RECOMMENDER SYSTEMS
COLLABORATIVE FILTERING

AUTHORS

AMIR HOSSEIN BINESH, SHAHRYAR SHAHBAZI,

MOHAMMAD MAHDI SABER MAHANI

SUPERVISOR

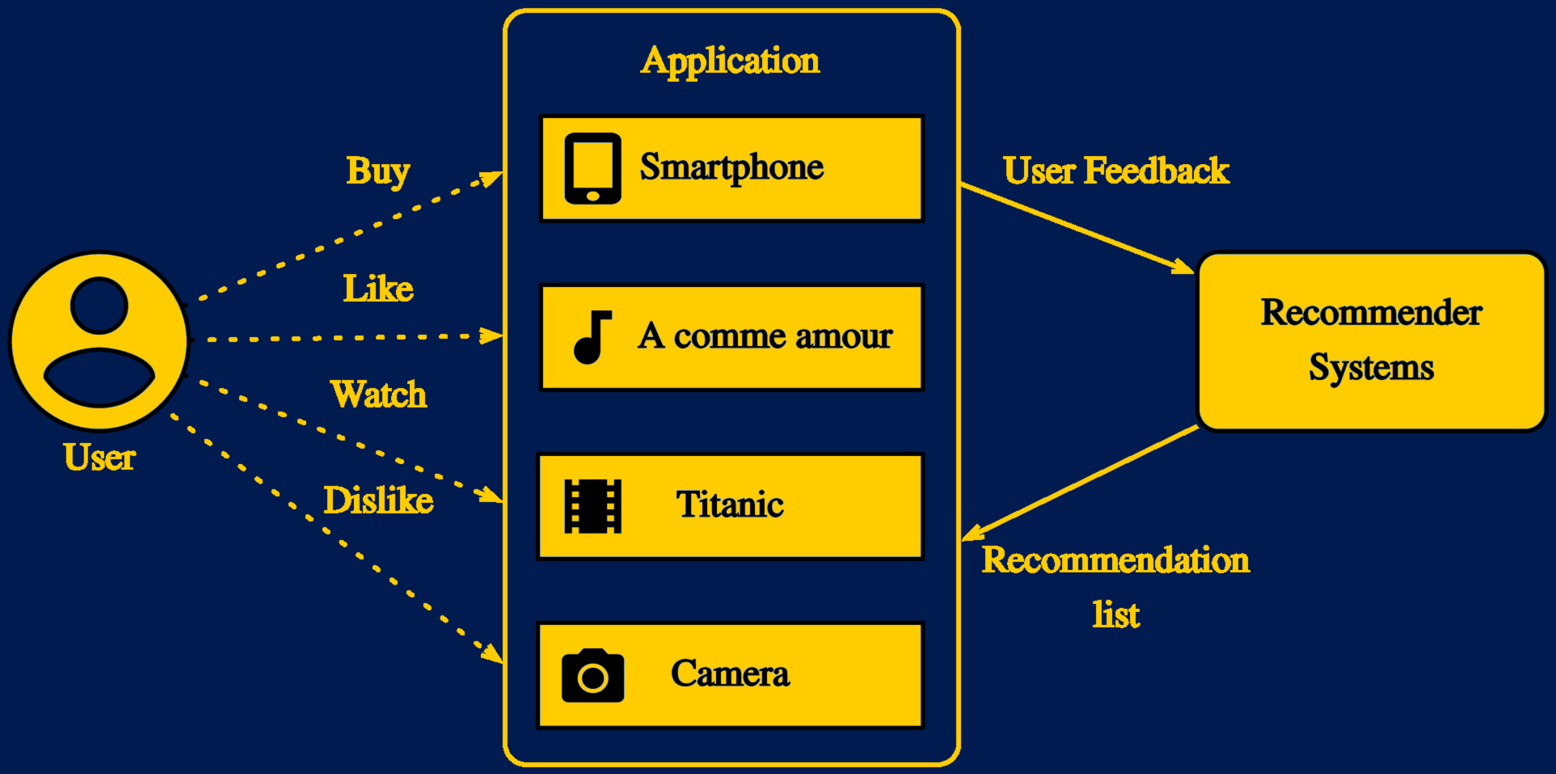
DR MAZLAGHANI

CONTENTS

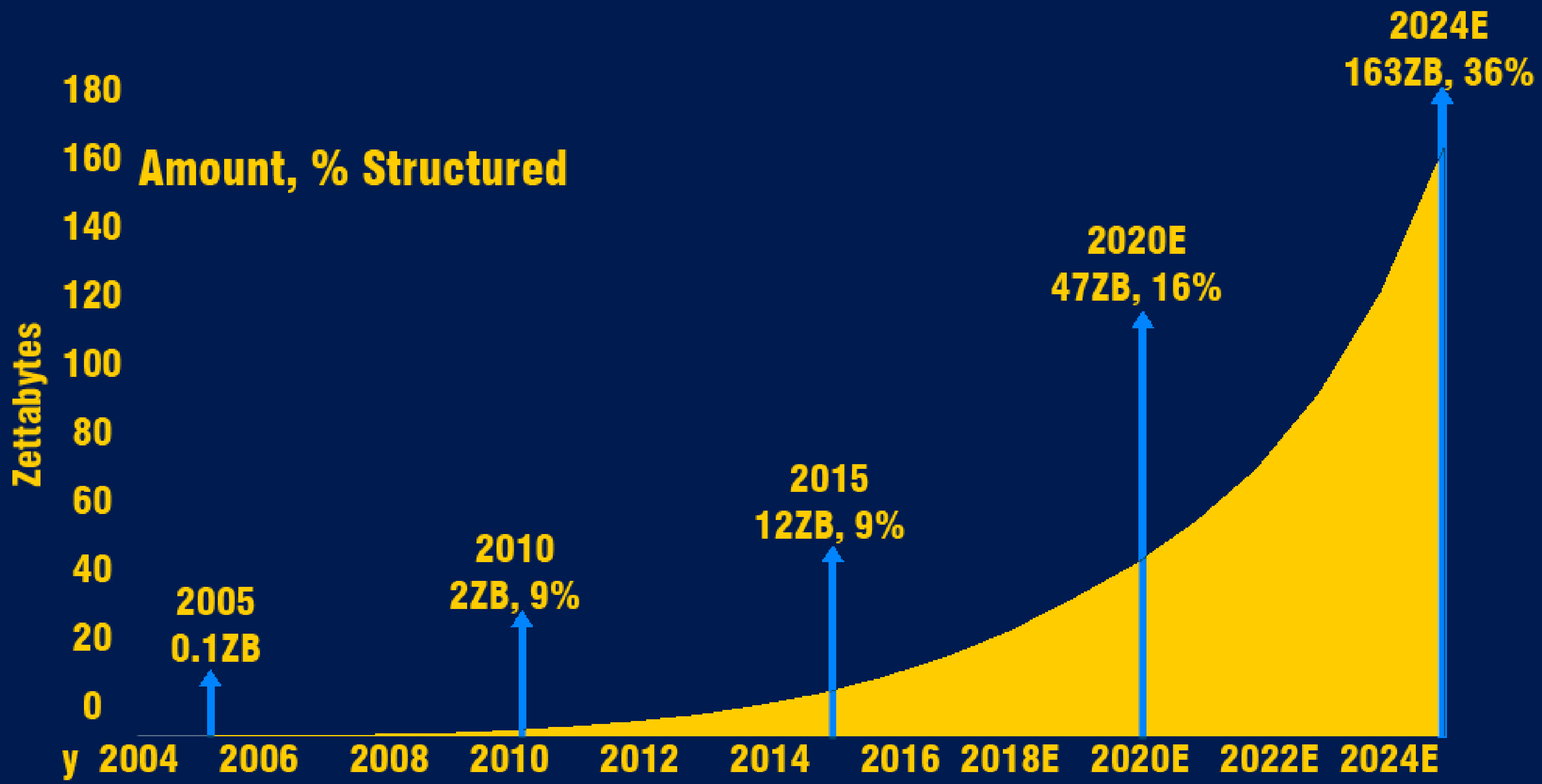
INTRODUCTION: What is a recommender system

Seeks to predict the “rating” a user would give to an item

A subclass of information filtering



INTRODUCTION: Why to use them



INTRODUCTION: Why to use them(cont.)

