Chapter 10: Recommender Systems

BUSINESS ANALYTICS: UNLOCKING VALUE THROUGH PERSONALIZATION

ASSOC. PROF. NGUYEN BINH MINH PH.D.

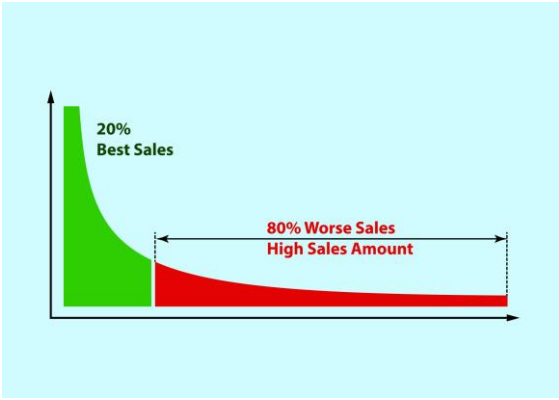
Agenda

- Business Value & The Long Tail
- Content-Based Filtering
- Collaborative Filtering (Implicit Feedback)
- Evaluation Metrics (Offline vs. Online)

The Business Problem: Why do we need Recommender Systems?

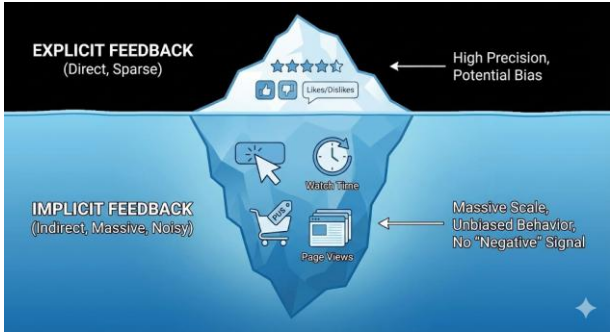
- The Context: Information Overload. Too many movies, products, or songs to choose from.
- The Paradox of Choice: More options → Analysis Paralysis → Customer Churn.
- The Solution: Shift from "Active Search" to "Passive Discovery".
- Key Stats:
  - Netflix: ~80% of content watched comes from recommendations.
  - Amazon: ~35% of revenue comes from cross-selling.

The Long Tail Effect



- The Head (Short Head):
  - Popular items (Blockbusters, Best-sellers).
  - High competition, low margin.
- The Long Tail:
  - Niche items. Individual demand is low, but aggregate volume is massive.
- RecSys Goal:
  - Help users discover items in the "Tail".
  - Increase catalog utilization.

Types of Feedback Data: Explicit vs. Implicit Feedback



1. Explicit Feedback (Direct):
   - Ratings (1-5 stars), Likes/Dislikes.
   - Pros: High precision.
   - Cons: Sparse data (users rarely rate), potential bias.
2. Implicit Feedback (Indirect) - Focus of this lecture:
   - Purchase history, Clicks, Watch time, Page views.
   - Pros: Massive scale, unbiased behavior.
   - Cons: Noisy (accidental clicks), no "negative" signal.

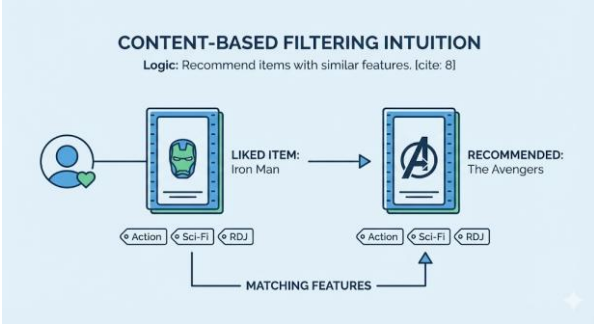
Taxonomy of Approaches: Main Approaches to Recommendation

- Content-Based Filtering:
  - Focus: Item Attributes.
  - Logic: "Show me more of what I liked."
- Collaborative Filtering (CF):
  - Focus: User Interactions.
  - Logic: "Show me what people like me liked."
- Hybrid Systems:Combining both strategies.

