Lecture 5: Recommendation System

https://github.com/kaopanboonyuen/SC310005_ArtificialIntelligence_2025s1

Teerapong Panboonyuen
https://kaopanboonyuen.github.io

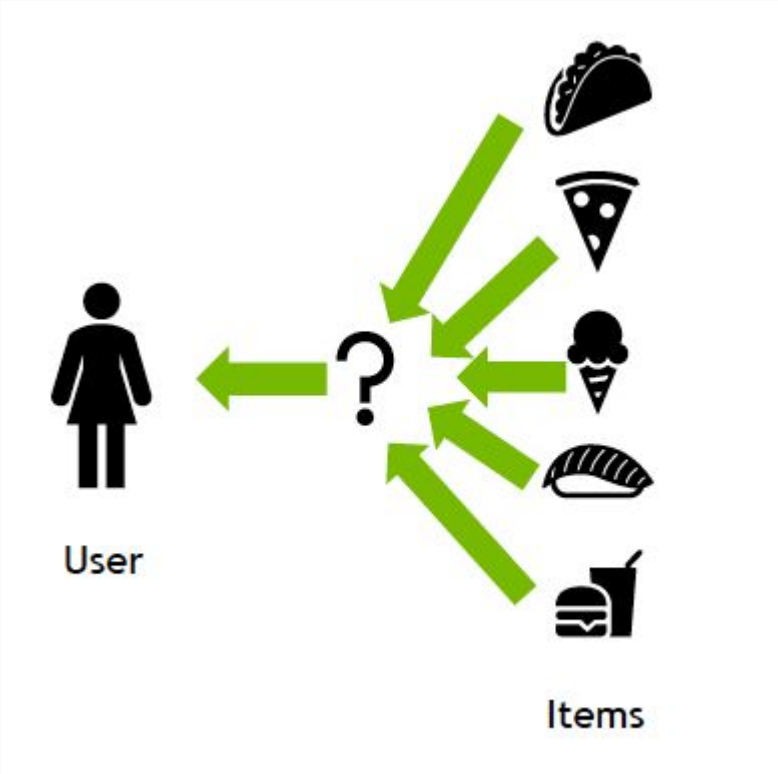
🎯 Introduction to Recommender Systems

- A **Recommender System** helps users discover products they might like by predicting preferences.
- Common in: Netflix, YouTube, Amazon, Spotify.
- Two main types:
  - **Content-Based Filtering**
  - **Collaborative Filtering**
- Hybrid models combine both!

🧭 Types of Recommender Systems

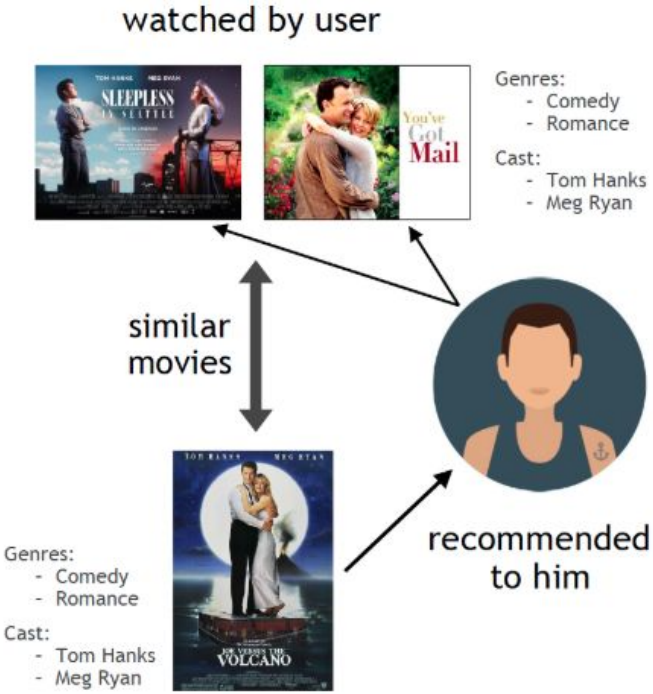
Type	Description	Example
Content-Based	Recommend items similar to what user liked before	"You watched Sci-Fi, here's another Sci-Fi!"
Collaborative Filtering	Recommend based on what similar users liked	"People similar to you loved this item!"
Hybrid	Combine both	Netflix, Spotify

