujari

Lecture Outline

- Introduction
  - Recommender Systems
  - Recommender System Types
- Memory based CF
  - User Based
  - Item Based
- Model based CF
  - Matrix Factorization
  - Deep Learning
- Hybrid Recommenders

RECOMMENDER SYSTEMS

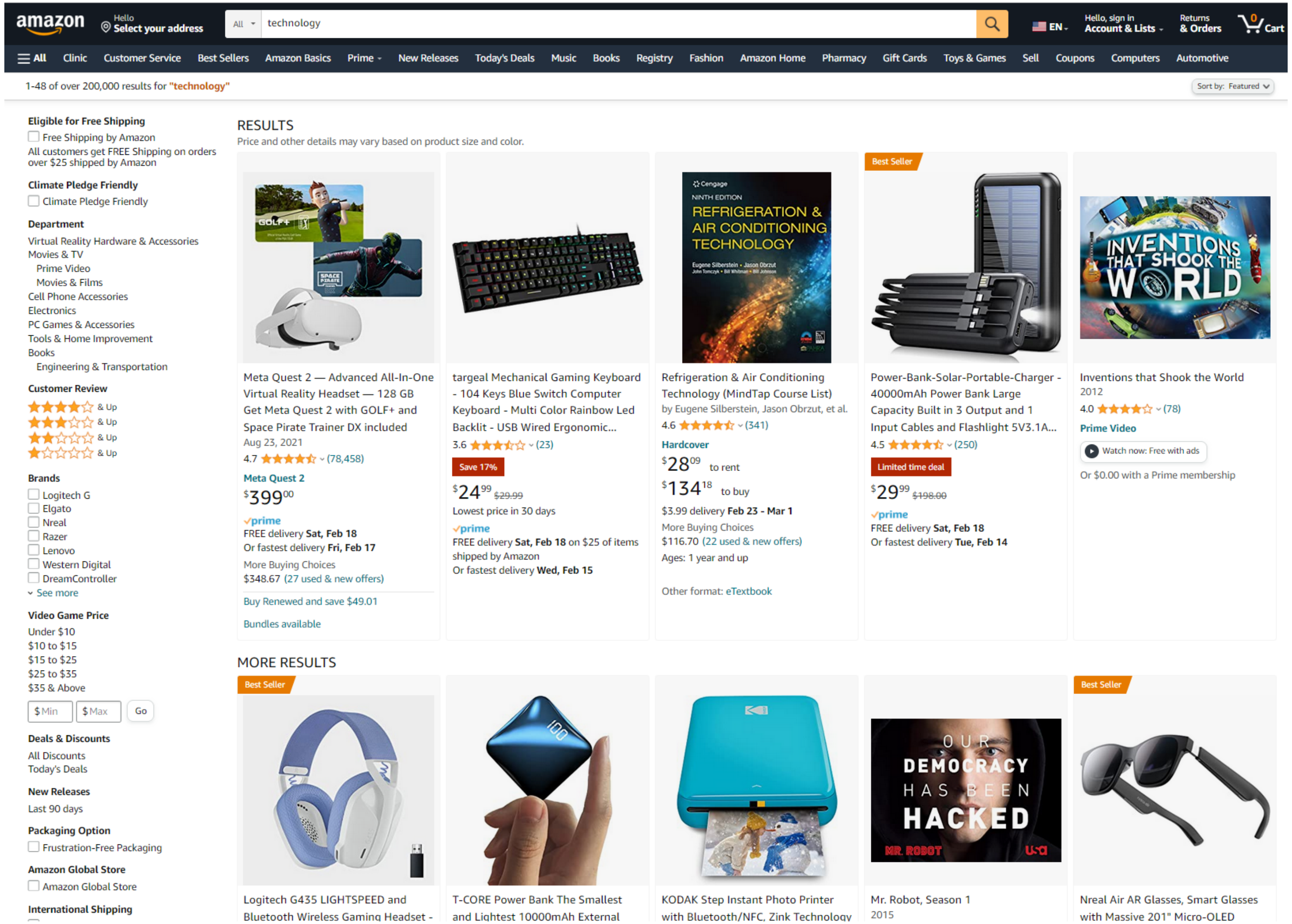
Introduction

Recommender Systems

- “Recommender Systems aim to help a user or a group of users to select items from a crowded item or information space. ” (MacNee et. al 2006)
- Functions
  - Increase the number of items sold
  - Sell more diverse items
  - Increase the user satisfaction
  - Increase user fidelity
  - Better understanding of what the user wants

Amazon

- 12 million items across all categories
- 350 million items on Amazon Marketplace
- Recommendation Engine

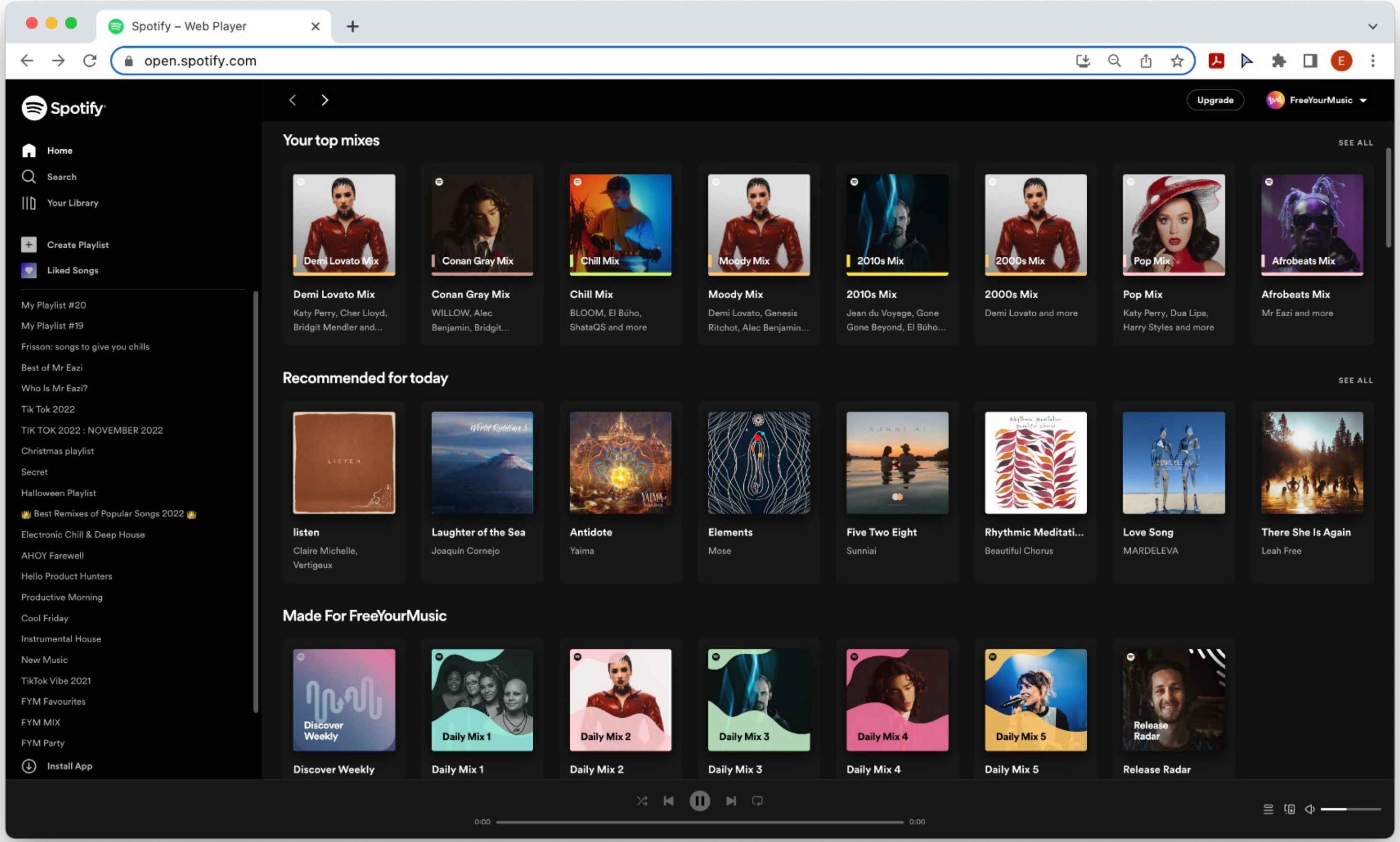


Netflix

- 223 Million Paid Subscribers
- 3,600+ movies,1,800+ TV Shows
- Recommendation Engine