Case Study: "MovieStream"

- Scenario: Optimizing a movie streaming platform.
- Data Structure:
  - Users: Alice, Bob, Charlie.
  - Items: Movies (Action, Romance, Sci-Fi).
  - Feedback: Number of minutes watched (Implicit).
- Objective: Predict which movie Alice will watch next.

Content-Based Intuition



- Core Idea: Recommendation based on item similarity.
- Logic:
  - User likes Item A.
  - Item B is very similar to Item A (features). Recommend Item B.
- Example:
  - Liked: Iron Man (Action, Robert Downey Jr, Sci-Fi)
  - .Recommend: The Avengers (Action, RDJ, Sci-Fi).

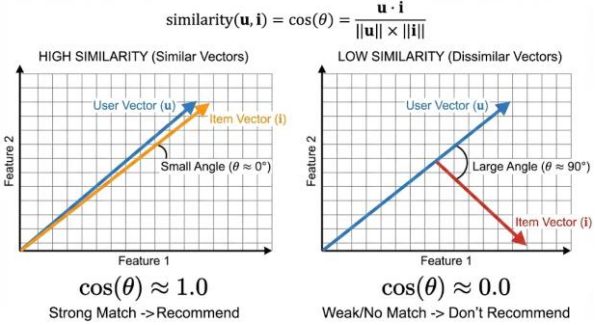
Step 1: Building Item Profiles

- We must convert items into Feature Vectors.
- Structured Data:
  - Genre: [1, 0, 0] (One-hot encoding).
  - Year: Normalized value.
- Unstructured Data (Text):
  - Plot summaries, descriptions.
  - Technique: TF-IDF (Term Frequency-Inverse Document Frequency).
  - Output: A vector representing important keywords.

Step 2: Building User Profiles

- A User Profile is an aggregation of the items they consumed.
- Method: Weighted Average of Item Vectors.
- Example:
  - Alice watched 3 Action movies and 0 Romance movies.
  - Alice's Vector ≈ [Action: High, Romance: Low].
- Dynamic: The profile updates instantly as the user watches more content.

Step 3: Cosine Similarity



- How do we measure if Item X is "close" to User Profile Y?
- The Formula:

$$\text{similarity}(u, i) = \cos(\theta) = \frac{u \cdot i}{||u|| \times ||i||}$$

- Interpretation:
  - 1.0: Perfect match (Vectors align).
  - 0.0: No correlation (Orthogonal).

Content-Based Algorithm Workflow

1. Input: Active User ID.
2. Fetch: User Profile Vector (U).
3. Candidate Generation: Get all Items (I) not yet seen.
4. Score: Calculate Cosine Similarity(U, I) for all candidates.
5. Rank: Sort by score (Descending).
6. Output: Top K items.

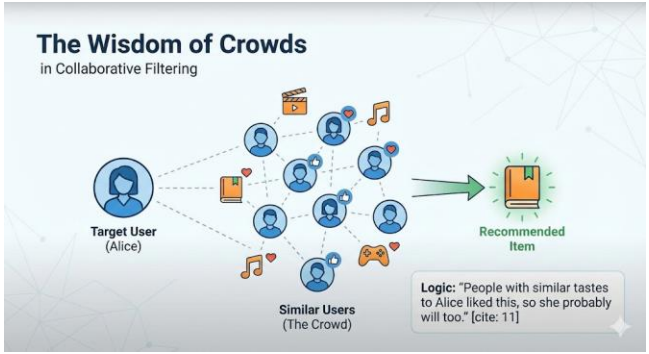
Pros of Content-Based

- User Independence: Works in isolation. Does not need a large community of users.
- Transparency: Easy to explain ("Recommended because you watched X").
- No Item Cold-Start: Can recommend a brand new item immediately if we have its metadata.