https://www.nvidia.com/en-us/glossary/recommendation-system/

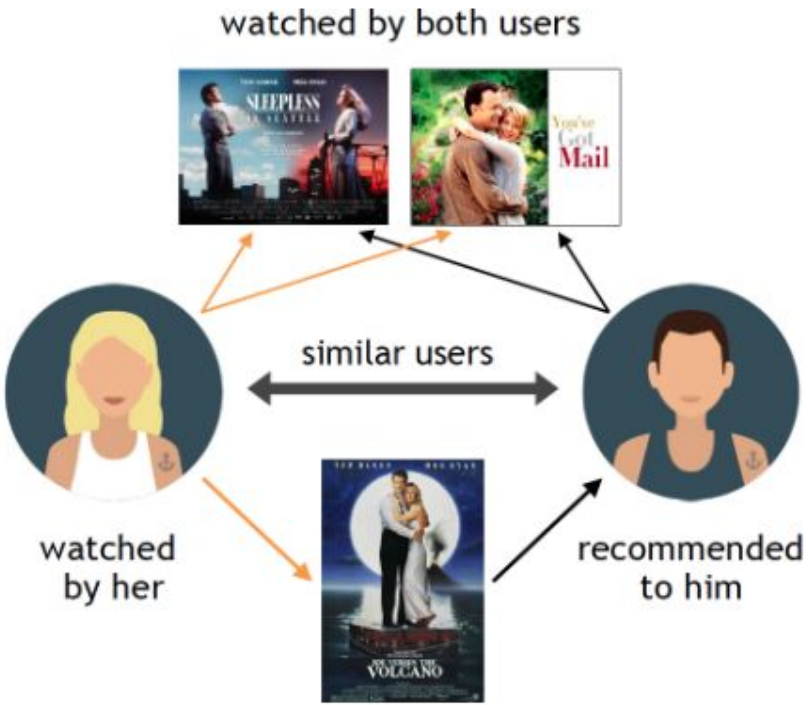


https://www.nvidia.com/en-us/glossary/recommendation-system/

Content-based Filtering

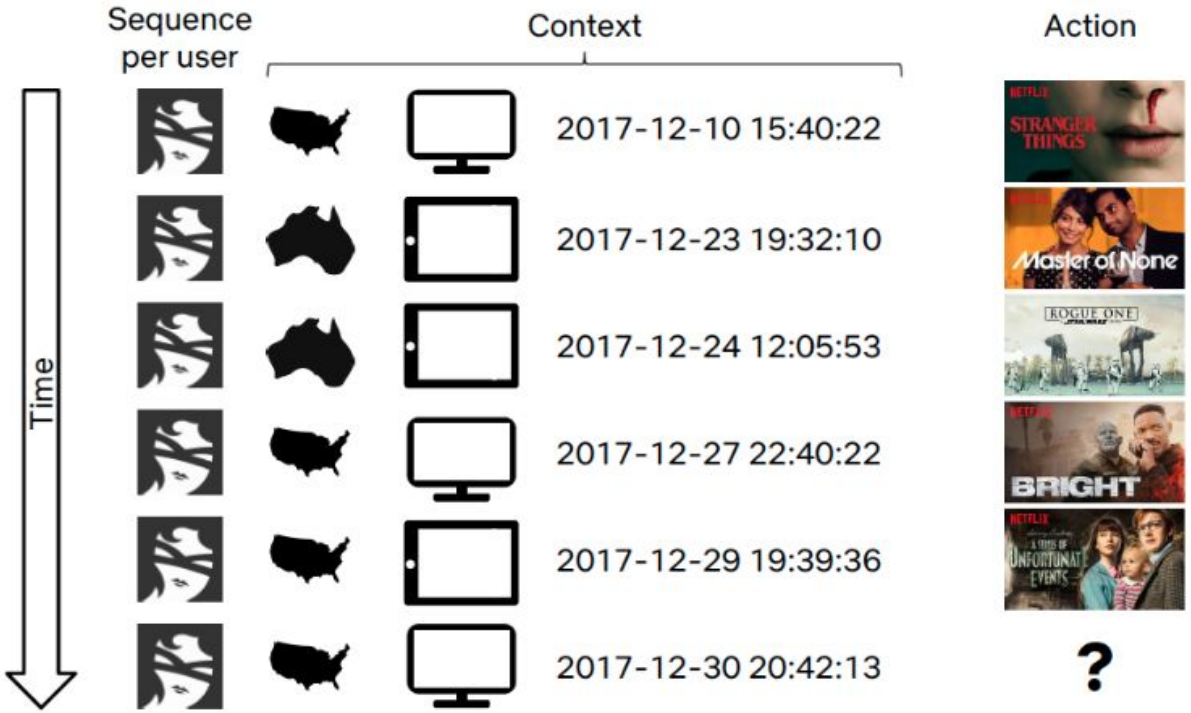


Collaborative Filtering

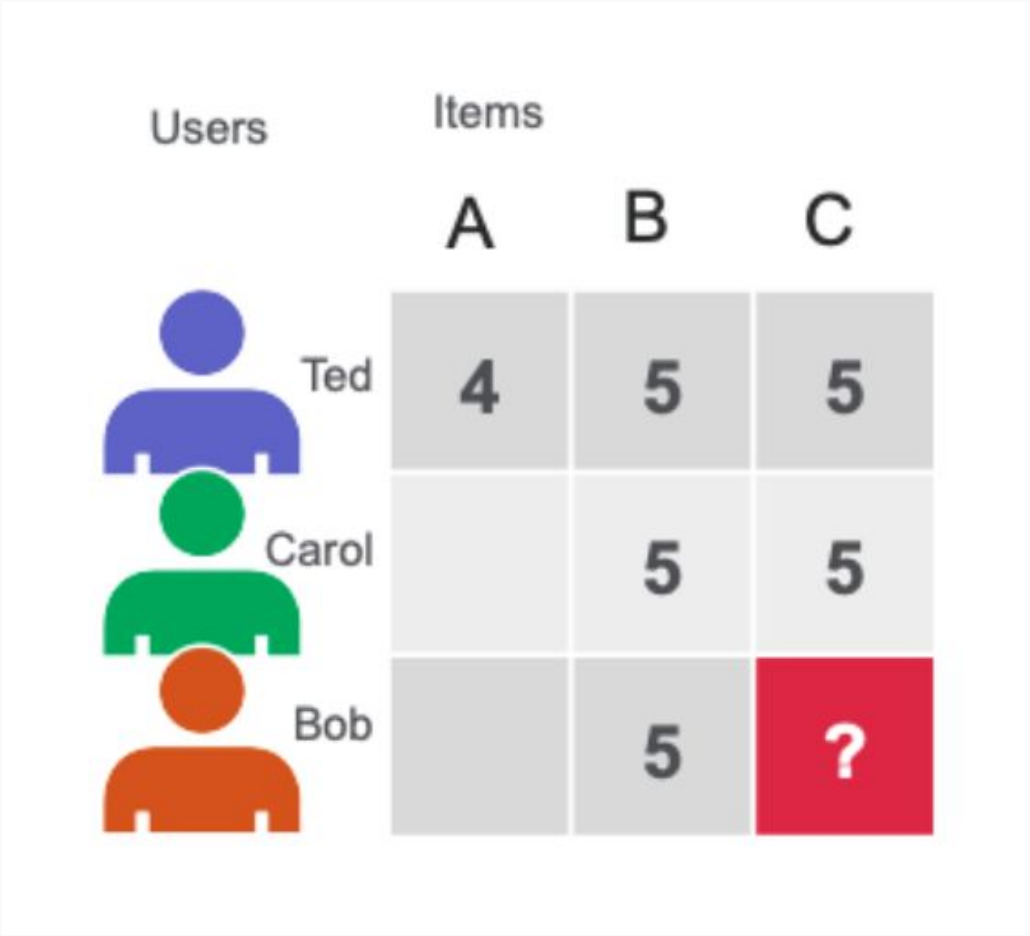


https://www.nvidia.com/en-us/glossary/recommendation-system/

