An Introduction to Recommender System

Poo Kuan Hoong

Agenda

- Introduction
  - Why do we need recommender system?
  - What is recommendation and personalization?
- Interactions
  - User/Items
- Recommender System Approaches
  - Content Based
  - Collaborative Filtering
    - Nearest Neighbor
    - Matrix Factorization
- Performance Metrics
- Simple Demo

Introduction

- Recommender System is widely used to provide relevant recommendations to users
- Websites such as Amazon, YouTube, Netflix, eBay and many more use recommendation engine

Why do we need recommender system?

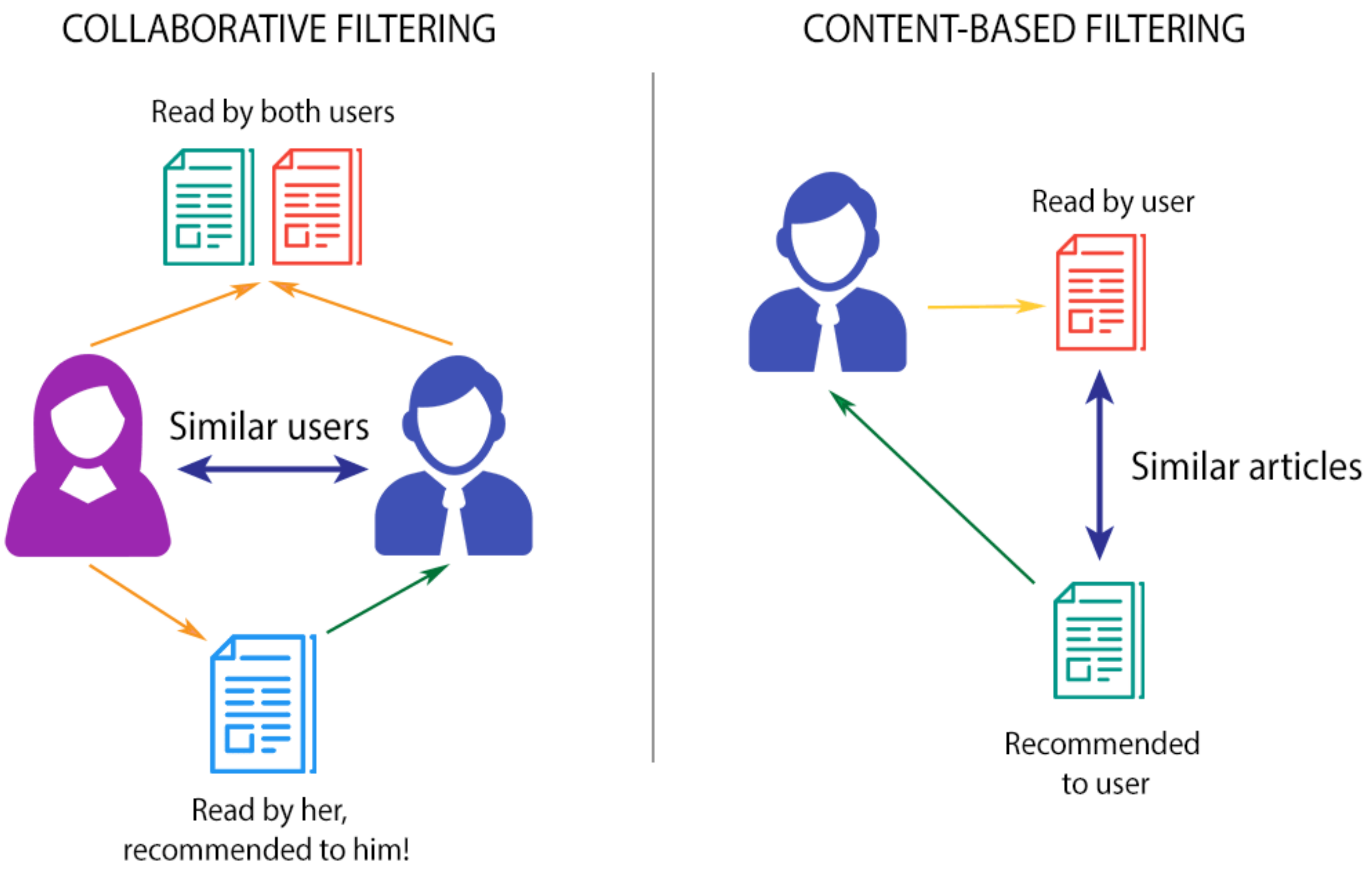
Relevant recommendation

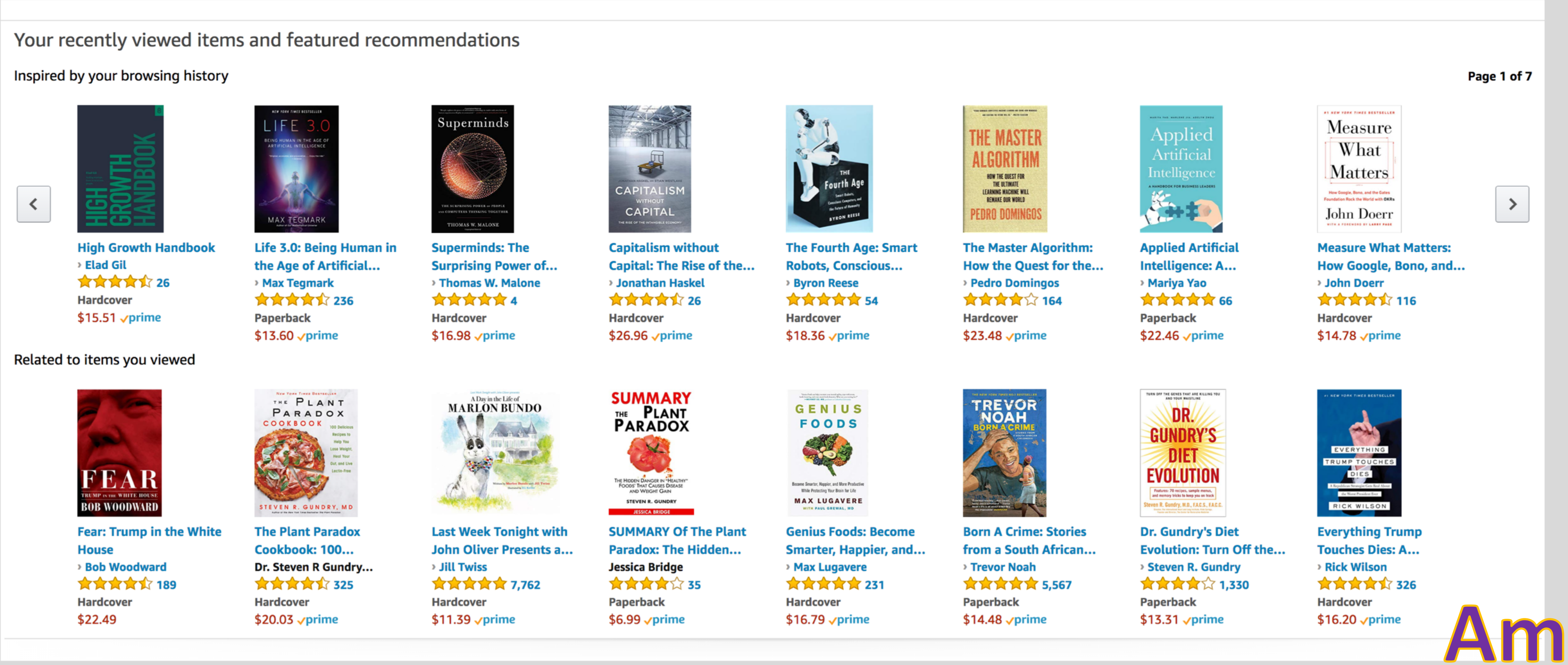
- Most people do not know what is available and what they are looking for