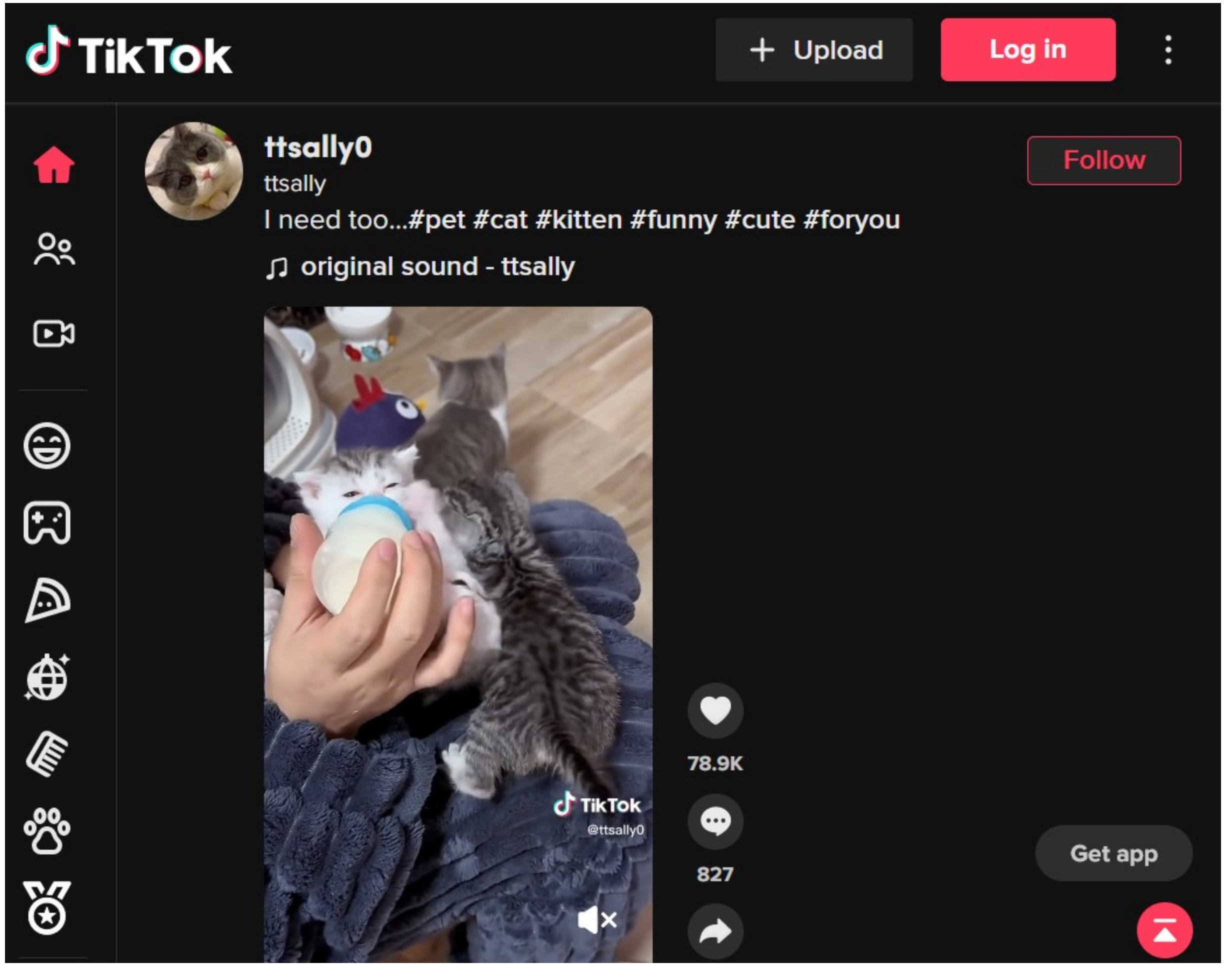
Spotify

- 188 million subscribers
- 100 million songs
- 5 million podcasts
- Recommendation Engine

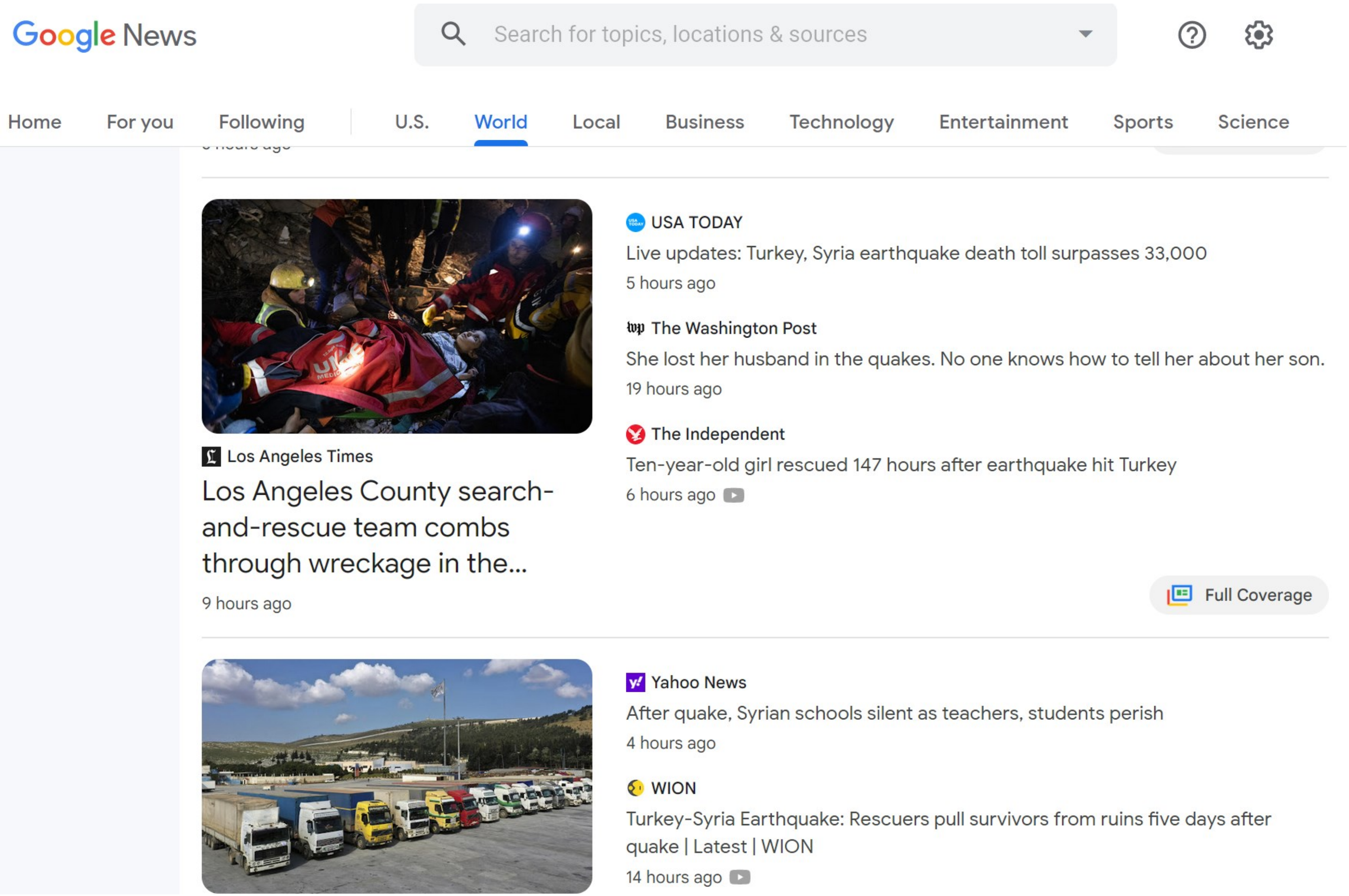


Tiktok

- Over 1.53 billion users
- 1 billion monthly active users
- Recommendation Engine