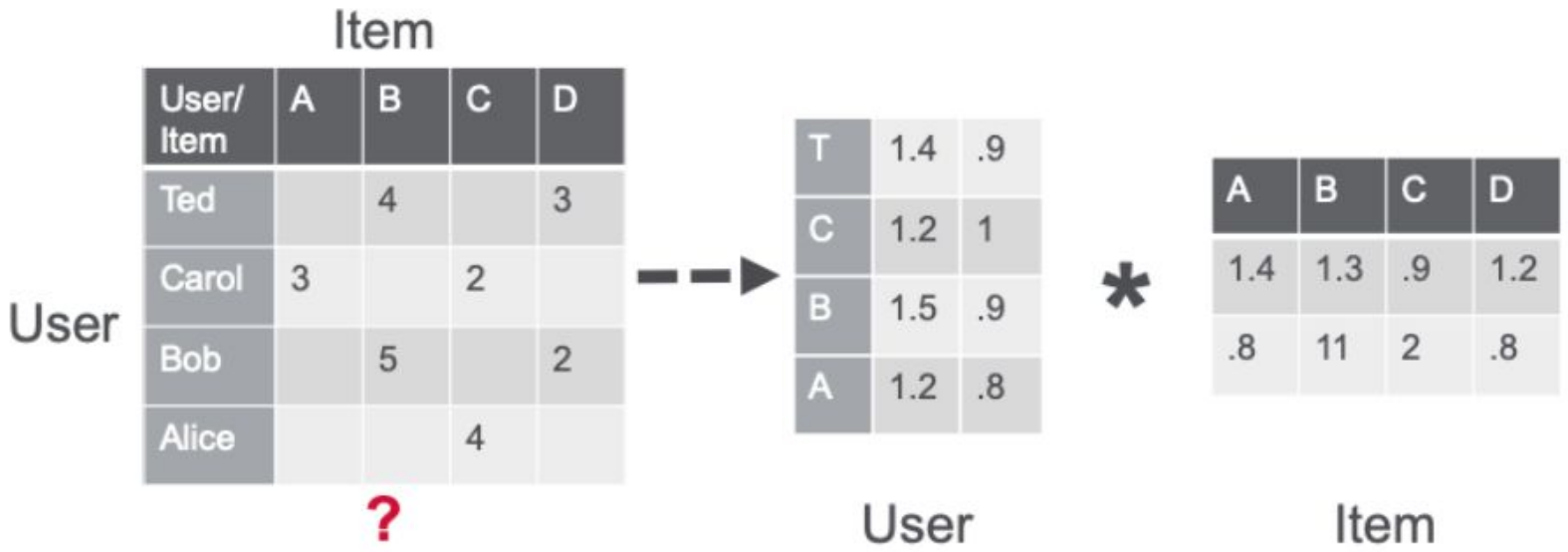
Contextual sequence data



https://www.nvidia.com/en-us/glossary/recommendation-system/

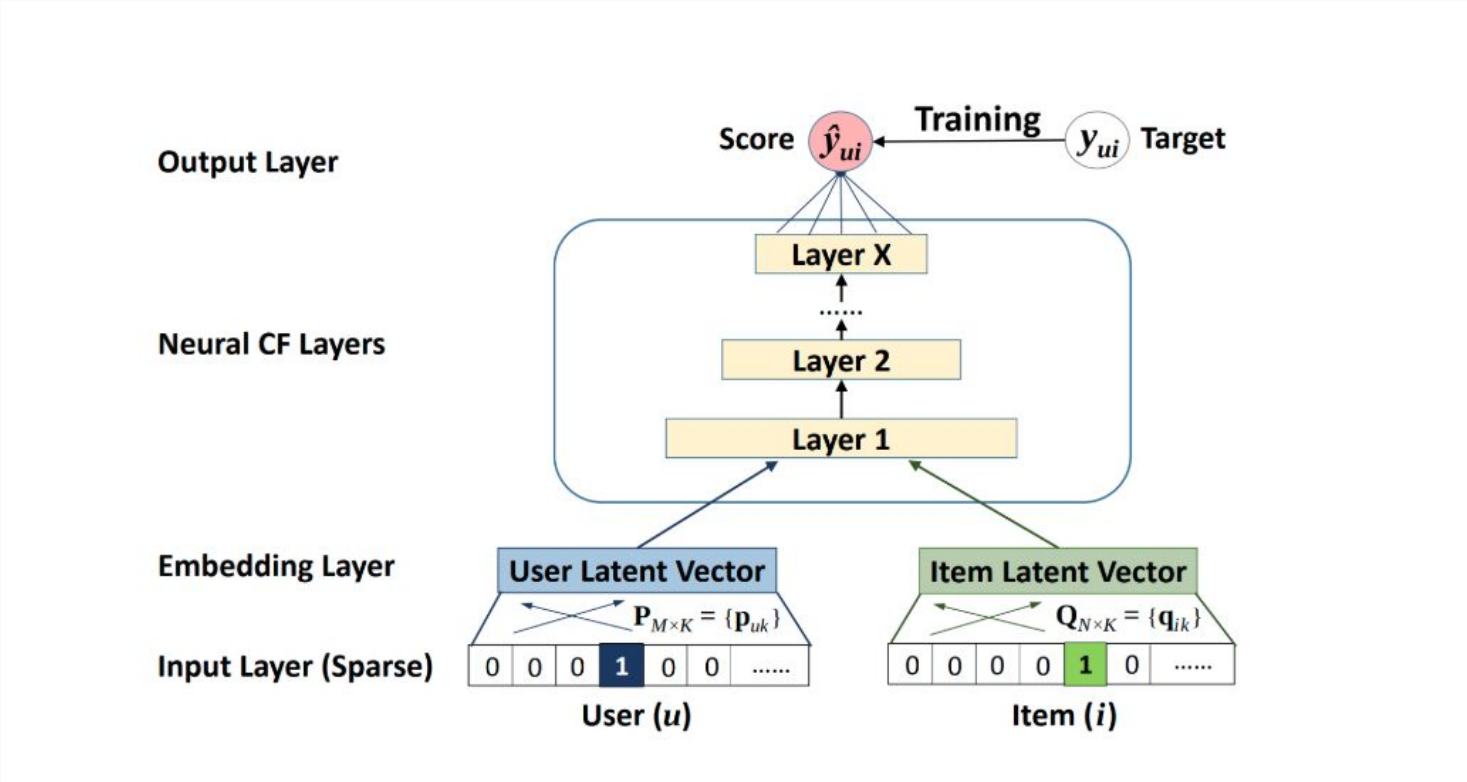


https://www.nvidia.com/en-us/glossary/recommendation-system/



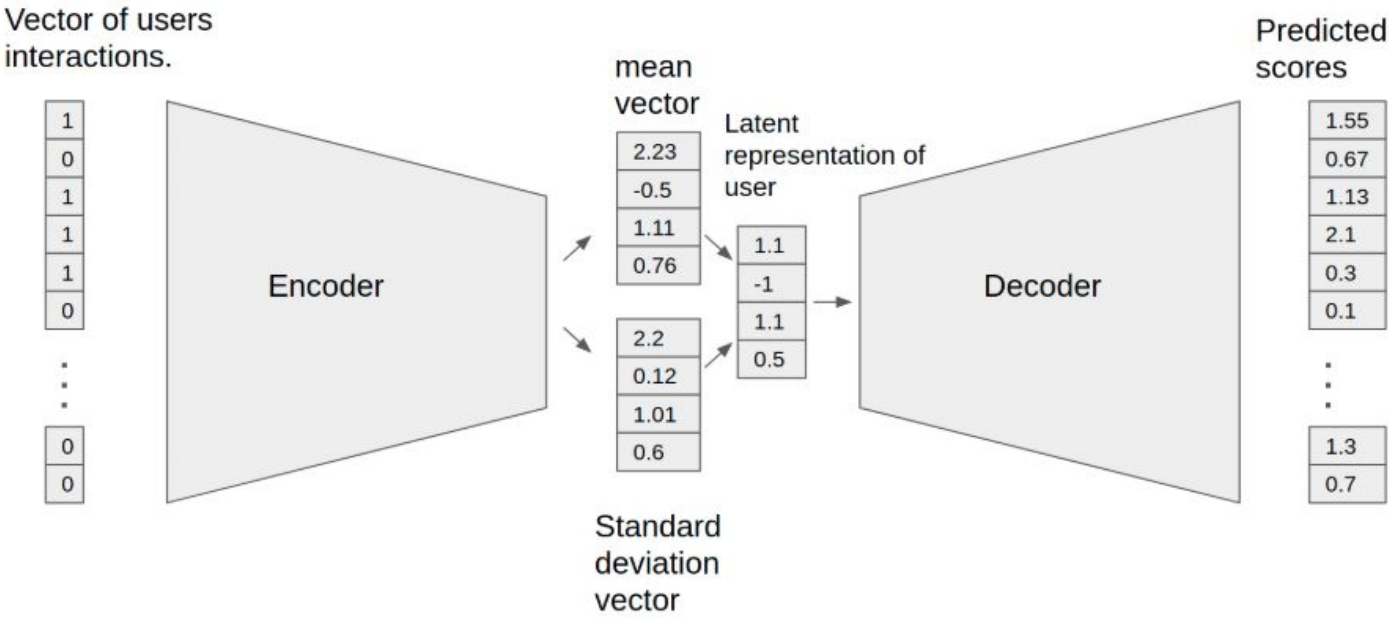
https://www.nvidia.com/en-us/glossary/recommendation-system/