The Long Tail Phenomenon

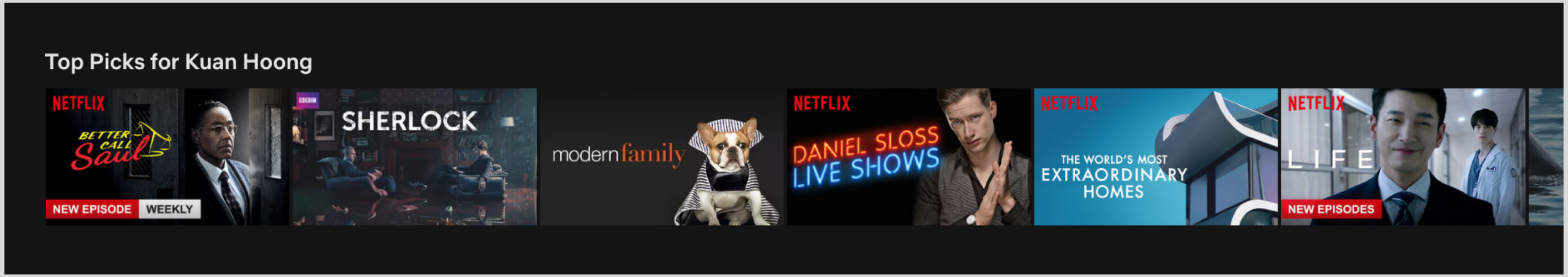
Recommendation

- A recommendation system presents **items** to **users** in a relevant way
- The definition of **relevant** is product/context-specific