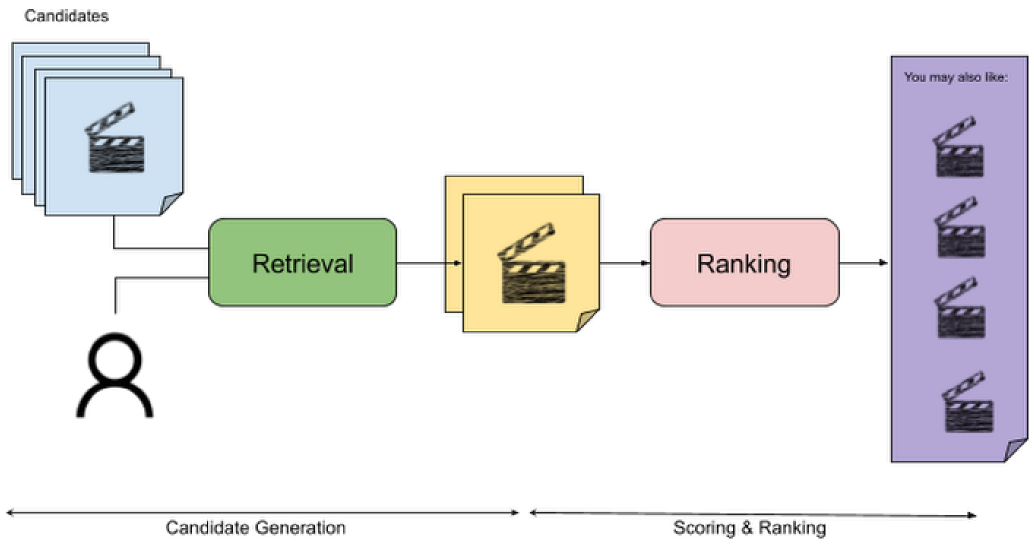


Google News


Google News
Search for topics, locations & sources
Home
For you
Following
U.S.
World
Local
Business
Technology
Entertainment
Sports
Science
Los Angeles Times
Los Angeles County search-and-rescue team combs through wreckage in the...
9 hours ago
USA TODAY
Live updates: Turkey, Syria earthquake death toll surpasses 33,000
5 hours ago
The Washington Post
She lost her husband in the quakes. No one knows how to tell her about her son.
19 hours ago
The Independent
Ten-year-old girl rescued 147 hours after earthquake hit Turkey
6 hours ago
Full Coverage
Yahoo News
After quake, Syrian schools silent as teachers, students perish
4 hours ago
WION
Turkey-Syria Earthquake: Rescuers pull survivors from ruins five days after quake | Latest | WION
14 hours ago

Recommendation Generation Steps

1. Candidate generation and retrieval
   - Generates a smaller subset of plausible candidates from a large collection/corpus
2. Scoring and ranking
   - Scores the candidates and provides an ordered list of items to users
   - Ranking function is learned from the labeled dataset to optimize the global performance
3. Re-ranking recommendations
   - Re-ranks candidates based on user feedback



Source: Google Cloud

Evaluation Metrics

- Prediction Accuracy
  - The ability of a CF system to predict a user's rating for an item
- Rank accuracy
  - Precision – percentage of items in a recommendation list that the user would rate as useful
- Novelty
  - Ability to recommend items that the user was not already aware of.
- Serendipity
  - Users are given recommendations for items that they would not have seen given their existing channels of discovery
- Coverage
  - The percentage of the items known to the CF system for which the CF system can generate predictions.
- Learning Rate
  - How quickly the CF system becomes an effective predictor of taste as data begins to arrive.
- Confidence
  - Ability to evaluate the likely quality of its predictions.
- User Satisfaction
  - By surveying the users or measuring retention and use statistics

Explicit Ratings

- Explicit ratings of customers to products (user to item).
- Some cells are not loaded as they are products that are not rated by any user
- Could be challenging to collect data

User	Item1	Item2	Item3	Item4	Item5
User 1	4	3	2	2	?
User 2	-	2	3	-	1
User 3	2	1	2	5	5
User 4	-	-	4	2	2
User 5	1	2	5	1	1

Problem1: Rating prediction

- Problem:
  - Predict the rating that a user u will give his or her unrated item i.
- Approach
  - When ratings are available, this task is a regression or a multi-class classification problem
  - Goal is to learn a function f that predicts the rating $f(u,i)$ of a user u for a new item i.
  - Ratings $\mathcal{R}$ are divided into a training set $\mathcal{R}_{Train}$ and a test set $\mathcal{R}_{Test}$
  - Prediction accuracy:

$$MAE(f) = \frac{1}{|\mathcal{R}_{Test}|} \sum_{r_{ui} \in \mathcal{R}_{Test}} |f(u,i) - r_{ui}|$$

$$RMSE(f) = \sqrt{\frac{1}{|\mathcal{R}_{Test}|} \sum_{r_{ui} \in \mathcal{R}_{Test}} (f(u,i) - r_{ui})^2}$$

Implicit Ratings

- Implicit ratings are not as obvious in terms of preference
- Observations of user behavior e.g., clicks, views, accessed, or purchased
- Can be collected with little or no cost to user
- Ratings inference may be imprecise

User	Item1	Item2	Item3	Item4	Item5
User 1	1	1	-	-	?
User 2	-	-	1	1	1
User 3	1	-	1	-	1
User 4	-	-	-	1	1
User 5	1	1	1	1	-

Problem2: Ratings Not Available

- Problem:
  - Ratings are unavailable, but list of items purchased by a user is known - the problem transforms into the task of recommending N items $L(u_{\mathrm{a}})$ to the user
- Approach
  - Split the items of I into a set I_{Train}, used to learn L, and a test set I_{Test}.
  - Let $T(u)$ be the subset of test items that a user u found relevant (purchased/accessed/liked)
  - Performance :

$$Precision(L) = \frac{1}{|U|}\sum_{u \in U} |L(u) \cap T(u)|/|L(u)| = \frac{Correctly\ Recommended\ Items}{Total\ recomended\ Items}$$

$$Recall(L) = \frac{1}{|U|}\sum_{u \in U} |L(u) \cap T(u)|/|T(u)| = \frac{Correctly\ Recommended\ Items}{Relevant\ Items}$$

  - A drawback of is that all items of $L(u_{\mathrm{a}})$ are considered equally interesting to user u.

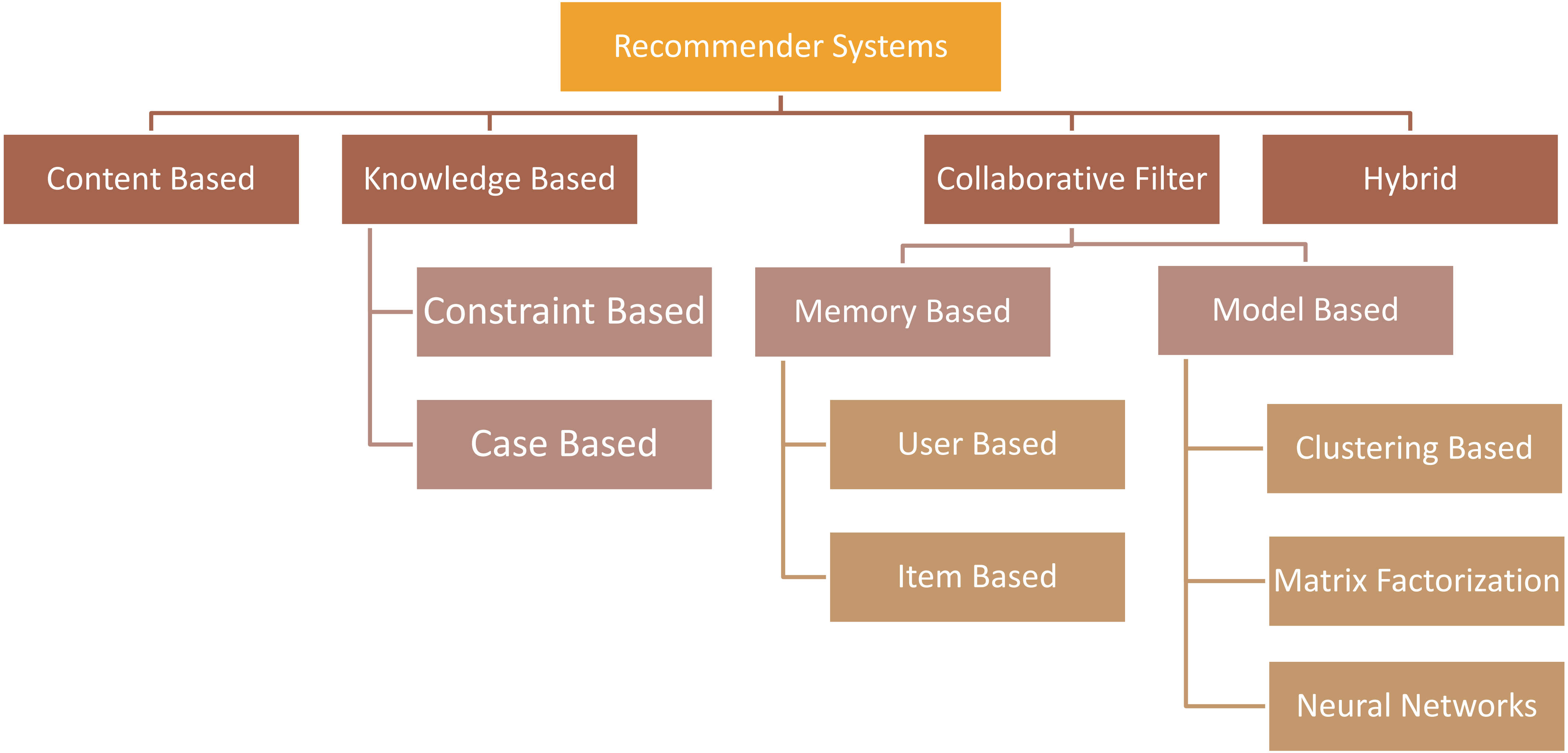
Problem2: Ratings Not Available

- Alternative setting:
  - Learning a function L that maps each user u to a list $L(u_{\mathrm{a}})$ where items are ordered by their "interestingness" to u.
  - If the test set is built by randomly selecting, for each user u, a single item i_u, the performance of L can be evaluated with the Average Reciprocal Hit-Rank (ARHR):