Neural Collaborative Filtering



https://www.nvidia.com/en-us/glossary/recommendation-system/

Variational Autoencoder for Collaborative Filtering

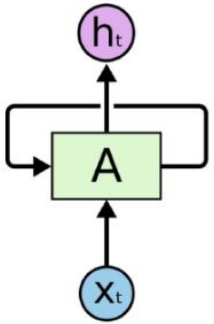


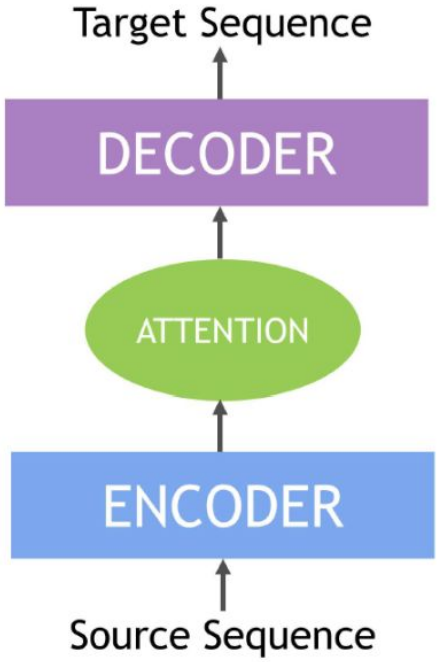
https://www.nvidia.com/en-us/glossary/recommendation-system/

Contextual Sequence Learning

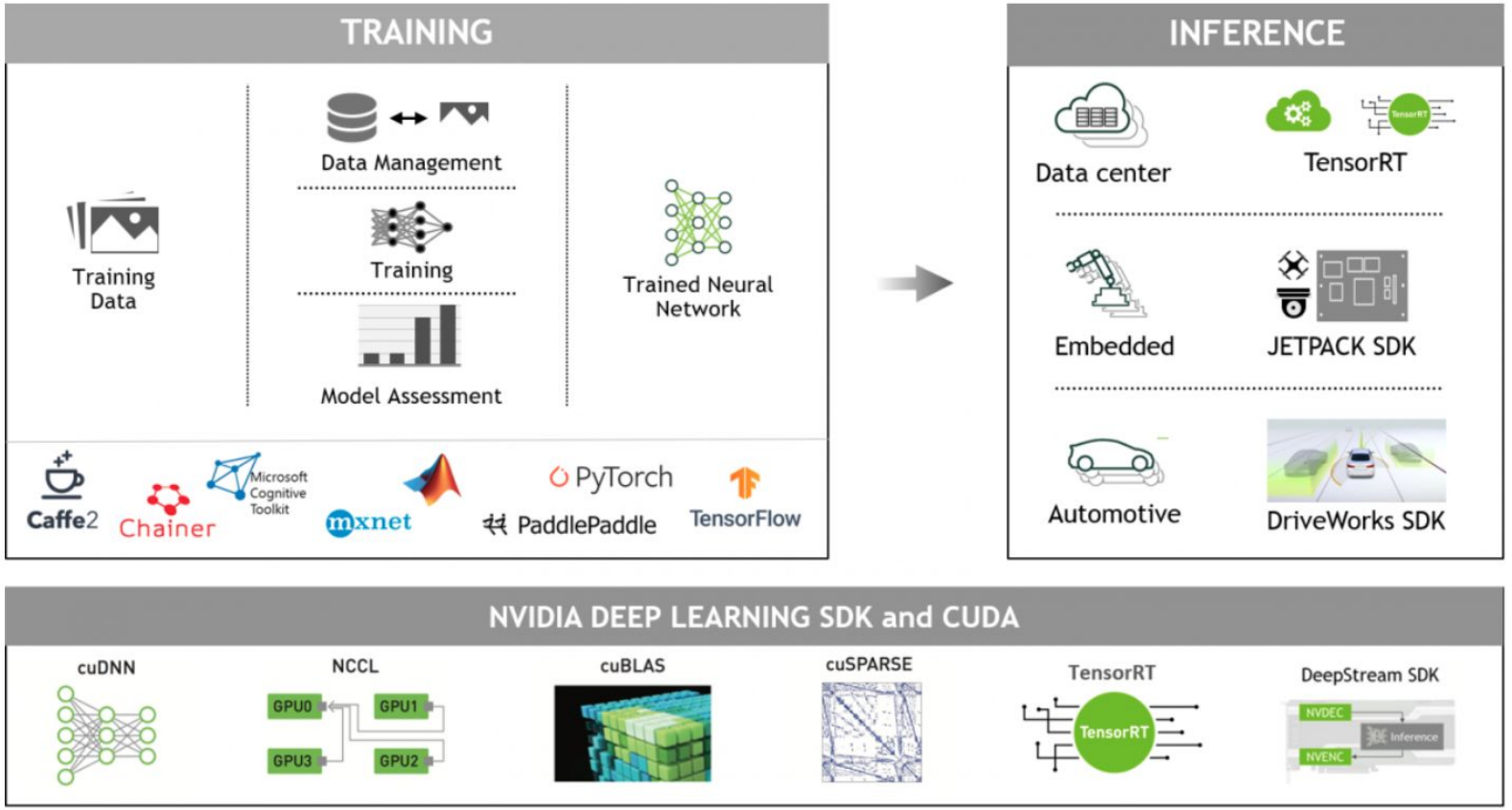
NMT Components

- Encoder
  - Embedding Layer
  - RNN cells
- Decoder
  - RNN cells
  - Embedding Layer
- Attention Layer