- Adaptive web
- Limited user attention
- Limited screen space

INTRODUCTION: How they help us

- as users
  - Less time searching
  - Easier decision making
  - High quality items
- as service providers
  - More revenue
  - User satisfaction
  - Increase user retention

INTRODUCTION: Recommender systems usages

- Video recommenders: Netflix, YouTube
- Music recommenders: Spotify
- Social friend/content recommenders: Instagram
- Shop recommenders: Amazon
- Game recommenders: Steam
- Search filtering: Google

INTRODUCTION: How they work

In a nutshell

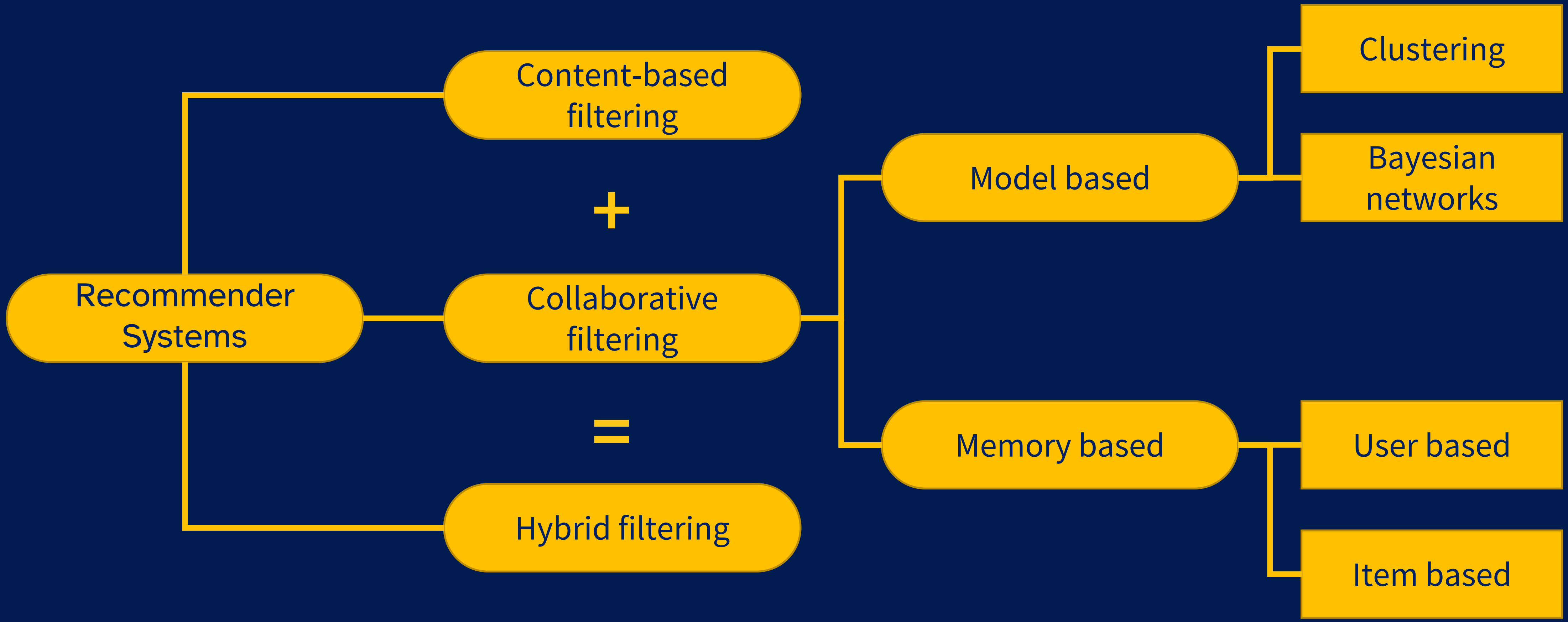
1. **Information collection:** A user rates an item
2. **Learning:** System learns user preferences
3. **Prediction:** System predicts future rates of that user
4. **Feedback:** Systems gets feedback of recommended item