$$ARHR(L) = \frac{1}{|U|} \sum_{u \in \mathrm{U}} \frac{1}{Rank\big(i_u, L(u)\big)}$$

  - where $Rank\big(i_u, L(u)\big)$ is the rank of item i_u in $L(u)$ equal to ∞ if $i_u \notin L(u)$.

Recommender Systems: Types



Content Based Recommender Systems

- The likeness of items is determined based on traits associated with the compared items
- Treat suggestions as a user-specific category problem and learn a classifier for the customer's preferences depending on product traits.

Attribute	Type	Example
Genre	Multi-valued qualitative	Drama
Country	Nominal single valued qualitative	EUA
Director	Nominal single valued qualitative	Steven Spielberg
Cast	Multi-valued qualitative	Tom Hanks, David Morse, Bonnie Hunt, Michael Clarke
Year	Ordinal single valued qualitative	1999
Description	Textual	The USA government offers one million dollars for some informa-

These are *attributes*

Attribute	Poland Spring 12-Pack	Cherry Coke Zero Single Serve Bottle	Cherry Coke Zero 12-Pack Cans
Manufacturer	Nestle	Coca Cola	Coca Cola
Brand	Poland Spring	Coke Zero	Coke Zero
Sub-Brand	Not Applicable	Cherry Coke Zero	Cherry Coke Zero
Category	Water	Carbonated Bev	Carbonated Bev
Type	Still Water	Diet	Diet
Sweetener	Not Applicable	Sucralose	Sucralose
Container	Bottle	Bottle	Can
Multi-Pack	12-Pack	Not Applicable	12-pack
Size	190.8 ounces	20 ounces	144 ounces

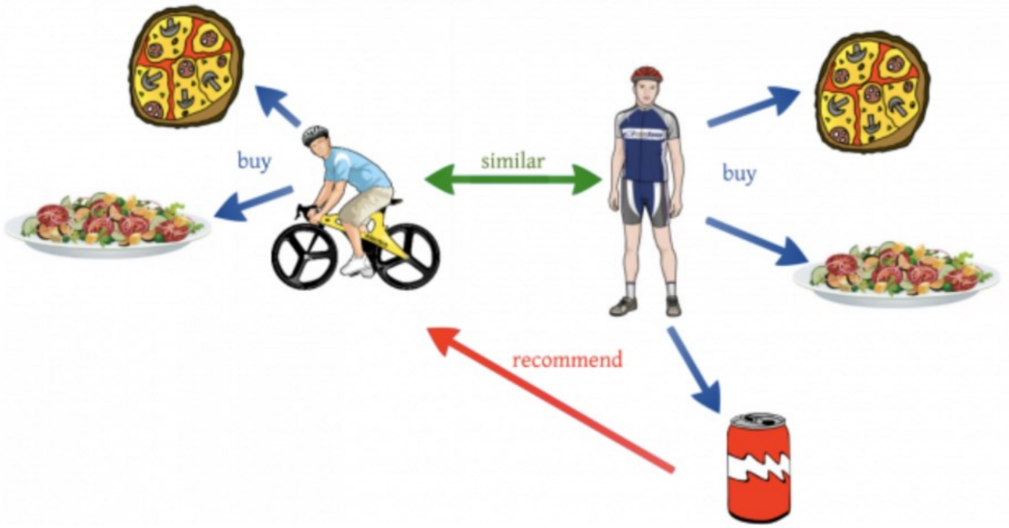
These are the *values* of the attributes

Knowledge Based Recommender Systems

- Recommend items based on domain knowledge about how certain item features fulfill users’ needs and preferences
- Similarity function considers problem description (needs) and solution of the problem (recommendation) and estimates how much the user needs them
- Constraint-based
  - based on explicitly defined set of recommendation rules
  - fulfill recommendation rules
- Case-based
  - based on different types of similarity measures
  - retrieve items that are similar to specified requirements

Collaborative Filtering (CF)

- Problem
  - “What items do users with interests similar to yours like?”
- Approach
  - Use the "wisdom of the crowd" to recommend items
  - Most effective when there is a rich history of user preferences or behavior.
  - No specific attributes are required for the content which CF can infer.

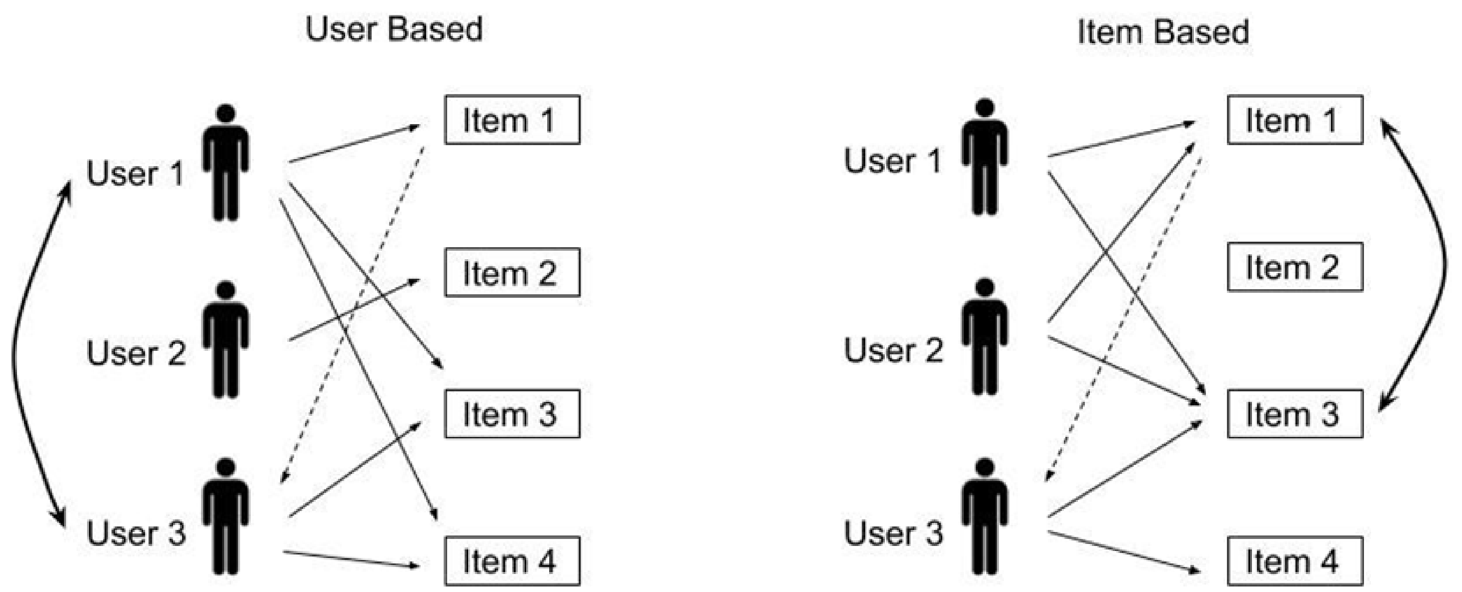


MEMORY BASED COLLABORATIVE FILTERS

User Based, Item Based

Memory Based CF

- Uses the database of user ratings to compute the similarity between users or items
- It involves 3 steps
  1. Similarity computation
  2. Neighborhood selection
  3. Ratings prediction