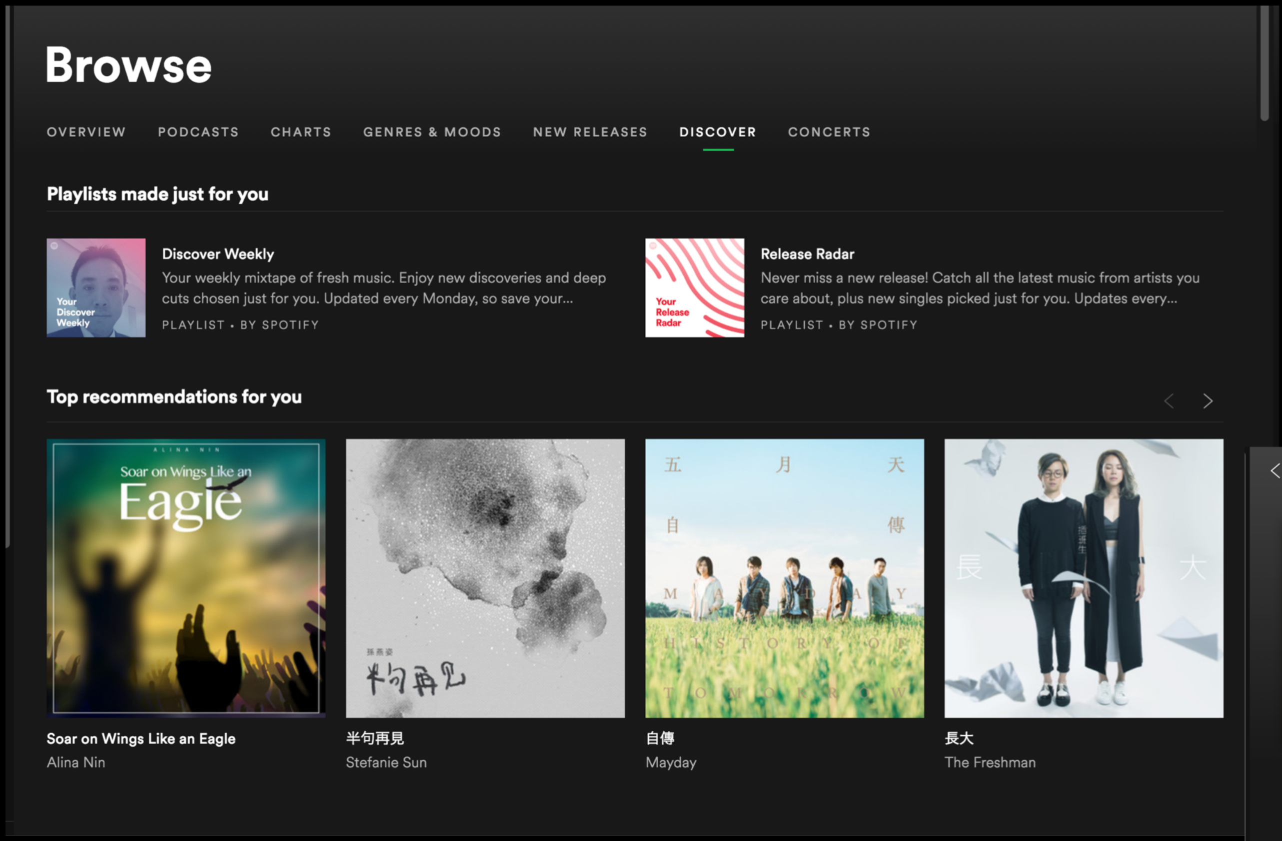
Amazon

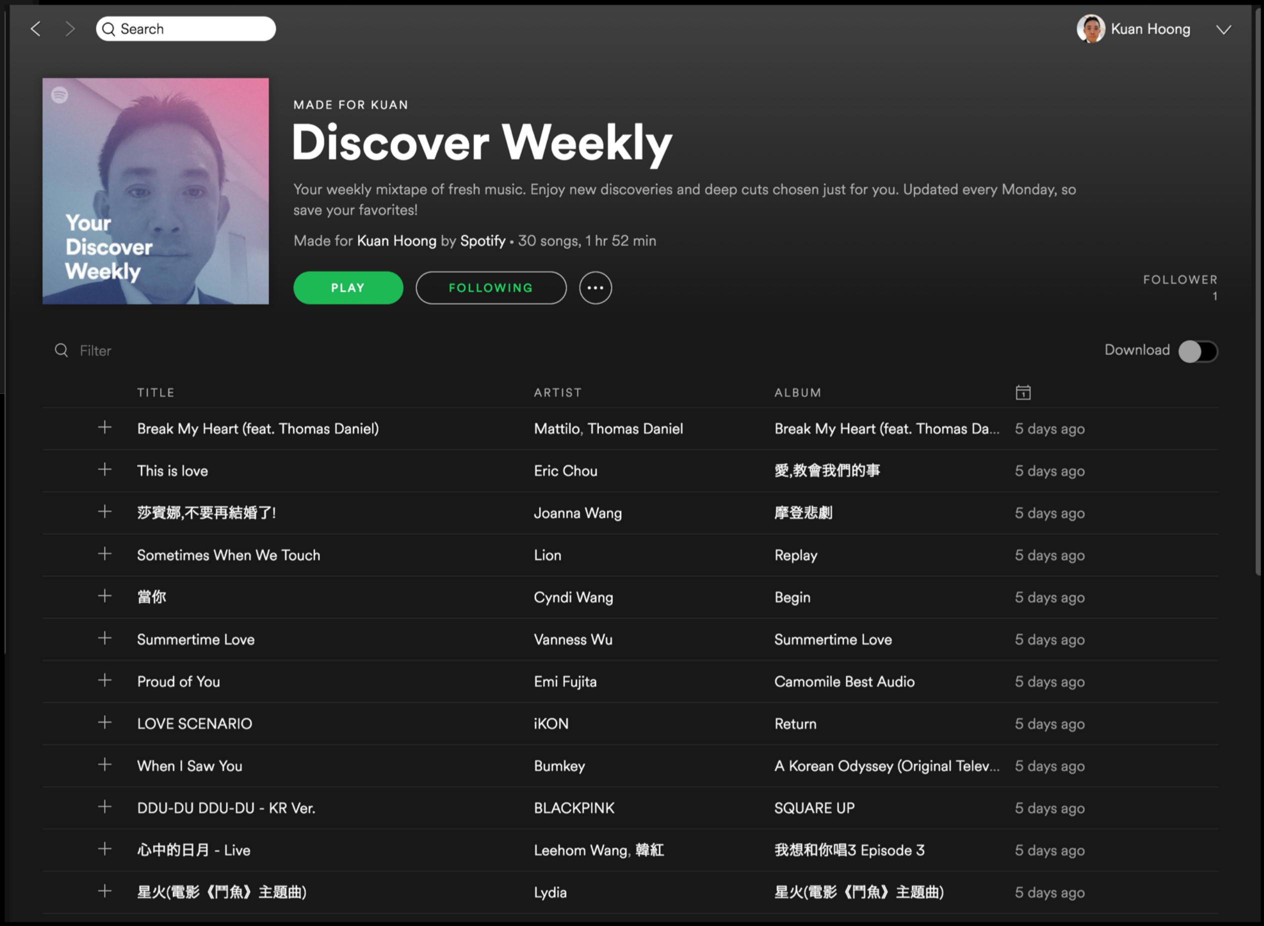


Netflix

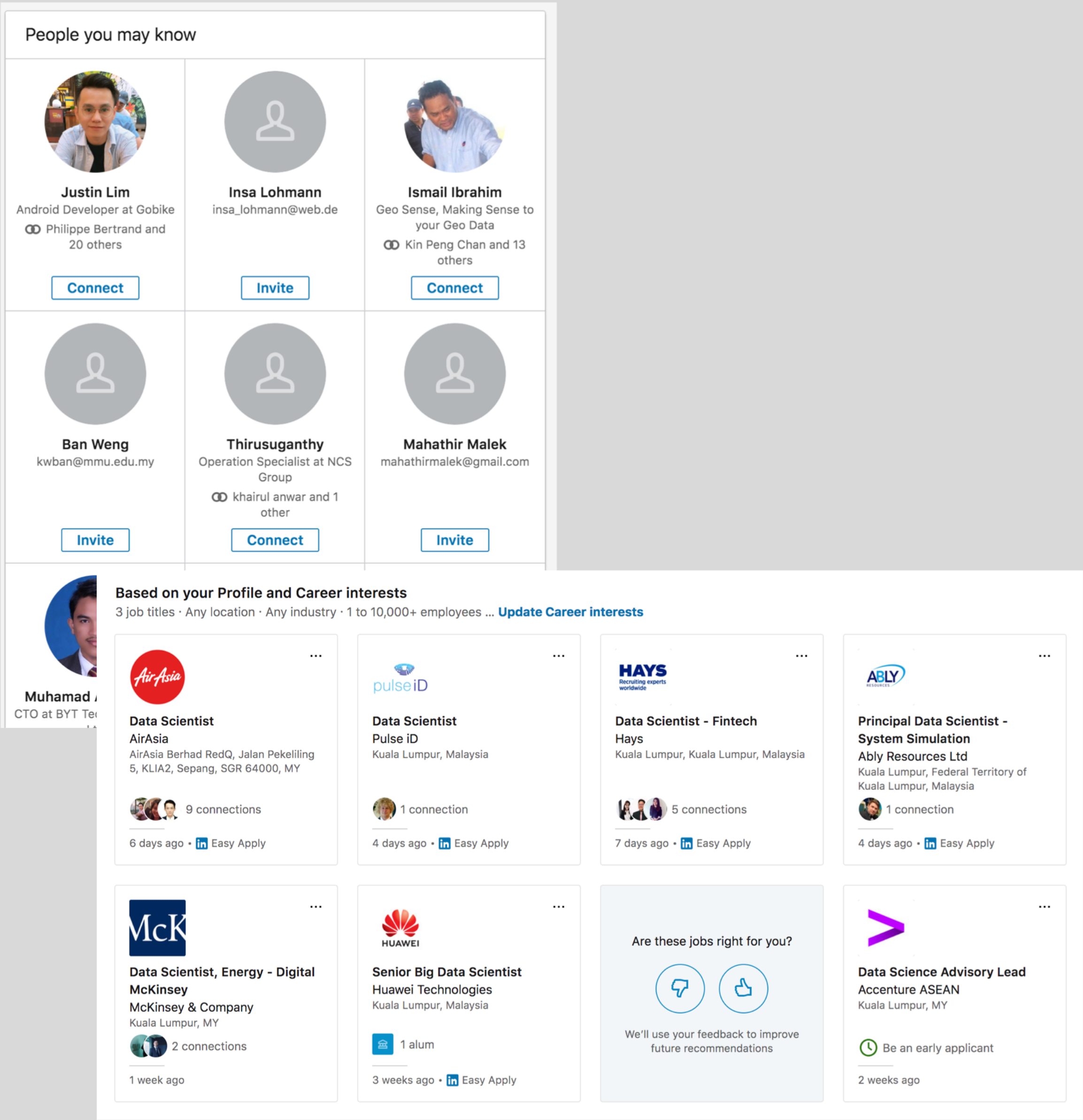
Personalization

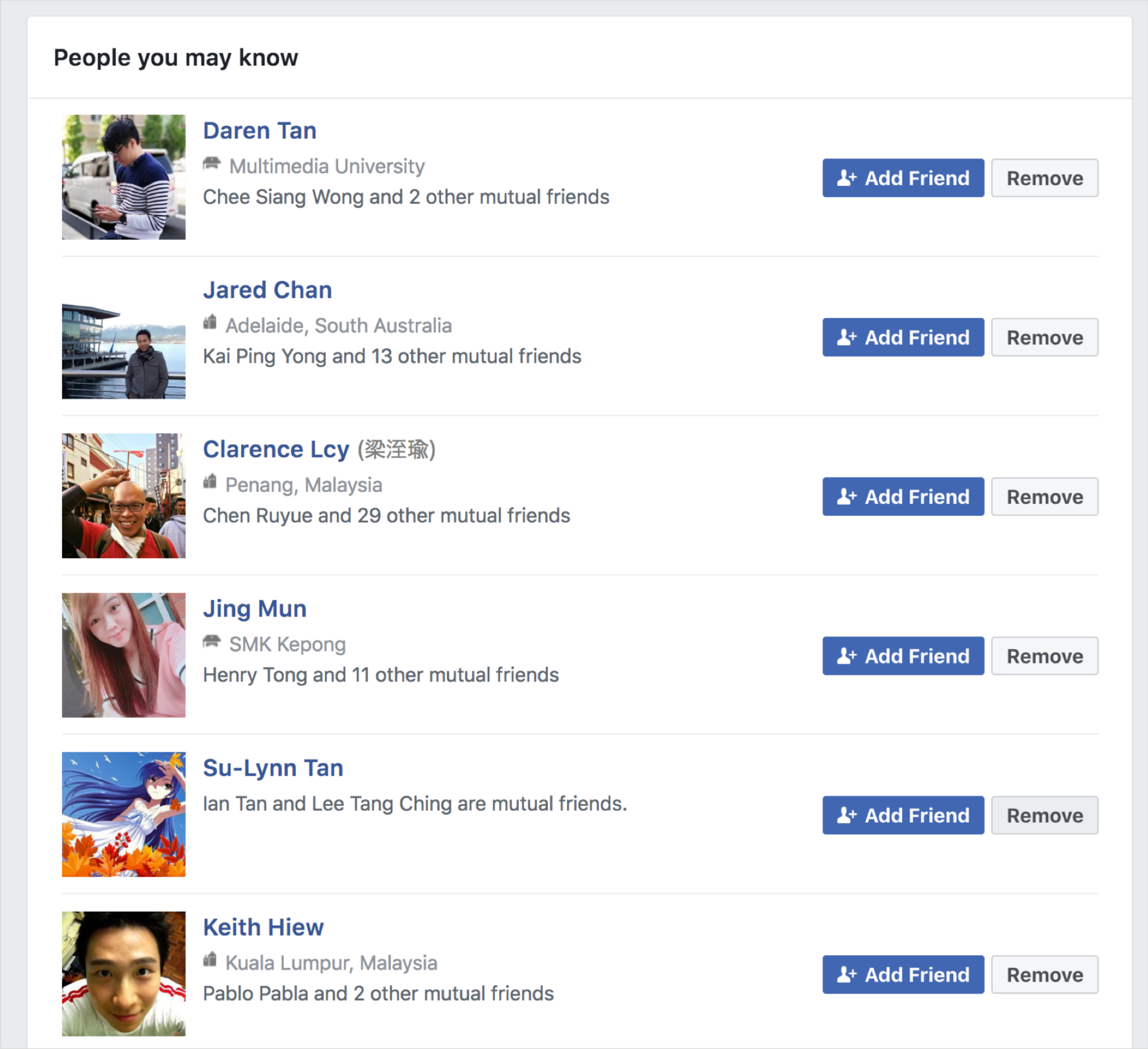
- A **personalization** system presents recommendations in a way that is **relevant** to the **individual user**
- The **user** expects their experience to change based on their **interactions** with the system
- **Relevance** can be in the context or product specific