INTRODUCTION: Challenges

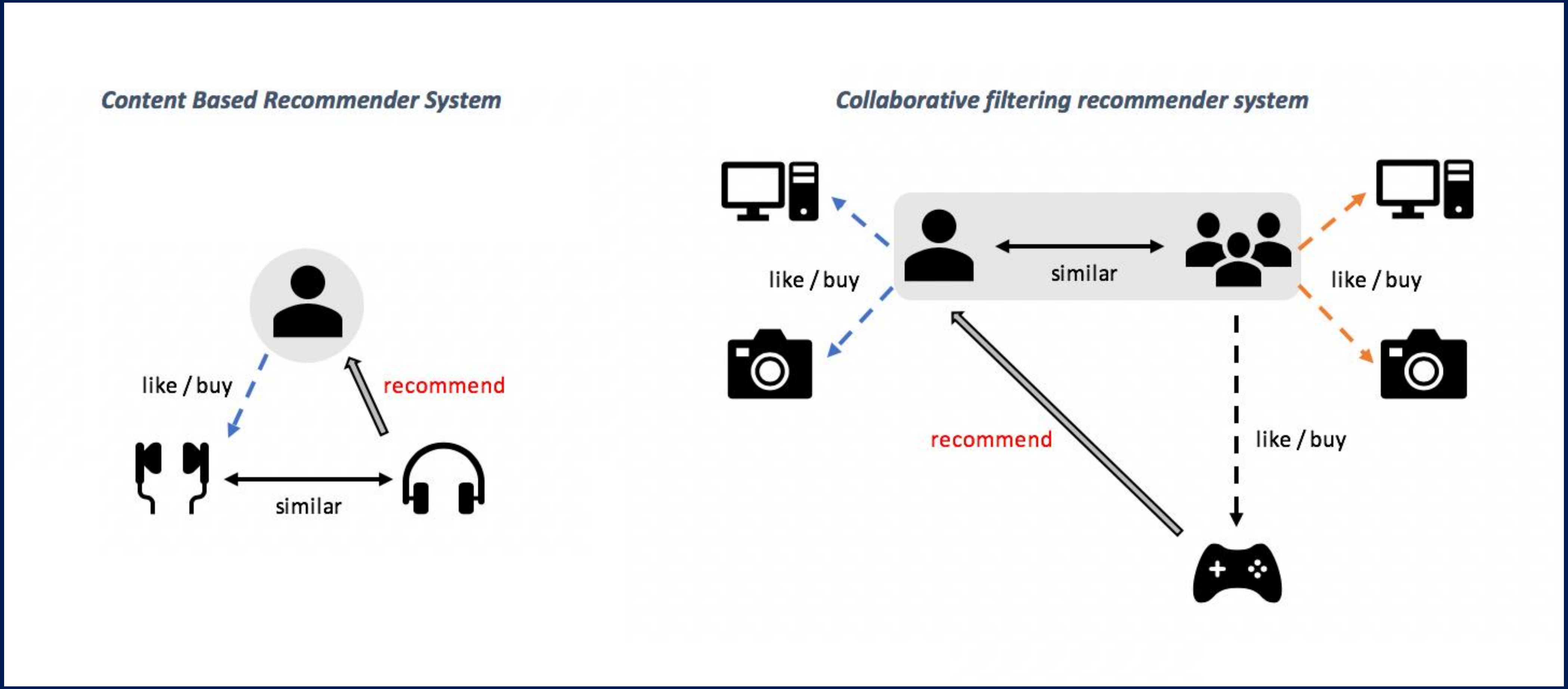
What makes it hard

- ? How should ratings look like?
- ? How to collect rating?
- ? How to learn?

APPROACHES: Preview



APPROACHES: Content based vs. Collaborative



APPROACHES: Content based filtering

- Domain dependent
- Based on analysis attributes of items
- Extracts Features of items a user has rated

Models used to find similar features(relations) between items:

- TF-IDF
- Naïve Bayes

APPROACHES: Content based filtering(cont.)