Similarity Computation

- Pearson’s correlation
- Spearman rank-order correlation
- Cosine similarity
- Adjusted cosine similarity
- Frequency-Weighted Pearson Correlation
- Adjusted mutual information
- Adjusted Rand index
- Jaccard index
- Euclidean distance

https://www.sciencedirect.com/science/article/pii/S1319157821002652

Neighborhood Selection

- Top k filtering
  - Selects the k-nearest neighbors (users or items) in terms of similarity
  - To avoid problems with efficiency or accuracy, k should be chosen carefully
- Threshold filtering
  - Defines a threshold and keeps only the users whose similarities, exceed the threshold w_{min}
  - More flexible but the right value of w_{min} might be hard to determine
- Negative filtering
  - Negative rating correlations are less reliable than positive ones.
- Hybrid Approach
  - Above approaches are not exclusive and may be combined to meet user requirements

User-Based Collaborative Filtering (UB-CF)

- Approach
  - Find similar users (or similar items) based on similarity metrics and take weighted average of ratings
  - Creates a $User\ x\ User$ matrix to calculate similarities between users
- Assumptions
  - If users had similar tastes in the past, they would continue to have similar tastes in the future
  - User preferences remain stable and consistent over time
- Algorithm
  1. Select like-minded peer group (k Nearest-Neighbor) for a target user
  2. Choose candidate items not in the target user's list but in the list of the peer group
  3. Produce a weighted score and predict the ratings for the given items
  4. Select the best candidate items and recommend them to the target user
  5. Redo all Steps 1-4 on a timely basis

UB-CF: Pearson's Similarity

- Pearson's correlation measures the linear correlation between two random variables, X and Y.
- It is a normalized version of the covariance:

$$\rho = \frac{\text{cov}(X,Y)}{\sqrt{\text{var}(X)var(Y)}} = \frac{E\big((X - E(X))\big)\big(X - E(Y)\big)}{\sqrt{\text{var}(X)var(Y)}} = \frac{E((X - \mu_X)(Y - \mu_Y)}{\sigma_X \sigma_Y}$$

- Pearson's similarity :

$$sim(a,b) = \frac{\sum_{p \in product(P)} (R_{ap} - \bar{R}_a)(R_{bp} - \bar{R}_b)}{\sqrt{\sum_{p \in product(P)} (R_{ap} - \bar{R}_a)^2} \sqrt{\sum_{p \in product(P)} (R_{bp} - \bar{R}_a)^2}}$$

UB-CF: Ratings Prediction

- Rating prediction :

$$pred(\mathrm{a,p}) = \frac{\sum_{b \in neighbors(n)} sim(a,b).(R_{bp})}{\sum_{b \in neighbors(n)} sim(a,b)}$$

- Mean-centered aggregation :

$$pred(\mathrm{a,p}) = \bar{R}_a + \frac{\sum_{b \in neighbors(n)} sim(a,b).(R_{bp} - \bar{R}_b)}{\sum_{b \in neighbors(n)} sim(a,b)}$$

  - Where, $\bar{R}_a$ and $\bar{R}_b$ are the average ratings for users a and b, respectively
  - $sim(a,b)$ is the similarity between users $a\ and\ b$

UB-CF: Example

User	Item1	Item2	Item3	Item4	Item5
Alice	4	3	2	2	?
Bob	4	2	3	3	1
Chaya	2	1	2	5	5
Dan	5	5	4	2	2
Ethan	1	2	5	1	1
Frank	2	4	1	4	2
Ganesh	3	1	3	3	5

Determine whether Alice will like or dislike Item5, which Alice has not yet rated or see

Item-based Collaborative Filtering (IB-CF)

- Approach
  - Recommend items which are the most similar to a set of items already rated with a high score by the active user
  - Creates a $Item\ x\ Item$ matrix to calculate similarities between items
- Algorithm
  1. Find similar items that the target user has already viewed or rated
  2. Forecast the ratings for unrated items.
  3. If the forecasts are higher than the threshold, then suggest items to the target user.

IB-CF: Cosine Similarity

- Cosine similarity is the cosine of the angle between two vectors, and it is used as a distance evaluation metric between two points in the plane

$$cos\ (\boldsymbol{x}_a, \boldsymbol{x}_b) = \frac{\boldsymbol{x}_a \cdot \boldsymbol{x}_b}{\|\boldsymbol{x}_a\| \|\boldsymbol{x}_b\|}$$

- Adjusted Cosine Similarity

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (R_{ui} - \bar{R}_u)(R_{uj} - \bar{R}_u)}{\sqrt{\sum_{u \in U_{ij}} (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{u \in U_{ij}} (R_{uj} - \bar{R}_u)^2}}$$

  - Where where U_i and U_j represents the set of users who rated items i and j, respectively
  - U_{ij} represents the set of users who rated both items i and j.
  - $\bar{R}_u$ denotes the average ratings by u.
  - R_{ui} and R_{uj} are the ratings of user u on items i and j, respectively.

IB-CF: Ratings Prediction

- Prediction
  - For a user u and item i is a weighted sum of the user $u's$ ratings for items most similar to i

$$pred(u, \mathrm{i}) = \frac{\sum_{j \in RatedItems(n)} sim(i, j) \cdot R_{uj}}{\sum_{j \in RatedItems(n)} sim(i, j)}$$

- Mean-centered aggregation :

$$pred(u, \mathrm{i}) = \bar{R}_i + \frac{\sum_{j \in RatedItems(n)} sim(i, j) \cdot (R_{uj} - \bar{R}_j)}{\sum_{j \in RatedItems(n)} sim(i, j)}$$

  - Where, $\bar{R}_i$ and $\bar{R}_j$ are the average ratings for items i and j, respectively
  - $sim(i, j)$ is the similarity between items i and j

Memory Based Methods

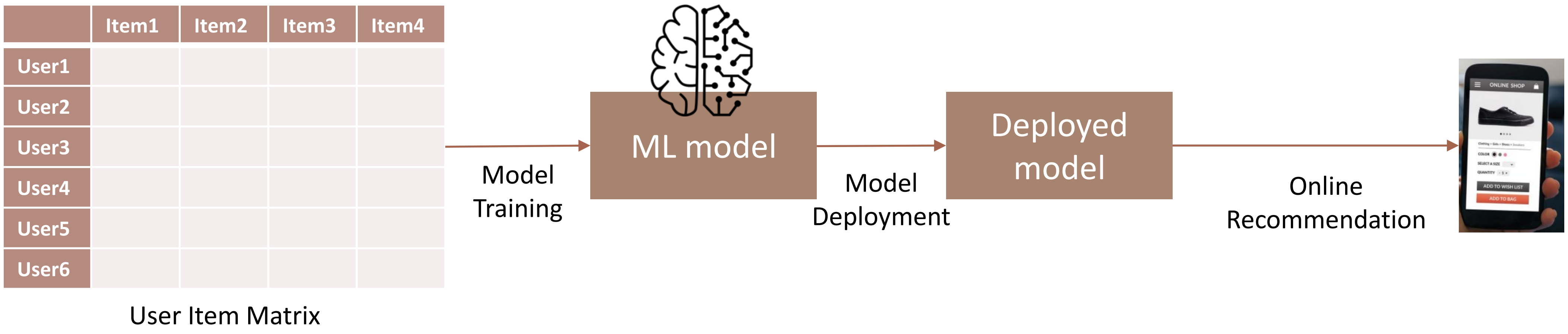
	User Based CF	Item Based CF
Approach	• Makes ratings predictions based on similar users to the target user	• Makes ratings predictions based on similar items rated by the target user
Pros	• Context independent. • Relatively simple algorithm to implement • More accurate compared to other content-based methods	• More computationally efficient than user-based nearest neighbors; directly pre-calculates similarity between the co-rated items, skipping K-neighborhood search • Compared with user-based approach that is affected by the small change of users' ratings, item-based approach is more stable.
Cons	• Sparsity: The percentage of people who rate items is really low. • Scalability: Greater cost of finding K nearest neighbors when number of users is large. • Cold-start: New users will have little information about them to be compared with other users.	