Personalization





LinkedIn

Facebook

Users vs Items

Users

- A **user** in a recommender system is the party that is **receiving and acting on** the recommendations.
- Sometimes the user is the **context**, not an actual person.

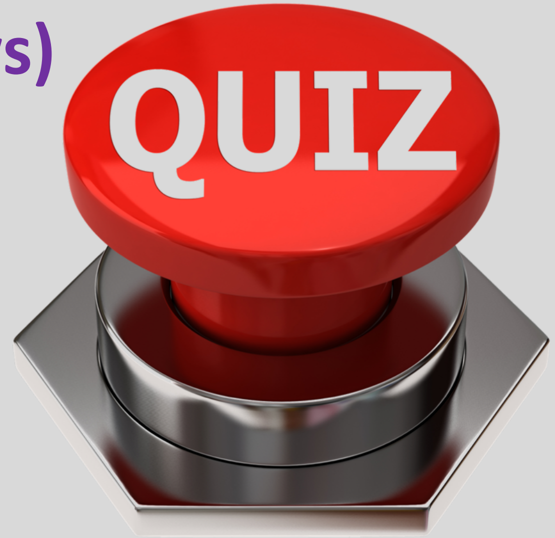
Items

- An **item** in a recommender system is the **passive party** that is being recommended to the users.

The line between these two can be blurry.