Suppose a user u watched some movies:

Movie	Liked
The Matrix	Yes
The Shining	No
Seven Samurai	No

Name	Action	Horror
The Matrix	1	0
The Shining	0	1
Seven Samurai	1	0
John Wick	1	0

$$P(John\ Wick \mid u) \propto P(u|John\ Wick) \times P(John\ Wick)$$

APPROACHES: Content based filtering(cont.)

Pros and cons

- + Can recommend items with no rating
- + Can adopt to changing tastes
- + User privacy

- – Depends on items metadata
- – Needs knowledge of item features
- – No serendipity

APPROACHES: Collaborative filtering

- Domain independent
- Based on user-item matrix
- Finds neighbors for users

In memory based: uses user-item matrix

In model based: makes a model using user-item matrix

APPROACHES: Collaborative filtering(cont.)

Pros and cons

- \+ Provides serendipitous
- \+ Doesn’t require content
- \+ Domain independent

- – Cold-start problem, community
- – Scalability
- – Trust

CF CONCEPTS: Intuition

	Item 1	Item 2	Item 3	Item 4
User 1	4	5	?	2
User 2	1		4	3
User 3	4	5	2	1
User 4	2	2		5

$$userSim(u, u') = \frac{u \,.\, u'}{|u|.\,|u'|}$$

$$pred(u, i) = avg(rating\ of\ similar\ users)$$

CF CONCEPTS: User tasks

1. Finding new items a user might like
2. Advising a user on a particular item
3. Finding new users a user might like
4. Finding an item that a group of users might like

CF CONCEPTS: Functionality

1. Recommend items
2. Predict for a given item
3. Constrained recommendation

CF CONCEPTS: Domain properties

- Data distribution
  - Many items
  - Many ratings per item
  - More users rating than items
  - Users rate multiple items

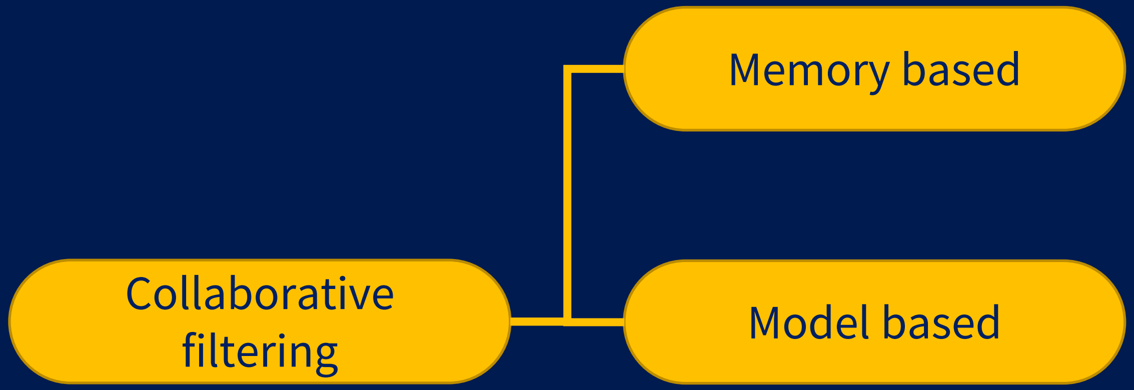
CF CONCEPTS: Domain properties(cont.)

- Underlying meaning
  - Similar users exist for each user
  - Item evaluation requires personal taste
  - Items are homogenous

CF CONCEPTS: Domain properties(cont.)

- Data persistence
  - Items persist
  - Taste persists

CF ALGORITHMS: Preview



Do not scale well for real application

Precomputation to reduce run-time

In real world applications, pure model based or hybrid methods are used

CF ALGORITHMS: User based

Generate predictions based on ratings from similar users

A naïve formula for prediction:

$$pred(u,i) = \frac{\sum_{n \subset neighbors(u)} r_{ni}}{number\ of\ neighbors}$$

This does not consider amount of similarity

$$pred(u,i) = \frac{\sum_{n \subset neighbors(u)} userSim(u,n).r_{ni}}{\sum_{n \subset neighbors(u)} userSim(u,n)}$$

CF ALGORITHMS: User based(cont.)