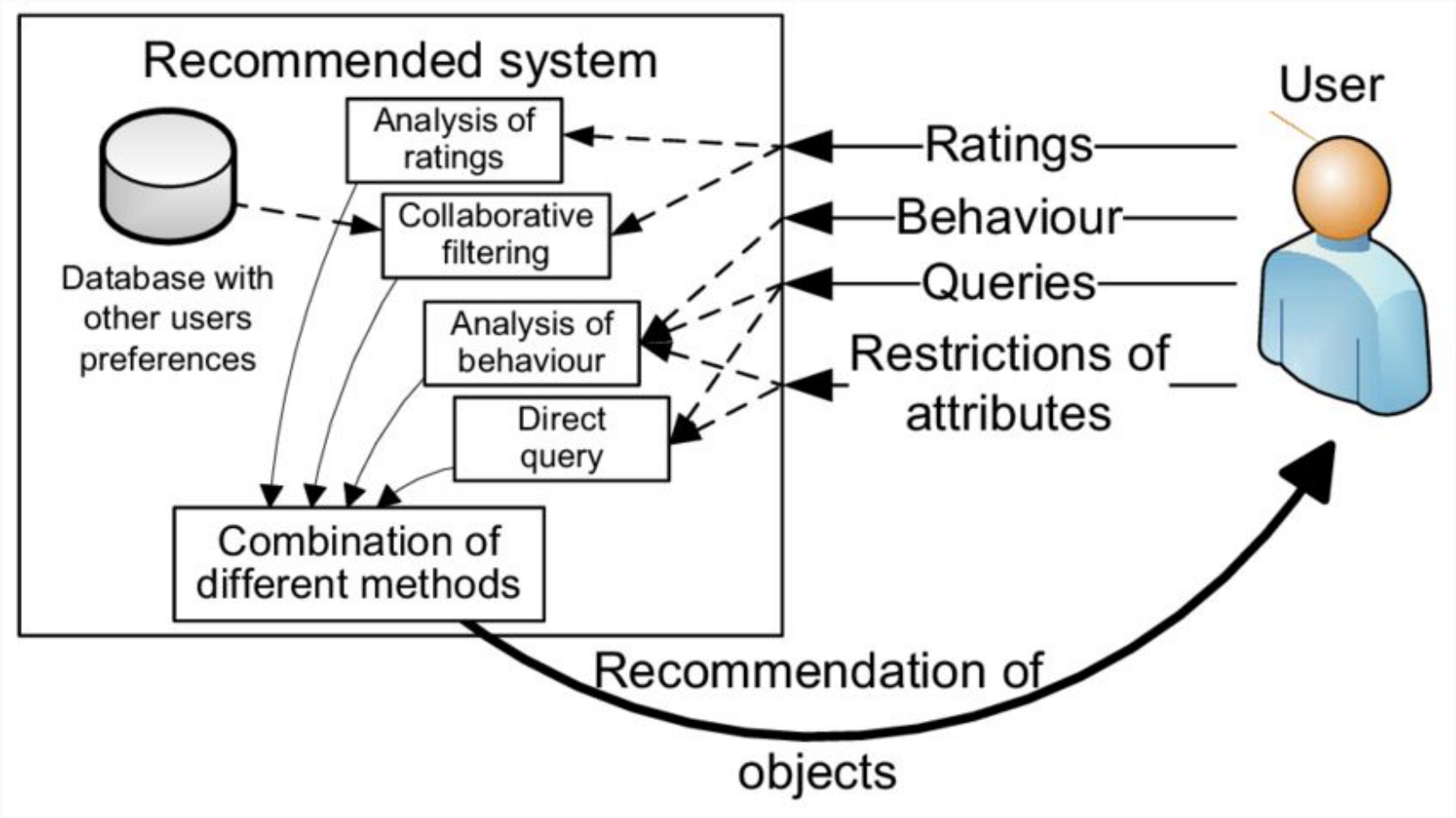




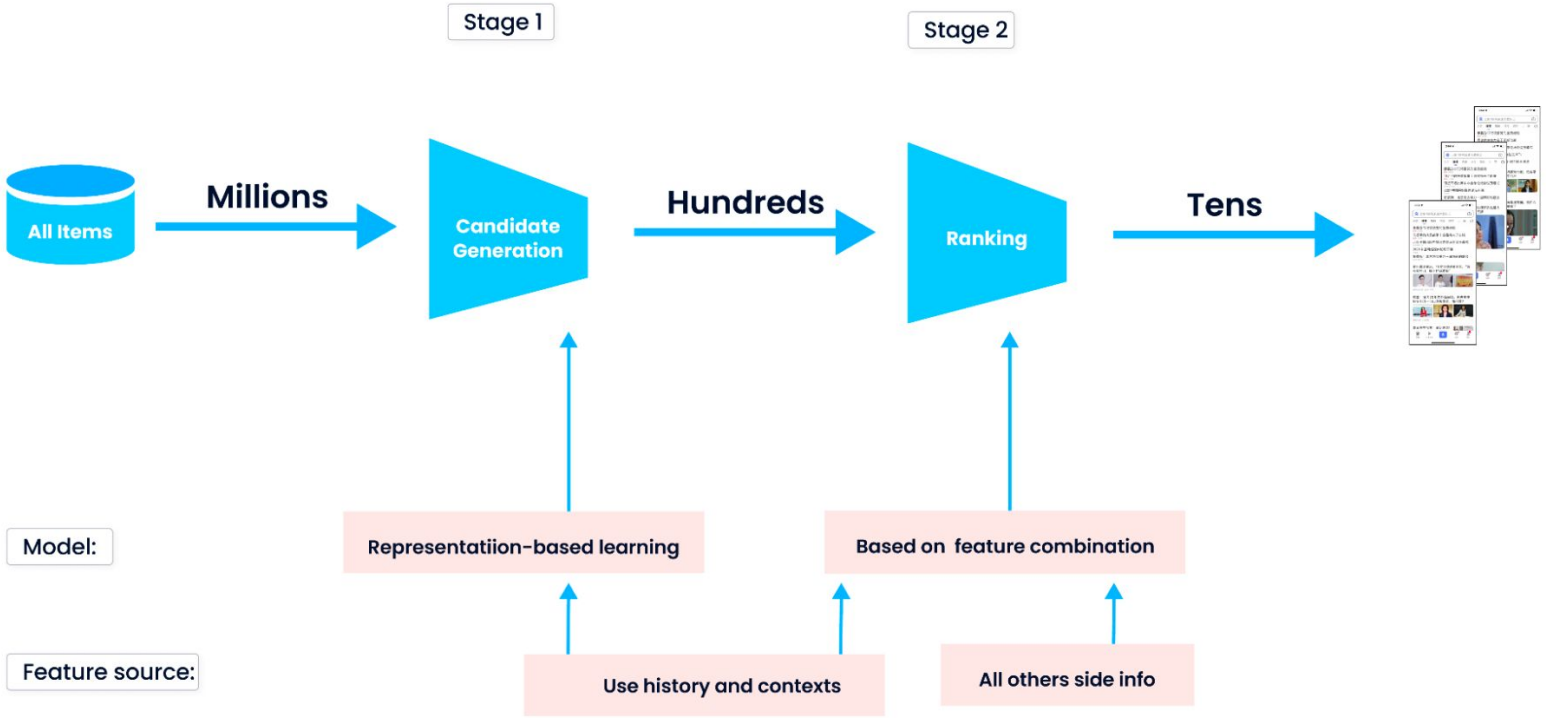
https://www.nvidia.com/en-us/glossary/recommendation-system/



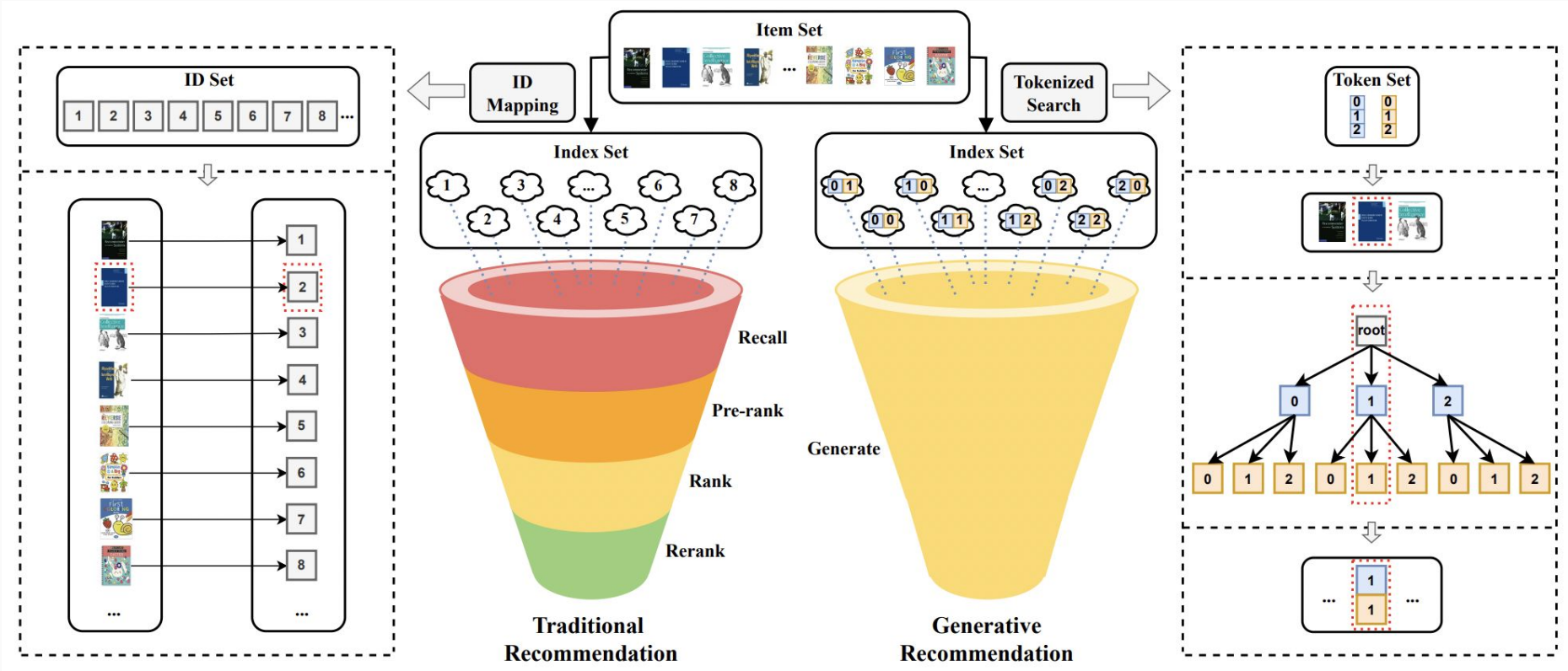
https://neptune.ai/blog/recommender-systems-lessons-from-building-and-deployment