Quiz

- What is the matching user vs items for the followings?
  - LinkedIn (Users: Members, Items: Jobs)
  - Facebook (Users: Members, Items: Friends)
  - Amazon (Users: Members, Items: Books/Products)
  - Netflix (Users: Members, Items: Movies/Shows)
  - Spotify (Users: Members, Items: Songs)



Interactions

Positive

- Thumb up, hearts, stars, listens, watches, follows, bids, purchases, reads, views, upvotes…

Negative

- Thumb down, skips, angry-face, 0-star review, unfollows, rejections, returns, downvotes..

Explicit vs Implicit

- Explicit actions are those that a user expects or intends to impact their personalized experience
- Implicit actions are all other interactions between users and items

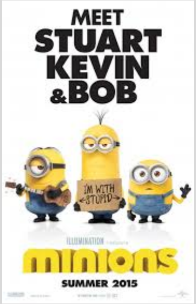
Recommender System (RS) Approaches

- **Content-Based (CB)** – use personal preferences to match and filter items
  - E.g. what sort of books do I like?
- **Collaborative Filtering (CF)** – match `like-minded' people
  - E.g. if two people have similar 'taste' they can recommend items to each other
- **Social Software** – the recommendation process is supported but not automated
  - E.g. Weblogs provide a medium for recommendation
- **Social Data Mining** – Mine log data of social activity to learn group preferences
  - E.g. web usage mining
- For this talk, we will concentrate on CB and CF

Content-Based

- Find me things that I liked in the past.
- Machine *learns* preferences through user feedback and builds a user *profile*
- Explicit feedback – user rates items
- Implicit feedback – system records user activity
  - Clickstream data classified according to page category and activity, e.g. browsing a product page
  - Time spent on an activity such as browsing a page
- Recommendation is viewed as a search process, with the user profile acting as the query and the set of items acting as the documents to match.

Example: Content Based



Animated	Yes	Yes	No	No
Superheroes	No	No	Yes	Yes
Super Villains	Yes	No	Yes	Yes
Came out on 2015	Yes	Yes	No	No
Marvel	No	No	Yes	Yes

2

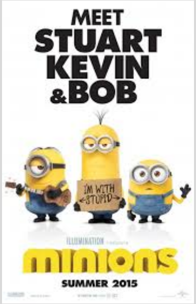
1

1

Collaborative Filtering (CF)

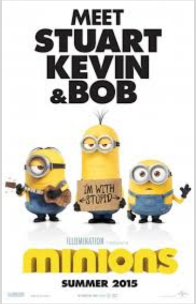
- Match people with similar interests as a basis for recommendation.
- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- Approach
  - use the "wisdom of the crowd" to recommend items
- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

Example



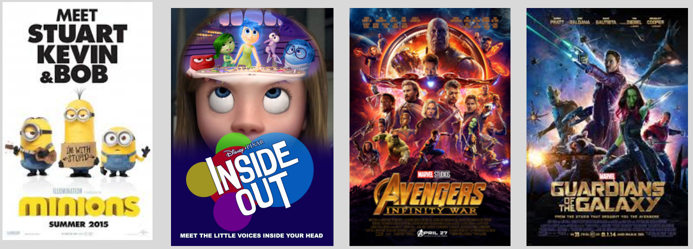
John	Yes	Yes	Yes	Yes
Peter	No	No	Yes	Yes
Ian	Yes	No	Yes	No
Tom	No	Yes	No	No
Marcus	Yes	?	?	?

Example: Collaborative Filtering (CD)



John	Yes	Yes	Yes	Yes
Peter	No	No	Yes	Yes
Ian	Yes	No	Yes	No
Tom	No	Yes	No	No
Marcus	Yes	?	?	?
		1 vote	**2 votes**	**1 vote**

Example: Collaborative Filtering (CD)



John	5	5	4	3
Peter	0	0	3	5
Ian	3	0	4	4
Tom	1	5	5	1
Marcus	?	?	?	?

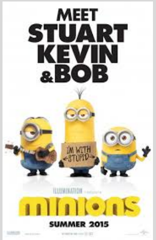
$$
\begin{array}{c c}
 & \begin{array}{cccc} \textbf{item1} & \textbf{item2} & \textbf{item3} & \textbf{item4} \end{array} \\
\begin{array}{c} \textbf{user1} \\ \textbf{user2} \\ \textbf{user3} \\ \textbf{user4} \\ \textbf{user5} \end{array} &
\begin{pmatrix} 5 & 5 & 4 & 3 \\ 0 & 0 & 3 & 5 \\ 3 & 0 & 4 & 4 \\ 1 & 5 & 5 & 1 \\ ? & ? & ? & ? \end{pmatrix}
\end{array}
$$

Collaborative Filtering (CF)

- Input
  - Only a matrix of given user–item ratings
- Output types
  - A (numerical) prediction indicating to what degree the current user will **like** or **dislike** a certain item
  - A top-N list of recommended items