Users vary in their use of scale, so we adjust using mean rating; for example some optimistic users may give a movie between 4 of 5 but a pessimistic user may gives it 3.

$$pred(u,i) = \overline{r_u} + \frac{\sum_{n \subset neighbors(u)} userSim(u,n).(r_{ni} - \overline{r})}{\sum_{n \subset neighbors(u)} userSim(u,n)}$$

CF ALGORITHMS: User based(cont.)

For userSim(), we can use Pearson correlation

Pearson correlation differs between -1 and 1

It compares ratings for all items that are rated by both users (corated)

$$userSim(u,n) = \frac{\sum_{i \subset CR_{u,n}} (r_{ui} - \overline{r_u}).(r_{ni} - \overline{r_n})}{\sqrt{\sum_{i \subset CR_{u,n}} (r_{ui} - \overline{r_u})^2} . \sqrt{\sum_{i \subset CR_{u,n}} (r_{ni} - \overline{r_n})^2}}$$

CR denotes the set of corated items between two users

CF ALGORITHMS: User based challenges

- Pairs of users with little corated items are prone to skewed correlation
- Pearson correlation fails to incorporate agreement of whole about an item
- It scales linearly with the number of users and items. To reduce processing time and memory consumption:
  - Subsampling
  - Clustering

CF ALGORITHMS: Item based

Generates predictions based on similarities between items

$$pred(u,i) = \frac{\sum_{j \in ratedItems(u)} itemSim(i,j).r_{uj}}{\sum_{j \in ratedItems(u)} itemSim(i,j)}$$

Average correcting is not needed, since all the ratings are from one user

CF ALGORITHMS: Item based(cont.)

For itemSim() adjusted cosine similarity can be used:

$$itemSim(i,j) = \frac{\sum_{u \subset RB_{i,j}} (r_{ui} - \overline{r_u}).(r_{uj} - \overline{r_u})}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui} - \overline{r_u})^2} \cdot \sqrt{\sum_{u \subset BR_{i,j}} (r_{uj} - \overline{r_u})^2}}$$

RB denotes the set of users who have rated both items i and j

It compares the ratings for all users who rated both movies(corating)

* It’s Like Pearson correlation but average adjusting is performed with respect to the user, not the item

CF ALGORITHMS: Item based challenges

- Size of model can be as large as square of the number of items
  - Can be reduced
    - Only store correlations with more than k corating
    - Prune by retaining top n correlations
  - But it makes it difficult to predict for any given item
- Items with few coratings can let skewed correlations dominate a prediction

CF ALGORITHMS: Dimensionality reduction

Big CF applications have millions of users and items.

The user-item matrix is very sparse and computation take a while

To solve sparsity we can reduce the dimension to represent

- Latent topics (item-based)
- Latent tastes (user-based)

These techniques map ratings to tastes

Techniques:

- PCA
- SVD

CF ALGORITHMS: Dimensionality reduction challenges

- Requires an extremely expensive offline computation
- Harder debugging
- Harder maintaining

CF ALGORITHMS: Association rule mining

Build models based on commonly occurring patterns in rating matrix

- Example: Users who rated item 1 highly, often rate item 2 highly

$$Rule\ r : Liking\ item\ 1 \rightarrow Liking\ item\ 2$$

$$Support(r) = \frac{\#\ users\ who\ rated\ item\ 1\ and\ item\ 2}{total\ \#\ users}$$

$$Confidence(r) = \frac{\#\ users\ who\ liked\ item\ 1}{\#\ users\ who\ like\ both\ item\ 1\ and\ item2}$$

CF ALGORITHMS: Association rule mining challenges

- We lose any notion of the numeric relationship between ratings

  To solve this

  - Divide ratings into two bins: high and low
  - Only consider ratings above a user’s average
  - Treat all ratings as identical when building rules