https://neptune.ai/blog/recommender-systems-lessons-from-building-and-deployment



https://llmrecsys.github.io/



https://aman.ai/recsys/LLM/

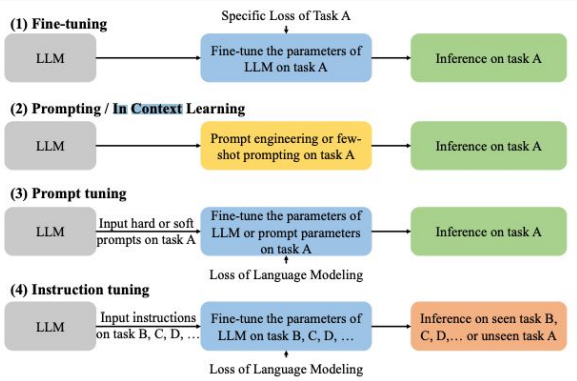


Figure 3: Detailed explanation of five different training (domain adaption) manners for LLM-based recommendations.

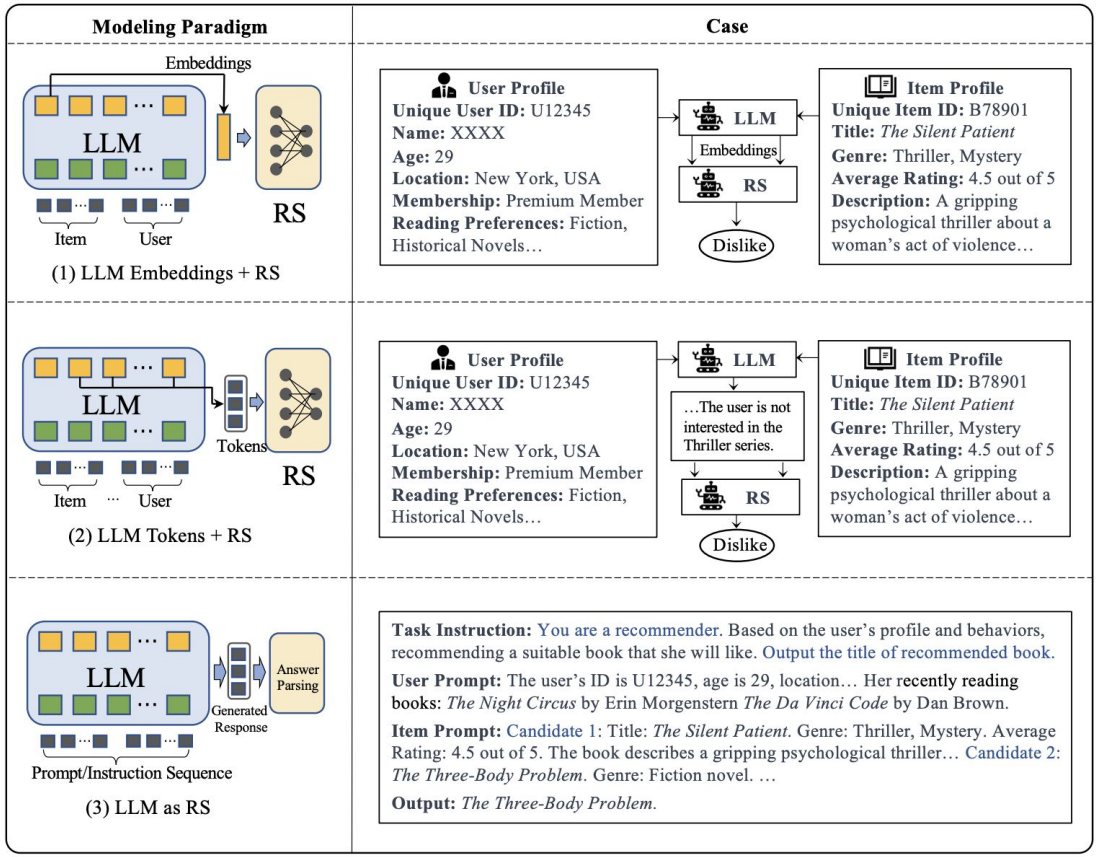
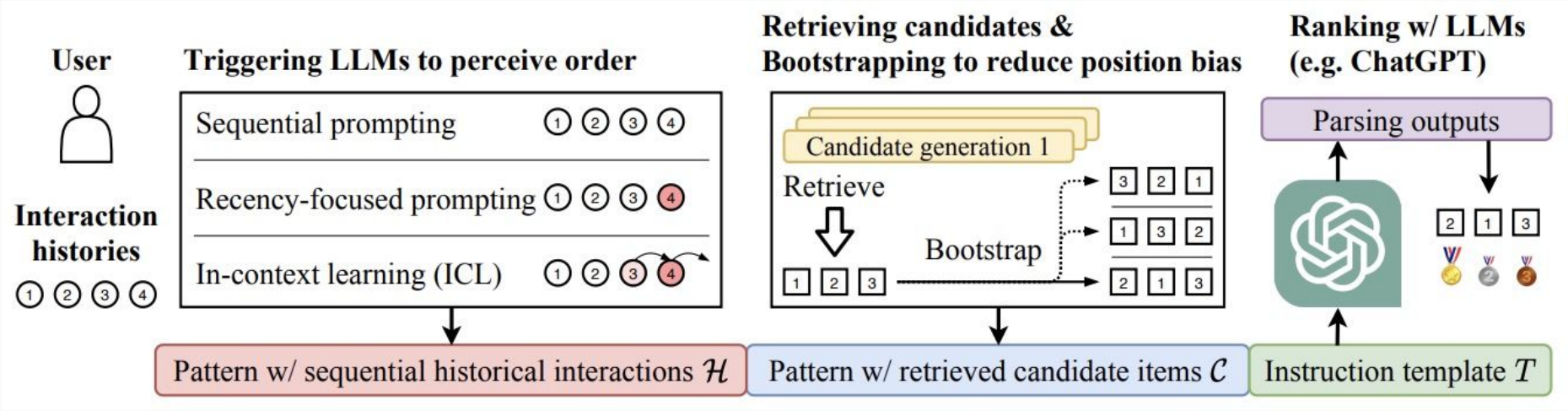


Figure 1: Three modeling paradigms of the research for large language models on recommendation systems.

https://aman.ai/recsys/LLM/



🧠 Content-Based Filtering (CBF)

- Build a profile for each user and item using **features** (e.g., genre, price, brand).
- Measure **similarity** between items.