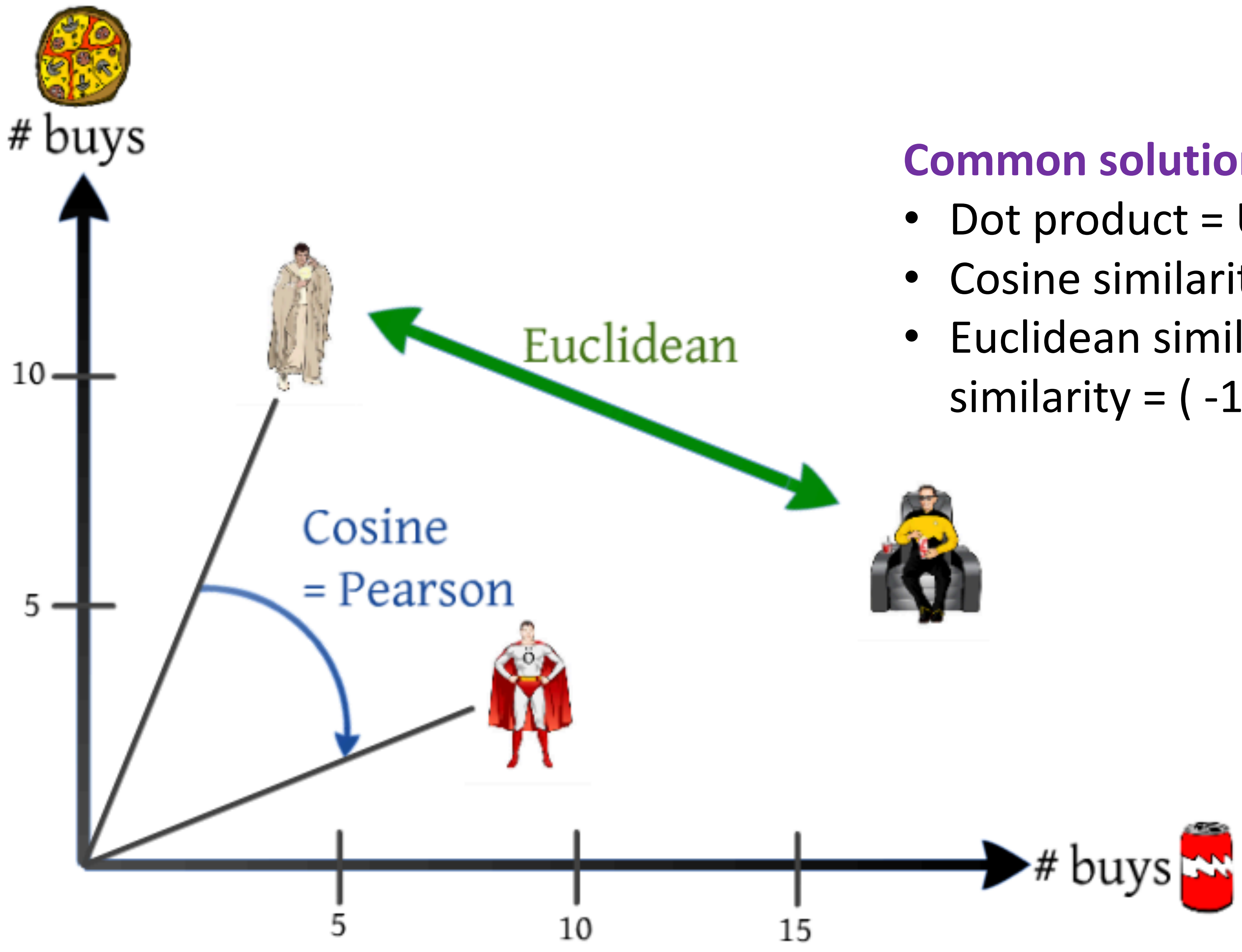
User-based k-nearest neighbor collaborative filtering

- The basic technique
  - Compute similarity of users
  - Find k most similar users to user ***a***
  - Recommend movies not seen by ***a***
- Basic assumption and idea
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time



John	5	5	4	3
Peter	0	0	3	5
Ian	3	0	4	4
Tom	1	5	5	1
Marcus	?	**?**	**?**	**?**

Cosine Similarity $$sim(a,b) = \frac{a.b}{||a||.||b||}$$



Common solutions:

- Dot product = User · Item
- Cosine similarity = cos(Θ)
- Euclidean similarity* = (-1 Manhattan similarity = (-1 * |User - Item|)

Matrix Factorization – Collaborative Filtering

- Identify latent (hidden) features from the input user x itemRatings matrix to represent users and items as vectors in N dimensional space.
- Matrix factorization based methods attempt to reduce dimensionality of the interaction matrix and approximate it by two or more small matrices with k latent components.

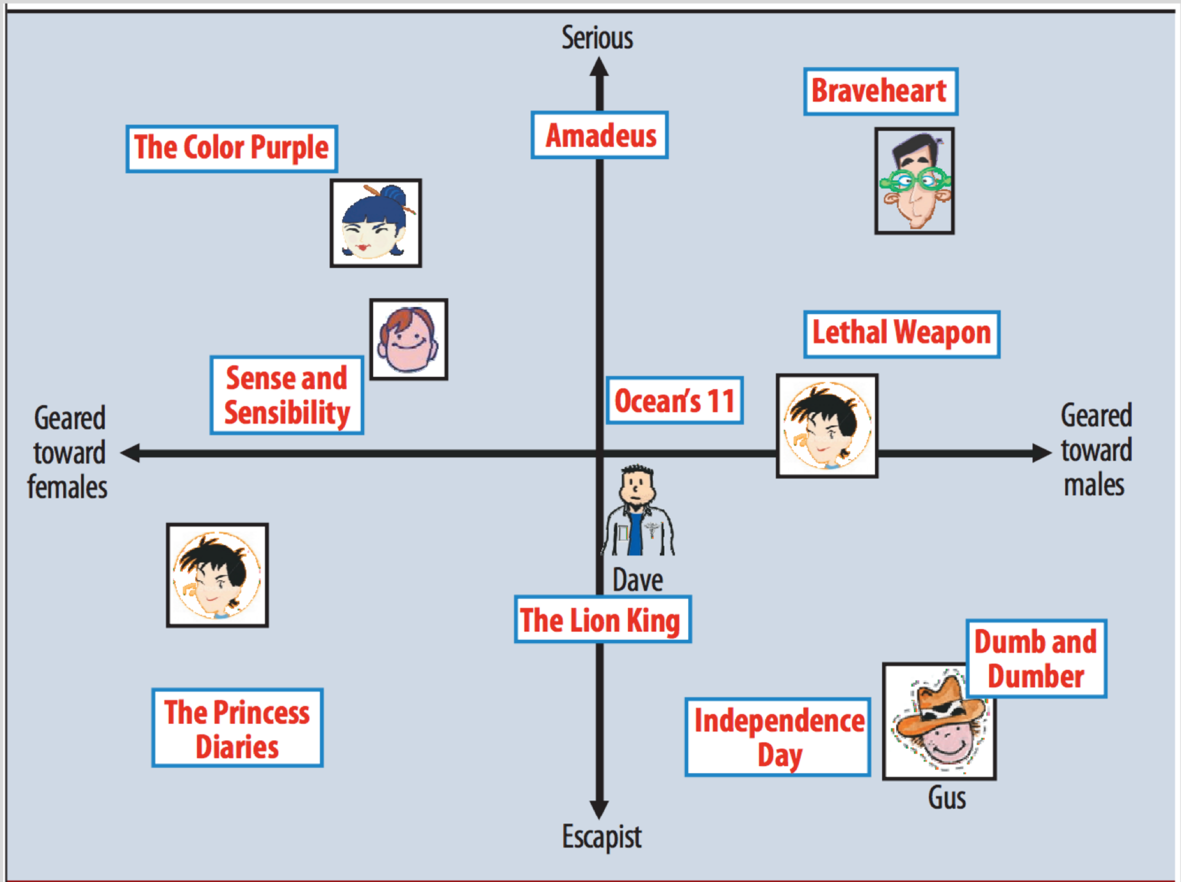
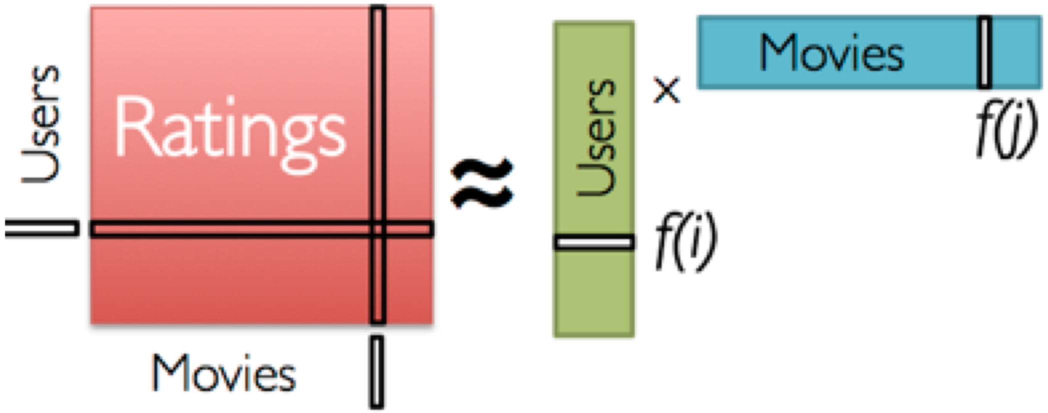