MODEL BASED COLLABORATIVE FILTERS

Matrix Factorization, Neural Networks

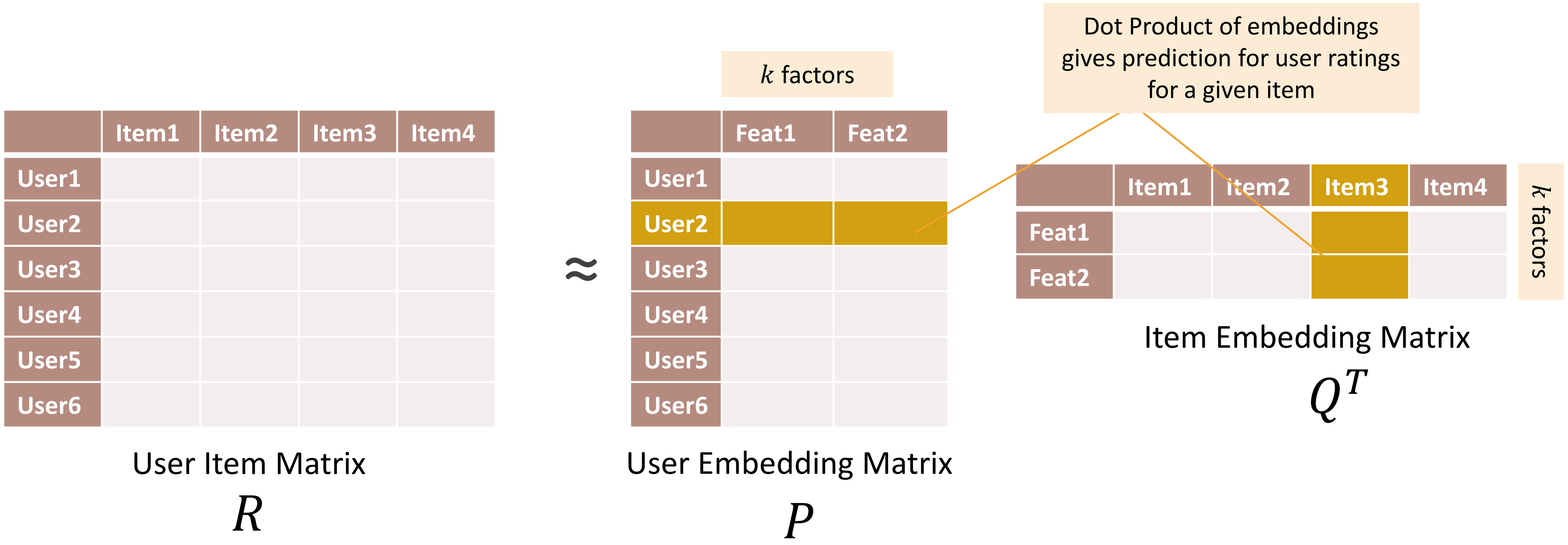
Model Based CF

- Model-based recommendation systems involve building machine learning models based on the dataset of user-item ratings
- Approach
  - Models can be trained offline or online
  - At run-time, only the learned model is used to make predictions
  - Models are updated / re-trained periodically



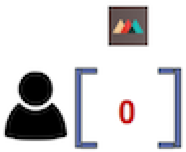
Matrix Factorization

- Factorization increases the information density of the sparse User-Item matrix
- Each user and item can be described by embeddings (a set of k features)



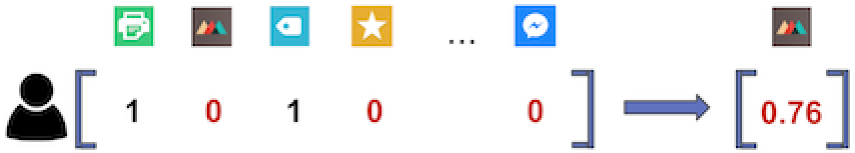
User and Item Embeddings

- A recommendation is possible if we express each user as a vector of their taste values, and at the same time express each item as a vector of what tastes they represent.



User-item interaction

User preference vector



User preference (left) and predicted relevance (right)

Matrix Factorization

- We can use low rank matrix factorization to represent each user and item by k-dimensional vectors
- For, entire matrix

$$R = PQ^T$$

- For, Individual Ratings

$$\hat{r}_{u,i} = q_i^T \mathrm{p}_u$$

- Loss Function

$$L = \sum_{u,i \in \kappa} \left(r_{u,i} - q_i^T \mathrm{p}_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

  - Where, R is the full user-item rating matrix
  - P is a matrix that contains the users and the k factors represented as $(user, factor)$
  - Q^T is a matrix that contains the items and the k factors
  - κ is the set of (u, i) pairs for which is known.
  - $\hat{r}_{u,i}$ represents our prediction for the true rating
  - L2 regularization terms to prevent overfitting of the user and item vectors

Alternating Least Squares (ALS)

- Problem
  - SVD for matrix factorization requires inverting a potentially very large matrix and could be computationally expensive
- ALS
  - Iterative optimization process where we for every iteration try to arrive closer and closer to a factorized representation of our original data.
  - Helps solve the overfitting issue in sparse data.
  - In each iteration, the algorithm alternatively fixes one factor matrix and solves for the other to minimize the weighted squared error until convergence.

ALS: Algorithm

- Initialize the P and Q matrices with random values
- ALS alternates between holding the q_i's constant and the p_u's constant
- While all q_i's are held constant, each p_u is computed by solving the least squares problem

$$p_u = \left(\sum_{r_{u,i} \in r_{u*}} q_i {q_i}^T + \lambda I_k \right)^{-1} \sum_{r_{u,i} \in r_{u*}} r_{u,i} q_i$$

- Then the p_u's are held constant while the q_i's are altered to solve the least squares problem, again, each independently.

$$q_i = \left(\sum_{r_{u,i} \in r_{u*}} p_u {p_u}^T + \lambda I_k \right)^{-1} \sum_{r_{u,i} \in r_{u*}} r_{u,i} p_u$$

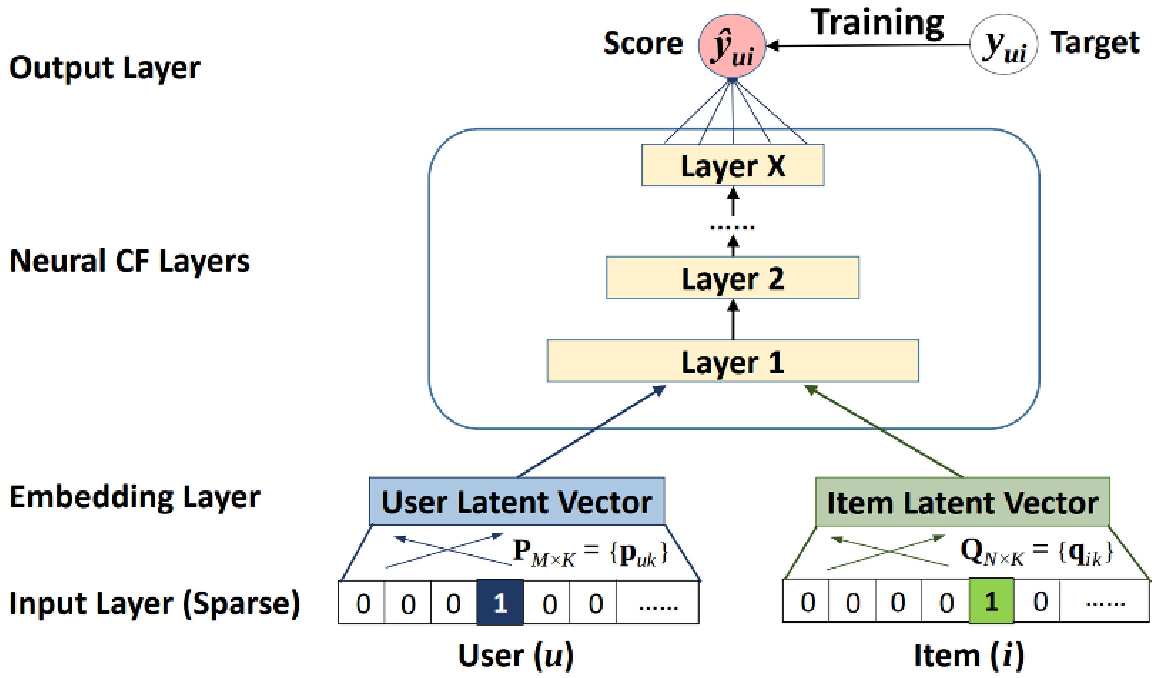
- Above two steps are iterated until convergence OR some stopping criterion is reached

Stochastic SVD

- ALS gives us an easy way to factorize the matrix of user preferences in MapReduce. However, this technique has the disadvantage of requiring several iterations, sometimes taking a long time to converge to a quality solution.
- Stochastic SVD a recently developed alternative method which approximates the well-known Singular Value Decomposition of a large matrix, and which admits a non iterative MapReduce implementation.