Cons of Content-Based

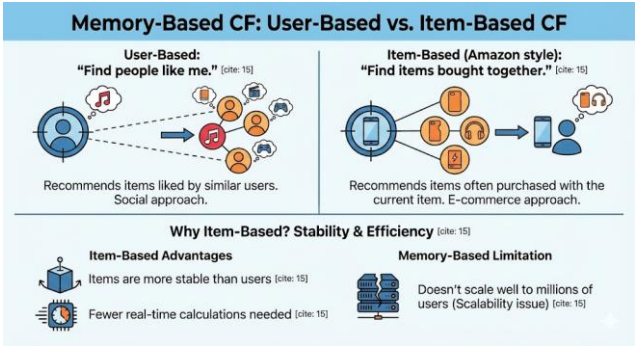
- Overspecialization: The "Filter Bubble" effect.
  - User never discovers new genres.
  - Result: Low serendipity.
- New User Cold-Start: Cannot build a profile for a user with 0 history.
- Feature Engineering: Requires domain knowledge to tag items correctly.

Collaborative Filtering: The "Wisdom of Crowds"



- Core Idea: Patterns in user behavior matter more than item content.
- Assumption: If User A and User B agreed in the past, they will agree in the future.
- Two Main Families:
  - Memory-Based: (User-User, Item-Item).
  - Model-Based: (Matrix Factorization).

Memory-Based CF: User-Based vs. Item-Based CF

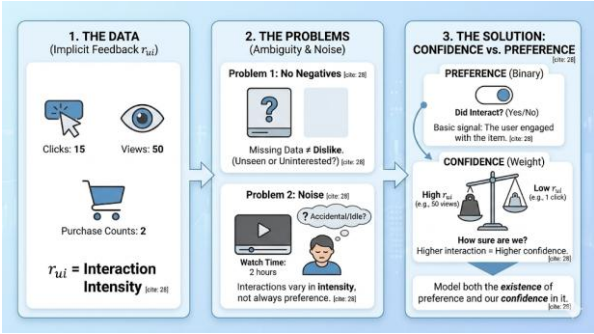


- User-Based: "Find people like me."
- Item-Based (Amazon style): "Find items that are bought together."
  - Why Item-Based? Items are more stable than users; fewer calculations needed in real-time.
- Limitation: Doesn't scale well to millions of users.

The User-Item Interaction Matrix

- Rows: Users (M).
- Columns: Items (N).
- Values: Ratings or Interactions.
- The Challenge: Sparsity.
  - A typical matrix is >99% empty.
  - Goal: Predict the missing values.

Dealing with Implicit Data



- The Data: Clicks, Views, Purchase counts (r_{ui}).
- The Problems:
  1. No Negatives: Missing data ≠ Dislike.
  2. Noise: Interactions vary in intensity.
- The Solution: Model Confidence vs. Preference.

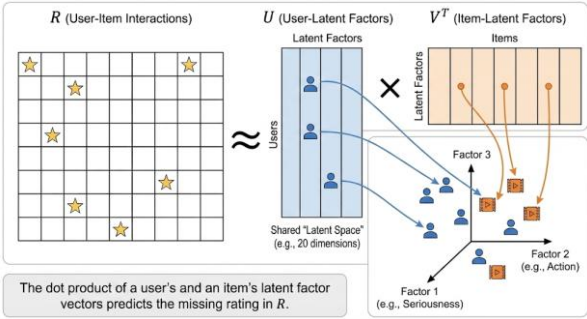
Modeling Implicit Feedback: Preference & Confidence Model

- Binary Preference (p_{ui}):
  - 1 if user interacted ($r_{ui} > 0$).
  - 0 if no interaction.
- Confidence (c_{ui}):

$$c_{ui} = 1 + \alpha \times r_{ui}$$

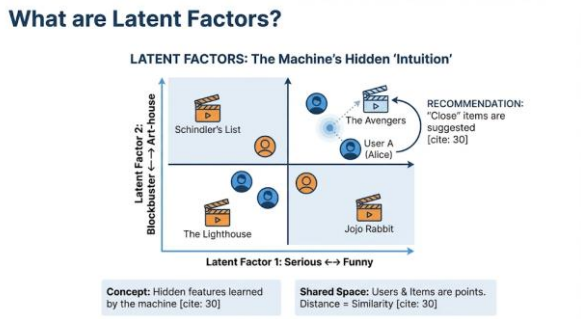
- Meaning: The more you watch, the more confident we are that you like it.