- Too slow for CF domain due to the extremely high dimensionality

CF ALGORITHMS: Probabilistic algorithms

Calculate $p(r|u,i)$ for a given user u and an item I

Expected rating:

$$E(r|u,i) = \sum_{r} r.p(r|u,i)$$

Most popular probabilistic frameworks

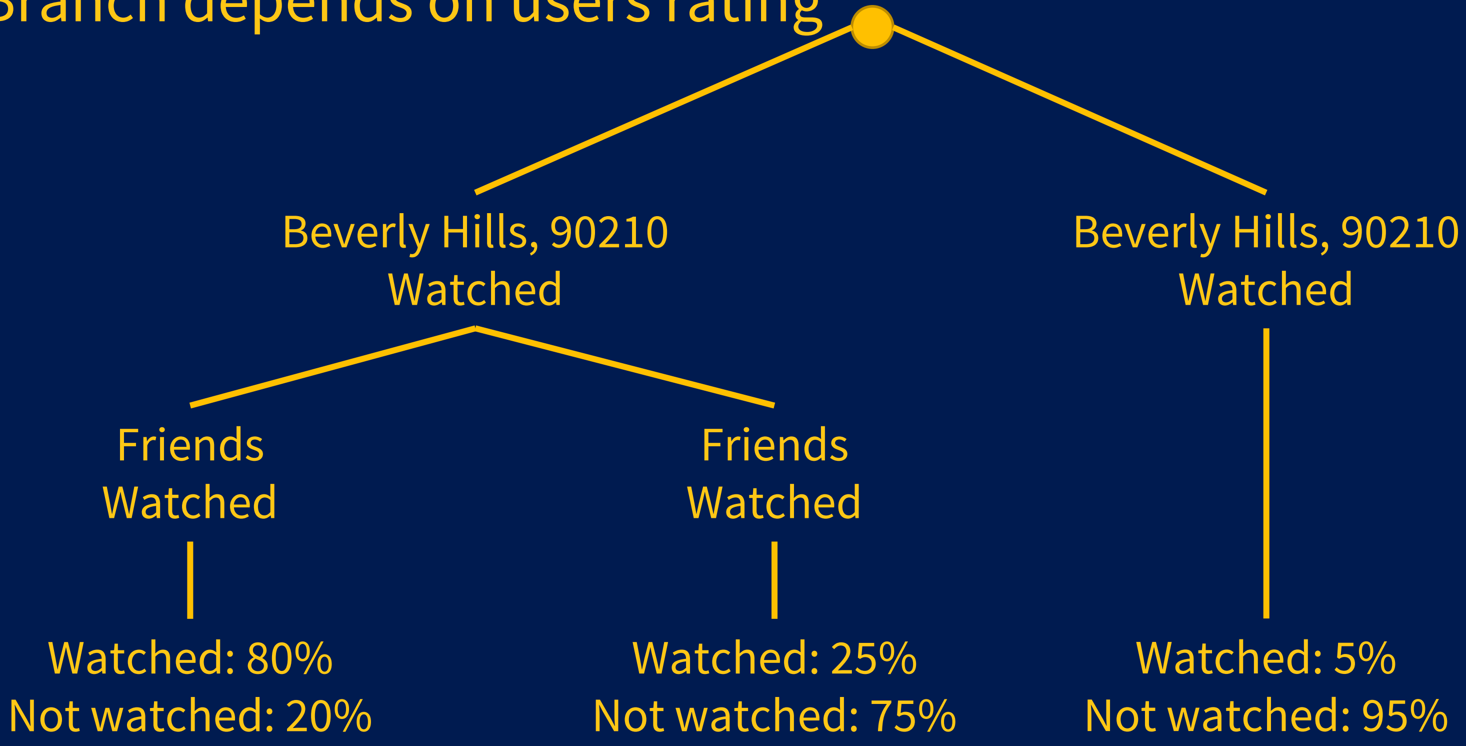
- Bayesian network: derive probabilistic dependencies among users or items
- Decision trees

Can also compute likelihood of a prediction being correct(confidence)

CF ALGORITHMS: Probabilistic algorithms(cont.)

Decision Trees

- A separate tree is constructed for each recommended item
- Branch depends on users rating



CF ALGORITHMS: Probabilistic algorithms(cont.)

Probabilistic dimension reduction

- Introduces a hidden variable $p(z|u)$ that represent the probability a user belong to hidden class z

$$p(r|u,i) = \sum_{z} p(r|i,z).\,p(z|u)$$

- The corresponding prediction is the expectation of rating value

$$E(r|u,i) = \sum_{r} r.\sum_{z} p(r|i,z)p(z|u)$$

CHALLENGES: Few ratings

Items and users with few ratings can bias CF results

Approaches

1. Discard rarely-rated items
   - In user based, discard neighbors with fewer than k common rated items
   - Will decrease coverage

CHALLENGES: Few ratings(cont.)