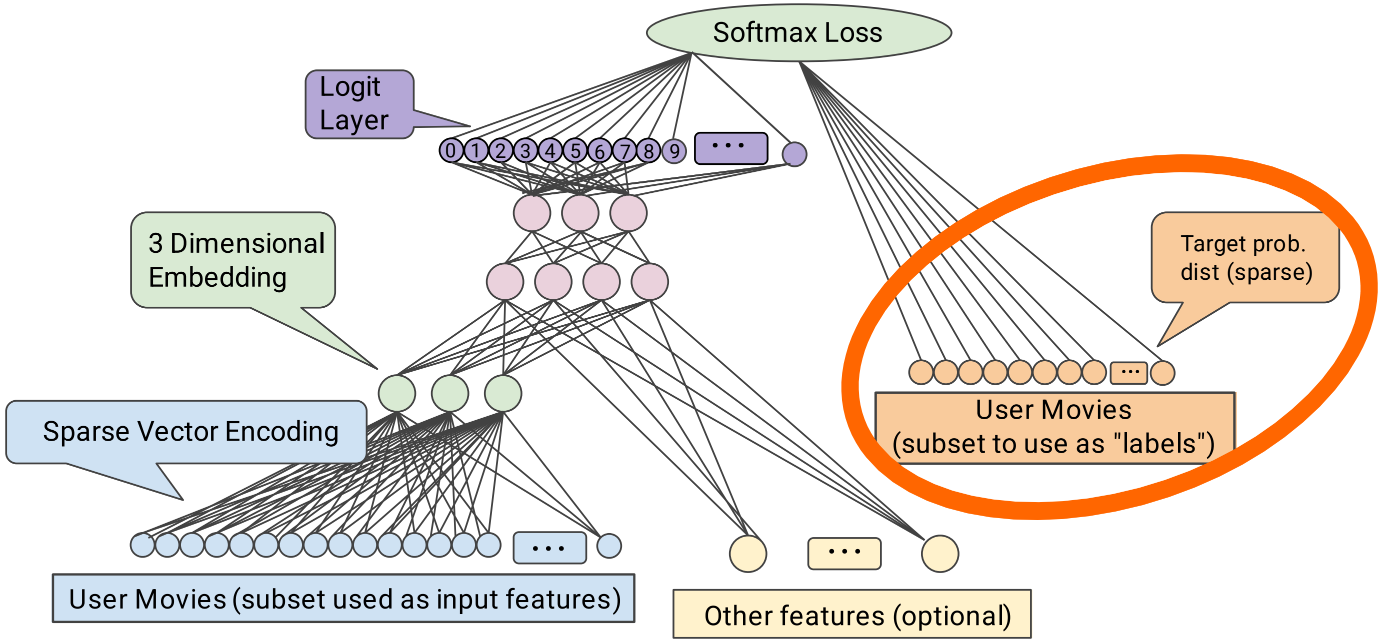
Deep Learning Based Recommenders

- Deep learning recommender models build upon existing techniques such as factorization to model the interactions between variables and embeddings to handle categorical variables
- An embedding is a learned vector of numbers representing entity features so that similar entities (users or items) have similar distances in the vector space



https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-2-deep-learning-for-recommender-systems/

Deep Learning Based Recommenders



Source: https://developers.google.com/

CF, CBF – Challenges/Solutions

- Sparse Matrices
  - Data sparsity problems
  - CF methods depend on matrix factorization of large sparse user-item matrices.
- Non-Linear Relationships
  - CF or CBF use linear techniques which fail to anticipate underlying nonlinear patterns.
  - To capture these nonlinearities, we build learning models with higher complexity
- Cold Start Problem
  - There may be no interaction/preference data for some users
  - To address this CBF techniques have evolved to leverage user and item features which makes recommender systems less sensitive to missing collaborative signals.

https://ebaytech.berlin/deep-learning-for-recommender-systems-48c786a20e1a

Cold Start

- New user
  - Rate some initial items
  - Non-personalized recommendations
  - Describe tastes
  - Demographic info.
- New Item
  - Non-CF : content analysis, metadata
  - Randomly selecting items
- New Community
  - Provide rating incentives to subset of community
  - Initially generate non-CF recommendation
  - Start with other set of ratings from another source outside community

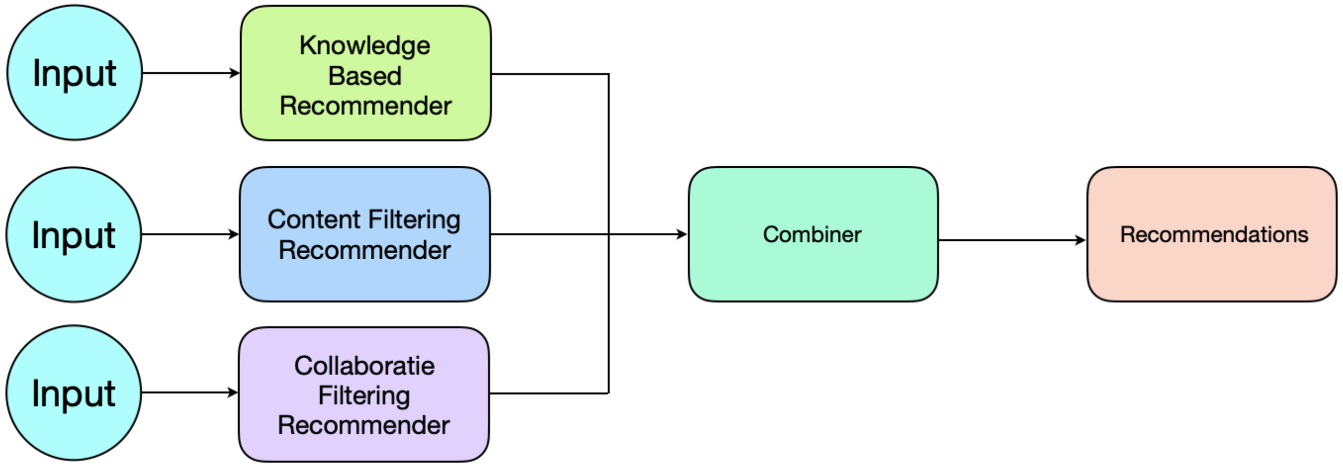
Collaborative Filtering Methods

	Memory Based	Model Based
Approach	• Calculates the similarity between users or items using the user's previous data based on ranking. • 3 Steps: Similarity computation, neighborhood selection and ratings prediction • E.g., User based, Item based CF	• Uses Machine Learning models to find user's ratings for unrated items • E.g., Matrix Factorization, SVD, Clustering, Neural Networks, etc.
Pros	• Easy creation and explainability of results • Simple to design and implement • It is very easy to update the database	• Can handle sparse matrices through dimensionality reduction • Scales well for large user-item matrices
Cons	• Scalability: Uses the entire database every time it makes a prediction. Does not scale for most real-world scenarios • Overfits the data; does not generalize well.	• Inference is intractable due to latent factors • Models can be time and resource intensive to train and deploy frequently