Matrix Factorization (MF)



- Concept: Decompose the huge sparse matrix (R) into two smaller, dense matrices (U and V).
- Equation: $R \approx U \times V^T$
- Latent Factors: Both Users and Items are mapped to a shared "Latent Space" (e.g., 20 dimensions).

What are Latent Factors?



- Hidden features learned by the machine.
- Example (Movies):
  - Dim 1: Serious vs. Funny.
  - Dim 2: Blockbuster vs. Art-house.
- Users and Items are placed as points in this space.
- Recommendation: Find items "close" to the user in this space.

Predicting the Score

- To predict User u's interest in Item i:

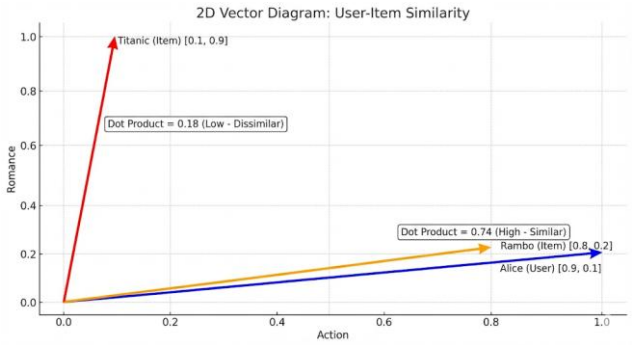
$$\widehat{y_{ui}} = x_u^T \cdot y_i$$

- (Dot product of User Vector and Item Vector).
- High Score: Vectors point in the same direction → Strong Recommendation.

Training with ALS (Alternating Least Squares)

- We need to find optimal U and V matrices.
- Why ALS? Standard Gradient Descent is hard with Implicit Data (too many zeros).
- Process:
  - Fix User vectors, solve for Item vectors.
  - Fix Item vectors, solve for User vectors.
  - Repeat until error is minimized.
- Key Benefit: Highly parallelizable (Big Data friendly).

Simplified Calculation Example



- User Alice: [0.9, 0.1] (Likes Action).
- Item Rambo: [0.8, 0.2] (Is Action).
- Item Titanic: [0.1, 0.9] (Is Romance).
- Dot Products:
  - Alice · Rambo = **0.74** (High).
  - Alice · Titanic = **0.18** (Low).
- Result: System recommends Rambo.

Advantages of CF

- Serendipity: Helps users discover items they wouldn't have found otherwise.
- Content Agnostic: Works for any domain (Movies, Tools, Fashion) without understanding the product features.
- Community Learning: The system gets smarter as more users join.

Disadvantages of CF

- Cold-Start Problem (Severe):
  - New Item: No one has clicked yet → Vector is undefined.
  - New User: No history → Cannot find neighbors.
- Popularity Bias: Tendency to recommend what is already popular (The "Harry Potter" problem).

Content-Based vs. Collaborative Filtering

Feature	Content-Based	Collaborative Filtering
Data Needed	Item Features	User Interactions
New Item	Easy to recommend	Hard (Cold Start)
New User	Hard	Hard
Diversity	Low (More of the same)	High (Discovery)
Common Algorithm	TF-IDF, Cosine	Matrix Factorization (ALS)

How to Evaluate a Recommender?

- The Challenge: Missing ground truth (We don't know if a user likes what they haven't seen).
- Two Stages:
  - Offline Evaluation: Historical data (Train/Test split).
  - Online Evaluation: Live A/B Testing.

Metric Categories

- Prediction Accuracy (Rating): RMSE, MAE.
  - Question: "How many stars will they give?"
- Ranking Accuracy (Top-N): Precision, Recall, NDCG.
  - Question: "Is the relevant item in the Top 10?"
  - Crucial for Implicit Feedback.