2 Adjust calculations for rarely-rated items

Pull them closer to an expected mean
Pearson similarities with few co-ratings may be adjusted closer to 0

3 Incorporate a prior belief

Match user's rating distribution to a probability distribution p
Use k artificial co-rated items

CHALLENGES: Prediction vs. Recommendation

- Recommend
  - Needs to predict some items(if not all)
  - Know about a subset of items
  - Pick a few alternatives according to user taste
- Predict
  - Store information about every item
  - Must have something to say about any item
  - Expensive

CHALLENGES: Confidence metrics

Two Metrics - Tradeoff

1. Recommend items with highest predicted rating
2. Recommend items with highest confidence on prediction

- User based: confidence measure that incorporates the agreement for an item in a user’s neighborhood

- Item based: measure the number of ratings for correlated pairs of items contributing to a prediction

CHALLENGES: Explicit vs. Implicit

EXPLICIT	IMPLICIT
Accurate	Imprecise
Additional work from user	Little of no cost to user
May not be enough	

More ratings leads to the ability to handle uncertainty

CHALLENGES: Explicit ratings collection

In order to succeed, CF needs relatively small number of “early adopters” who rate frequently and continuously.

Users rate because

- Feeling of contribution to advance a community
- Gratification from having one’s opinion voiced and valued

Also using incentives like “site points” or t-shirts can encourage users to rate

CHALLENGES: Rating Scales

The finer grained the scale:

- + The more information CF can have
- - More complex user interface
- - Increase uncertainty if too fine grained

Rating Scales

Rating scale	Description
Unary	Good or “don’t know”, Heart
Binary	Good or bad, Like/Dislike
Scalar	Stars, 1-5, 1-10 or ordinal

CHALLENGES: Cold start issue

A situation in which a recommender system is unable to make good recommendations due to an initial lack of ratings.

Three scenarios

1. New user

- Having a user rate some initial items before they can use the service
- Displaying non-personalized recommendations (population averages) until the user has rated enough
- Asking the user to describe their taste

CHALLENGES: Cold start issue

2 New item

If there may be many "sleepers" (unrated good items)

- Recommending new items using non-CF techniques such as content-based filtering
- Randomly selecting items with few or no rating and asking users to rate them

CHALLENGES: Cold start issue

3 New community

- Provide rating incentives to a small "bootstrap" subset of the community
- Maintain users' interest using non-CF methods
- Start with a set of ratings from another source

EVALUATION: Accuracy

Magnitude of error predicted rating and the true rating

$$mean\ absolute\ error = \sum_{i \subset PR} \frac{|rate(u,i) - pred(u,i)|}{|PR|}$$

Where PR is a set of items both predicted by recommender system (and recommended) and rated by a user for each user

EVALUATION: Accuracy

Mean absolute error does not differentiate between errors at the top and errors at the bottom of recommendation list

Using half-life utility metric, mistakes at the top of the ranked list are weighted exponentially grater than mistakes further down the list

Precision: Fewer false positives, less coverage

Recall: More coverage, More false positives

EVALUATION: Beyond accuracy

- Novelty
  Recommend items that the user was not already aware of
  Example: does not recommend news that I’ve seen already
- Serendipity
  Recommend items in new categories
  Example: recommends news topics that I have never read before
- Coverage
  Percentage of items that have the potential of being recommended

CONCLUSION

- Content based filtering can be effective in limited circumstances
- Machines can not automatically recognize subtleties of information that are important (at least for now)
- So we need to include people in the loop using Collaborative filtering methods

REFERENCES

Schafer, Ben & J, Ben & Frankowski, Dan & Dan, & Herlocker, & Jon, & Shilad, & Sen, Shilad. (2007). Collaborative Filtering Recommender Systems.

F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh (2015), Recommendation systems:Principles, methods and evaluation

https://www.monsoonblockchainstorage.com/data-growth/

https://visualbi.com/

THANKS