🔢 Math Example: Cosine Similarity

Given two item vectors:

Item A = [1, 1, 0],

Item B = [1, 0, 1]

$$\text{Cosine}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = 0.5$$

✅ High cosine value = more similar.

🤝 Collaborative Filtering (CF)

Two main approaches:

- **User-Based**: "Find users like me."
- **Item-Based**: "Find items similar to this one."

📘 Mini Example (User-Based CF):

	Book A	Book B	Book C
Alice	5	3	?
Bob	5	2	1

Find similarity between Alice and Bob. Predict rating for Book C using **weighted average**.

🔢 Similarity Measures in CF

1. Cosine Similarity

Same as in content-based:

$$\text{Cosine}(U_1, U_2) = \frac{U_1 \cdot U_2}{\|U_1\| \cdot \|U_2\|}$$

2. Jaccard Similarity (for binary interactions)

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Use when we care only whether user clicked/liked, not how much.

❄️ Cold Start Problem

- New item: No user interactions.
- New user: No prior history.
- Solution:
  - Content-based filtering helps cold-start items.
  - Hybrid systems also help.

🔬 Matrix Factorization (SVD)

- Represent large user-item matrix using **latent factors**.

Example:

$$R_{m \times n} \approx U_{m \times k} \cdot \Sigma_{k \times k} \cdot V^T_{k \times n}$$

Where:

- R: Original ratings
- U, V: User/item embeddings
- k: Latent dimension

Predict missing values using:

$$\hat{R}_{i,j} = U_i^T \cdot V_j$$

🧪 Evaluation Metrics

Metric	Description
RMSE / MAE	How close are predicted ratings to real ones?
Precision@k / Recall@k	How many top-k recommendations were relevant?
Hit Rate	Whether the correct item was ever recommended
Coverage	% of items recommended at least once

🔗 Content-Based vs Collaborative Filtering

Feature	Content-Based	Collaborative
Needs user ratings	❌	✅
Needs item features	✅	❌
Cold start user	❌	✅
Cold start item	✅	❌
Personalization	✅	✅

🔄 Hybrid Recommender System

- Combine content and collaborative scores:

$$\text{Final Score} = \alpha \cdot \text{CF Score} + (1 - \alpha) \cdot \text{CBF Score}$$

- Choose α (e.g., 0.5) depending on the dataset.

🧠 Main Techniques in Recommender Systems

1. **Content-Based Filtering (CBF)**

   Recommends items similar to those the user liked — based on item features.

2. **Collaborative Filtering (CF)**

   Recommends items that similar users liked — based on user-item interactions.

3. **Matrix Factorization (SVD)**

   Factorizes the user-item matrix into lower-dimensional latent space to discover hidden patterns.

4. **Hybrid Approach**

   Combines multiple techniques (e.g., CBF + CF) to improve recommendations.

📐 Similarity Metrics

1. Cosine Similarity

Formula:

$$\text{cosine\_sim}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

Example:

Item	User A	User B
Book1	5	3
Book2	0	4
Book3	2	0

Example:

Item	User A	User B
Book1	5	3
Book2	0	4
Book3	2	0

Vectors:

- A = [5, 0, 2]
- B = [3, 4, 0]

$$\text{cosine\_sim}(A, B) = \frac{(5 \times 3 + 0 \times 4 + 2 \times 0)}{\sqrt{5^2 + 0^2 + 2^2} \times \sqrt{3^2 + 4^2 + 0^2}} = \frac{15}{\sqrt{29} \times 5} \approx 0.557$$

2. Jaccard Similarity

Used when we have binary data (liked or not).

Formula:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Example:

Item	A Liked?	B Liked?
Book1	1	1
Book2	1	0
Book3	0	1
Book4	1	1

Item	A Liked?	B Liked?
Book1	1	1
Book2	1	0
Book3	0	1
Book4	1	1

- A = {Book1, Book2, Book4}
- B = {Book1, Book3, Book4}
- Intersection = 2 (Book1, Book4)
- Union = 4

$$\text{Jaccard} = \frac{2}{4} = 0.5$$

👥 User-Based Collaborative Filtering

Goal: Recommend items liked by similar users.

Example matrix:

	Book A	Book B	Book C
U1	5	0	3
U2	4	1	2
U3	?	5	0

- Use cosine similarity between U3 and others.
- Pick top-k similar users.
- Predict U3's rating for Book A based on weighted average of similar users.

📦 Item-Based Collaborative Filtering

Goal: Recommend items similar to items the user already likes.

How:

- Compare items (rows instead of columns)
- Recommend items similar to what user rated high

Example:

User likes Book A → find items similar to Book A using cosine or Jaccard → recommend those.

📉 Matrix Factorization (SVD)

Step-by-step:

Given a matrix:

	Item1	Item2	Item3
U1	5	3	0
U2	4	0	2

Factor into:

$$\text{Rating} \approx U \times \Sigma \times V^T$$

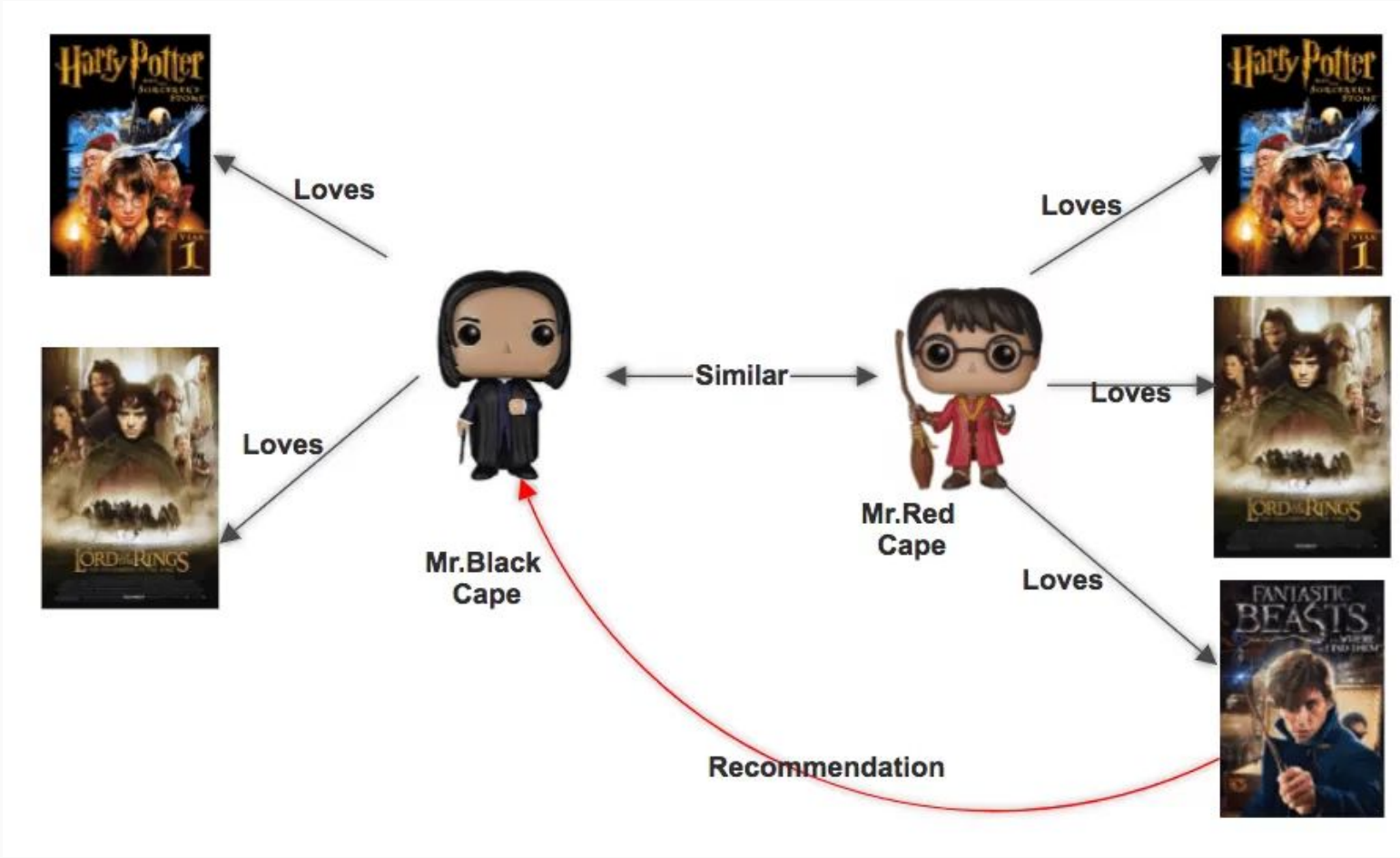
Where:

- U = User latent features
- Σ = Singular values
- V = Item latent features

Prediction:

Multiply latent features to estimate missing ratings.

https://www.mariogerard.com/recommendation-systems-filtering-techniques/



Example: Harry Potter Recommendation Scenario

Step 1: User-Item Ratings Table

User	Harry Potter	Hobbit	Machine Learning	Python Programming
U1	5	4	0	0
U2	0	0	5	4
U3	4	3	?	?

Step 2: Calculate Similarity between Users (User-Based CF)

Goal: Find users similar to U3 to predict missing ratings.

2.1 Cosine Similarity between U3 and U1

Vectors (only on rated books):

- U3 = [4, 3] (Harry Potter, Hobbit)
- U1 = [5, 4]

Calculate:

$$\text{cosine}(U3, U1) = \frac{(4 \times 5) + (3 \times 4)}{\sqrt{4^2 + 3^2} \times \sqrt{5^2 + 4^2}} = \frac{20 + 12}{\sqrt{16 + 9} \times \sqrt{25 + 16}} = \frac{32}{\sqrt{25} \times \sqrt{41}} = \frac{32}{5 \times 6.4} = \frac{32}{32} = 1$$

Interpretation: U3 and U1 are very similar (cosine = 1).

2.2 Cosine Similarity between U3 and U2

Common rated books? None — no overlap, so cosine similarity = 0.

Step 3: Predict U3's Rating for Machine Learning Book using User-Based CF

- Only U2 rated Machine Learning = 5.
- U3 similar to U1 (similarity 1), but U1 didn't rate ML.
- U3 dissimilar to U2 (similarity 0).

Prediction weighted by similarity:

$$\hat{r}_{U3,ML} = \frac{\sum_{\text{users}} \text{sim}(U3, user) \times r_{\text{user},ML}}{\sum_{\text{users}} \text{sim}(U3, user)} = \frac{0 \times 5 + 1 \times 0}{0 + 1} = 0$$

Meaning **no reliable prediction from user-based CF here** for ML book.

Step 4: Use Item-Based CF to Recommend Books Similar to Harry Potter

Calculate cosine similarity between items Harry Potter and others:

- Harry Potter vector: ratings by users = [5, 0, 4]
- Hobbit vector: [4, 0, 3]

$$\text{cosine}(HarryPotter, Hobbit) = \frac{5 \times 4 + 0 + 4 \times 3}{\sqrt{5^2 + 0 + 4^2} \times \sqrt{4^2 + 0 + 3^2}} = \frac{20 + 12}{\sqrt{25 + 16} \times \sqrt{16 + 9}} = \frac{32}{\sqrt{41} \times \sqrt{25}} = \frac{32}{6.4 \times 5} = 1$$

Perfect similarity between Harry Potter and Hobbit.

Step 5: Matrix Factorization Intuition (SVD)

- Represent user-item matrix with latent features, for example:
  - Latent factor 1: "Fantasy interest"
  - Latent factor 2: "Technical interest"

User	Fantasy	Technical
U1	0.9	0.1
U2	0.1	0.9
U3	0.8	0.2

Item	Fantasy	Technical
Harry Potter	1.0	0.0
Hobbit	1.0	0.0
Machine Learning	0.0	1.0
Python Programming	0.0	1.0

Predict rating by dot product:

$$\hat{r}_{U3,ML} = (0.8 \times 0.0) + (0.2 \times 1.0) = 0.2$$

Scaled back to rating scale (e.g., multiply by 5):

$$0.2 \times 5 = 1$$

So SVD predicts U3 might rate Machine Learning book about 1 (low interest).

Summary:

Technique	Harry Potter Recommendation Reasoning	Sample Result
Cosine Similarity (User-Based)	U3 similar to U1 who likes Harry Potter → recommend Harry Potter	Similarity = 1 (max)
Cosine Similarity (Item-Based)	Harry Potter similar to Hobbit (both fantasy)	Similarity = 1
Collaborative Filtering Prediction	Weighted average ratings from similar users	ML book prediction = 0
Matrix Factorization	Latent features explain preferences, predicts ratings on unseen items	ML book predicted rating ≈ 1

🧪 Recommender Systems: A/B Testing

What is A/B Testing?

A/B testing is a method to compare two versions of a recommendation system (A and B) to see which performs better on real users.

💡 Why Use A/B Testing?

- Validate **algorithm changes** before rolling out.
- Understand **user behavior** on different models.
- Make **data-driven decisions**.

🧵 **How It Works:**

1. **Split Users Randomly** into two groups:
    - Group A → **Current model** (control)
    - Group B → **New model** (variant)
2. Show each group their respective recommendations.
3. Measure key metrics over time (e.g., 1-2 weeks).

📊 **Metrics to Compare:**

- **Click-Through Rate (CTR)**
- **Conversion Rate**
- **Time Spent**
- **Average Order Value**
- **User Satisfaction (Surveys or Ratings)**

🧠 **Example:**

Testing a new **Hybrid Recommender** vs the old **Content-Based**:

- Group A gets content-based recs (e.g., Harry Potter → Fantasy books).
- Group B gets hybrid recs (e.g., Harry Potter → Also watched & similar genre).
- After 2 weeks:

  ✅ Group B has 15% higher CTR → **Deploy Hybrid Model!**

https://www.researchgate.net/figure/Flowchart-of-product-recommendation_fig4_365372515