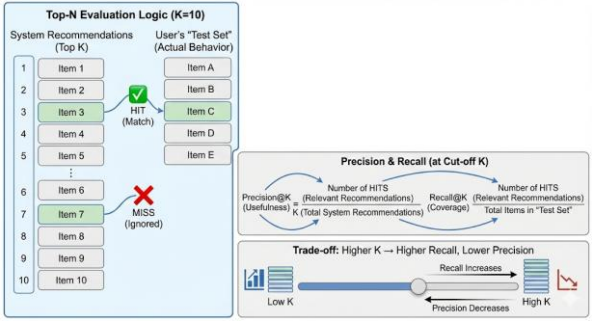
Prediction Metric: RMSE

- Root Mean Squared Error:

$$RMSE = \sqrt{\text{Avg}\big((\text{Actual} - \text{Predicted})^2\big)}$$

- Usage: Explicit Ratings (1–5 stars).
- Limitation: Good RMSE ≠ Good Experience.
- Users care about the Ranking Order, not the exact decimal rating.

Top-N Evaluation Logic

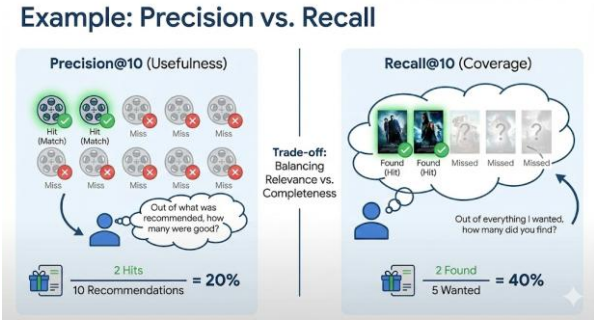


- Scenario: System generates a list of K items (e.g., Top 10).
- Check against "Test Set" (Items user actually consumed).
- Hit: Recommendation matches user behavior.
- Miss: Recommendation ignored.

Precision & Recall

- K = Cut-off rank (e.g., 10).
- Precision@K: Usefulness.
  - % of recommendations that are relevant.
- Recall@K: Coverage.
  - % of relevant items found by the system.
- Trade-off: Higher K → Higher Recall, Lower Precision.

Example: Precision vs. Recall



- User Logic: Wants 5 specific movies.
- System: Recommends 10 movies.
- Result: 2 matches (Hits).
  - Calculations:Precision@10 = $2/10 = 20\%$.
  - Recall@10 = $2/5 = 40\%$.

Mean Average Precision (MAP)

- Precision ignores Position.
  - Recommending a hit at #1 is better than at #10.
- MAP: Calculates precision at every rank and averages it.
- Rewards systems that put relevant items at the very top.

NDCG (Normalized Discounted Cumulative Gain)

- The Gold Standard for Ranking.
- Concept:
  - Gain: Relevance of the item.
  - Discount: Divide score by $\log(\text{Rank})$.
- Result: A hit at Rank 10 is worth much less than a hit at Rank 1.
- Normalized: 0 to 1 score.

Online Evaluation: A/B Testing

- Offline metrics are just proxies.
- A/B Test Setup:
  - Group A (Control): Existing Algorithm.
  - Group B (Treatment): New Algorithm.
- Measure Business Impact: Do they watch more? Do they buy more?

Business KPIs

- CTR (Click-Through Rate): Engagement.
- CVR (Conversion Rate): Revenue.
- Diversity: Preventing boredom (not showing 10 similar items).
- Novelty: Helping users find unpopular/new items.

Beyond Matrix Factorization

- Deep Learning: Neural Collaborative Filtering (NCF).
- Sequence-Based: RNNs/Transformers (Session-based RecSys).
- Context-Aware: Time, Location, Weather.
- LLMs: Using ChatGPT-like models for explanation and content understanding.

Python Ecosystem

- $Surprise$: Good for explicit ratings (RMSE).
- $Implicit$: Optimized for Implicit Feedback (ALS).
- $TensorFlow\ Recommenders$: Production-scale Deep Learning.
- $LightFM$: Hybrid models.

Key Takeaways

1. RecSys shifts users from Search to Discovery (Long Tail).
2. Content-Based: Good for new items, low serendipity.
3. Collaborative Filtering: Good for discovery, struggles with Cold-Start.
4. Implicit Feedback requires specific handling (Confidence weights).
5. Evaluation must focus on Ranking (NDCG), not just Error (RMSE).

Q&A

- **Reading:** "Matrix Factorization Techniques for Recommender Systems" (Koren et al.).