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Matrix Factorization – Collaborative Filtering

- **Training:** Use Matrix factorization approaches (Eg. Singular value Decomposition or SVD) to split the Rating Matrix into constituent User Matrix and Item Matrix with minimum Sum of squared error (SSE).
- **Goal:** Predict unknown ratings for the remaining set of movies using the learned User Matrix and Item Matrix

	item1	item2	item3	item4
user1	5	5	4	3
user2	0	0	3	5
user3	3	0	4	4
user4	1	5	5	1
user5	?	?	?	?



MATRIX FACTORISATION

users (columns) × items (rows)

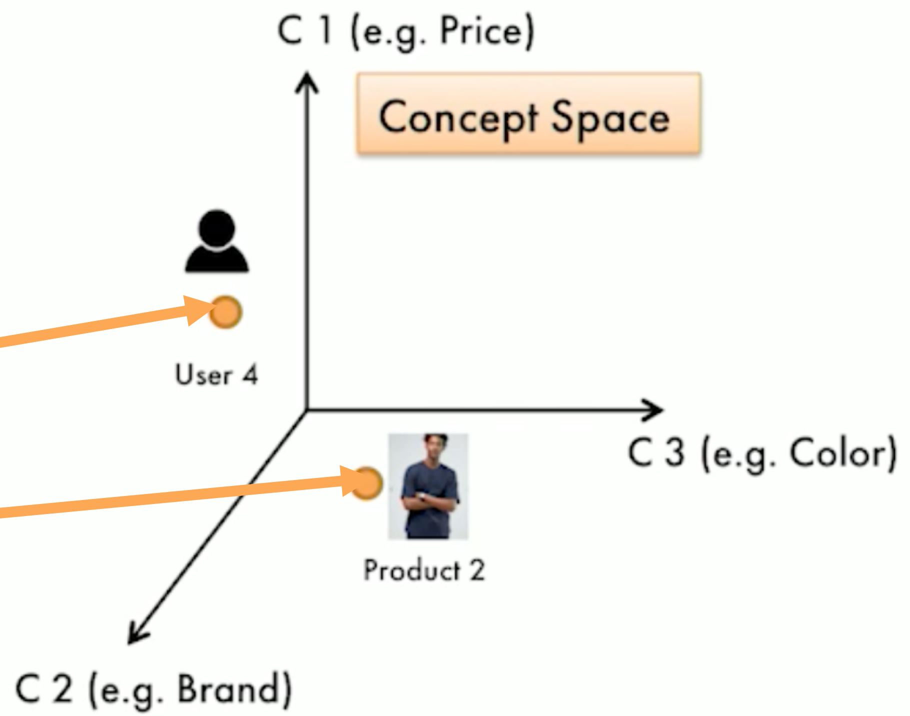
1		3			5			5		4	
		5	4			4		2	1	3	
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

factors (columns) × items (rows)

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.2	2.1	.3
-.7	2.1	-2
-1	.7	.3

users (columns) × factors (rows)

1.1	-2	.3	.5	-2	-.5	8	-.4	.3
.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8



Interactions → Features → Learning

Interactions

Identify interaction values

- Types of behaviors
- Explicit vs Implicit
- Do you need negative interactions?