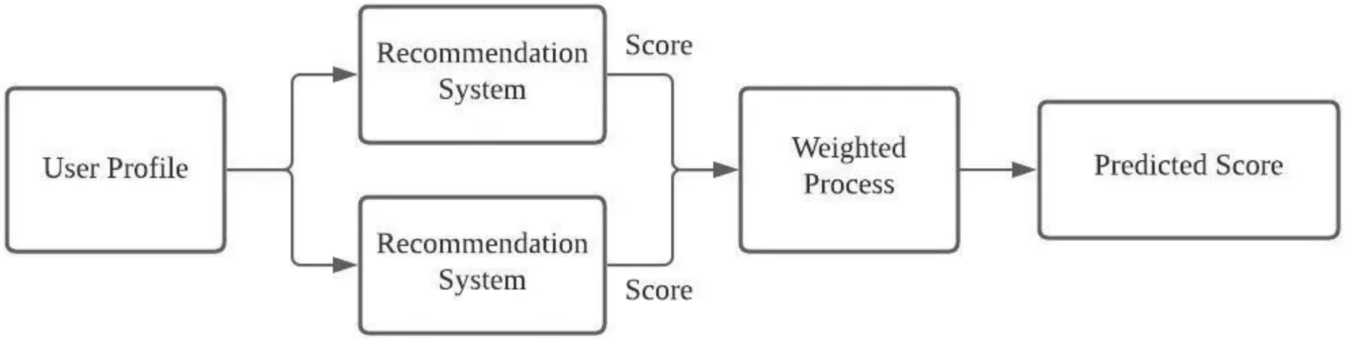
HYBRID RECOMMENDERS

Hybrid Recommenders

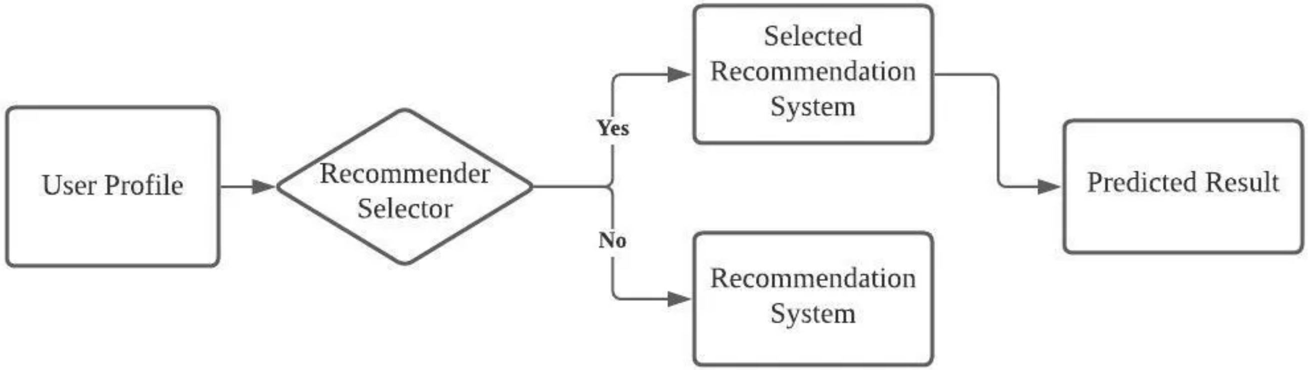
- Hybrid systems combine two or more techniques thereby alleviating the disadvantages of each technique.
- In practice, most recommender systems use a hybrid approach
- Seven categories of hybrid recommendation systems proposed by Burke (2002)
  - Weighted, switching, mixed, feature combination, feature augmentation, cascade, meta-level



Hybrid Recommenders – Types 1, 2



Weighted Hybrid Recommender

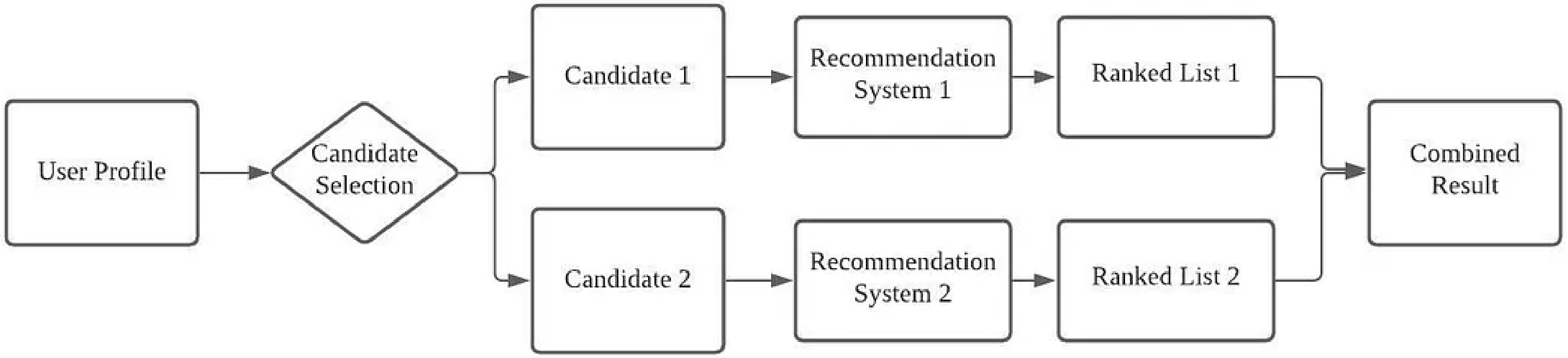


Switching Recommender

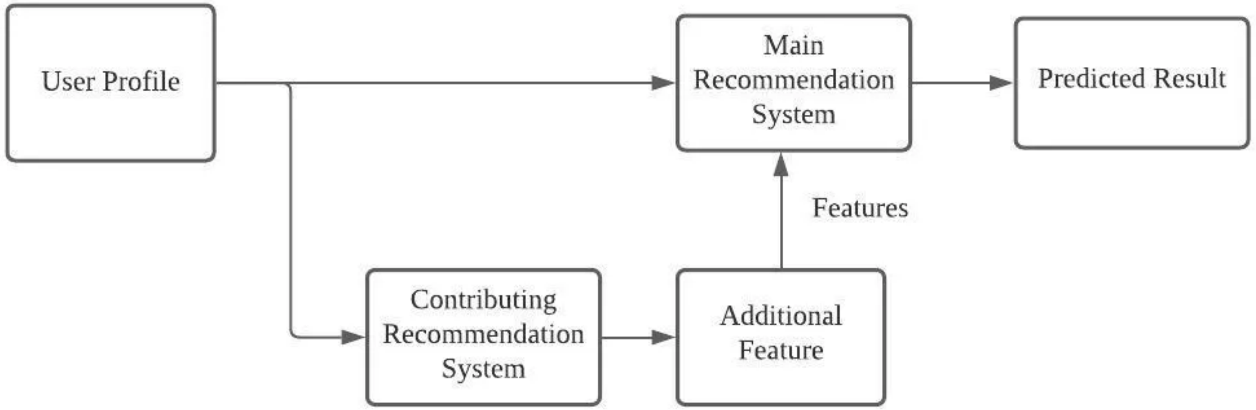
https://link.springer.com/article/10.1023/A:1021240730564

https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8

Hybrid Recommenders – Types 3, 4



Mixed Recommender

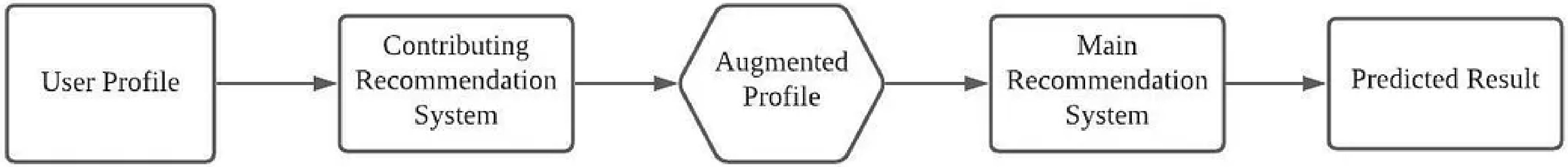


Feature Combination

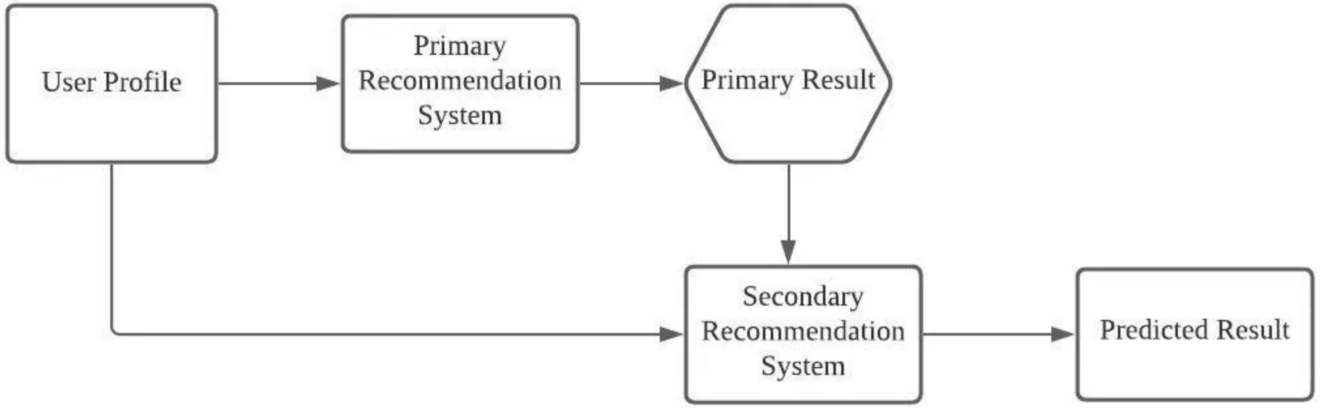
https://link.springer.com/article/10.1023/A:1021240730564

https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8

Hybrid Recommenders – Types 5, 6



Feature Augmentation System



Cascade Recommender

https://link.springer.com/article/10.1023/A:1021240730564

https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8