Features

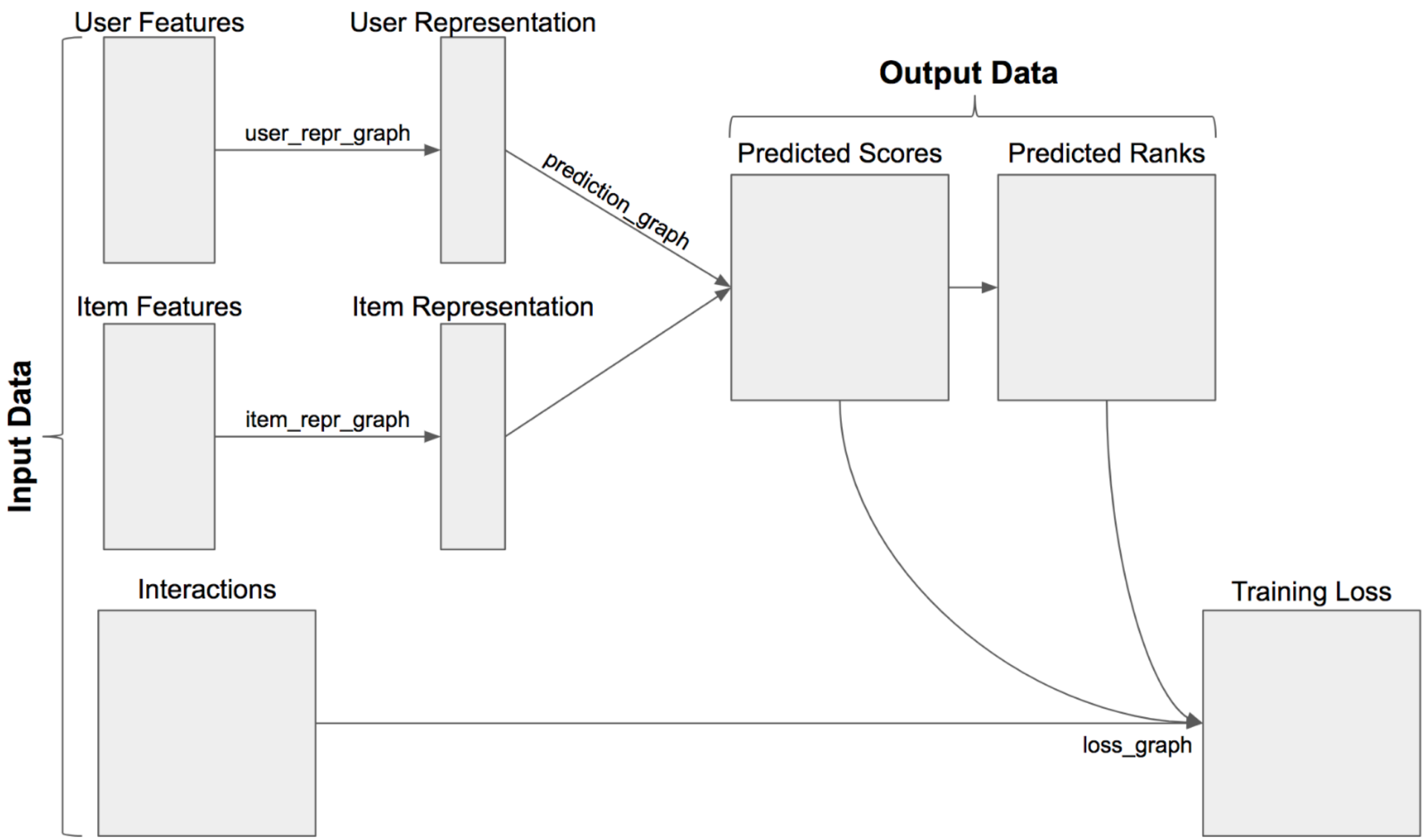
Identify user/item features

- Indicator features
- Use metadata
- Feature Engineering

Learning

Identify learning process

- Which prediction method
- Identify loss function
- How do you evaluate?

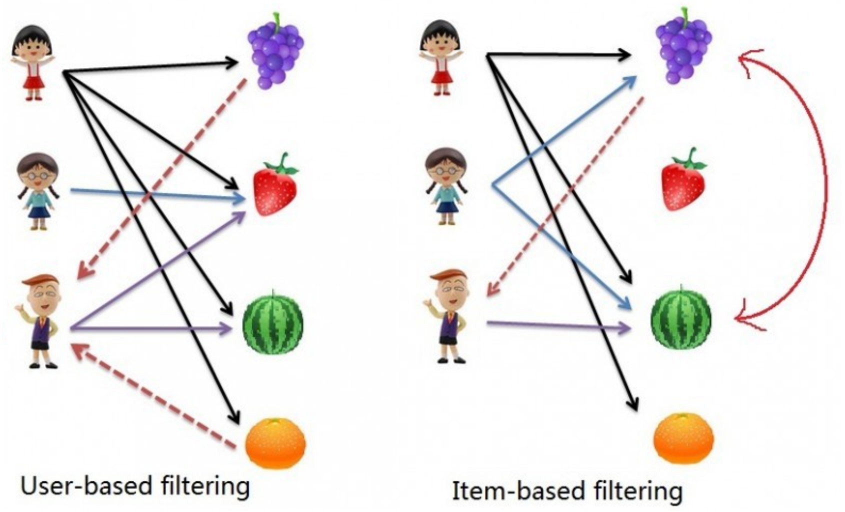

User Features
User Representation
user_repr_graph
prediction_graph
Output Data
Predicted Scores
Predicted Ranks
Item Features
Item Representation
item_repr_graph
Input Data
Interactions
Training Loss
loss_graph

Identify Interaction Values

- Do we allow negative interactions?
  - Negative interactions can be valuable statements of what content to avoid
  - Negative interactions can be confusing when learning-to-rank
  - Not all loss functions accommodate negative interactions

Identify user/item features

- Indicator features allow for powerful personalization but are as numerous as user/item
- recommenders with user interactions can not effective make recommendations for new users (cold-start problem)
- Many users means many indicator features – this may not scale
- Identify preprocessing processes to improve recommender learning
- Remove useless/misleading features
- The choice of representation function impacts the usefulness of feature engineering



Identify learning process

- What is the effective ways for the recommender system to learn?
- Use of linear kernel or well engineered features (matrix factorization) are effective
- How about complicated models such as neural networks or deep learning?
- Can the learning process will represented without interactions such as auto-encoders, word2vec, etc
    - Identify loss functions – Root Mean Square Error (RMSE), Weighted approximately ranked pairwise (WARP), Weighted margin-rank batch (WMRB), etc
- What is the evaluation process?

Performance Metrics

- What do we want to measure?
  - Clicks
  - Likes/Saves
  - Actions – Download, Install, etc

Loss Function

- The process that **converts predictions and interactions** in to error for **learning**

- Common examples:
  - Root-mean-square error (RMSE)
  - Kullback-Leibler divergence (KLD)
  - Alternating least squares* (ALS)
  - Bayesian personalized ranking* (BPR)
  - Weighted approximately ranked pairwise (WARP)
  - Weighted margin-rank batch (WMRB)

Other considerations

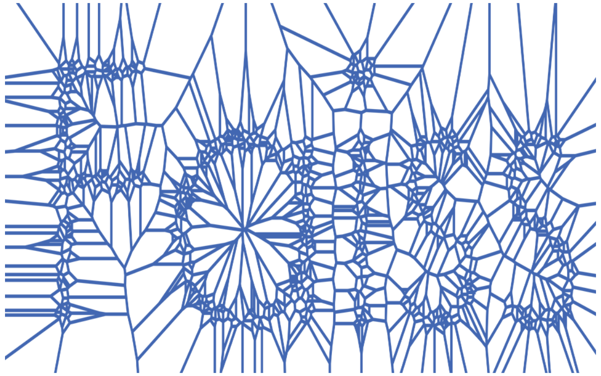
- Balancing Personalization vs Other Objectives
- Balancing Popularity vs Novelty
- Temporal effects
- Cold start problems

Libraries

- [Python Scikit-Learn – SVD, PCA etc]
- [LightFM]
- [TensorRec]
- [FAISS (Facebook AI Similarity Search)]
- [NMSLib (Non-Metric Space Library)]

Tutorial

- [Building a Recommendation System in TensorFlow: Overview]



Q&A

MovieLens (100k) Dataset

DEMO

http://bit.ly/TFDLM_recommender

Github

Thanks!

Questions?

@kuanhoong

https://www.linkedin.com/in/kuanhoong

kuanhoong@gmail.com

Group Photo