Wharton
University of Pennsylvania

Decentralized Recommendation System

Matthew Kuo, Laura Li, Vivian Xiao, Megan Yang

March 4, 2025

Lighting presentation: 1. **Time:** 5-7 minutes 2. **Content:** Problem statement & Motivation, technical approach, initial results, next steps
Important:

- write out the optimization formulation mathematically, as well as any algorithm's you've tried. You can't skip this.

- each person on your team should speak.

Motivation

- Users often browse for **similar products** on **different shopping websites**
- Repetitive searching is frustrating – requires manual searches and filtering through endless options
- Current recommendation systems are siloed – most platforms only suggest items based on their own data

Problem Statement

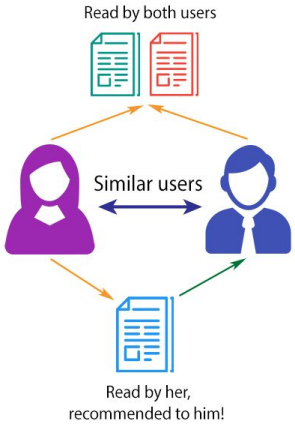
Current recommendation systems are platform-specific and disconnected, forcing users to manually search for alternatives across different shopping websites.

Goal: Develop an online shopping recommendation system that leverages product metadata (e.g., images, descriptions, etc.) to provide seamless, personalized suggestions across platforms, reducing the need for repetitive searches.

Technical Approach

Collaborative Filtering

- **User-based:** finding similar users and suggesting what they like
- **Item-based:** finding similar items based on user interactions



Optimization Function

$$\min_{U,V} \sum_{(i,j)\in\Omega} (R_{ij} - U_i^T V_j)^2 + \lambda(||U||^2 + ||V||^2)$$

where:

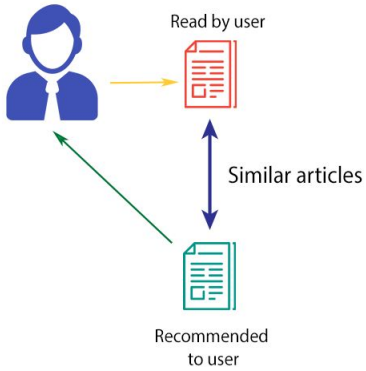
- R_{ij}: the observed rating for user i and item j
- U_i and V_j: user and item vectors
- λ: regularization parameter
- Ω: set of user-item interactions

Implementation Steps

1. Extract user & item vectors from dataset.
2. Compute cosine similarity between the target user and others.
3. Select top-N similar users/items as weights.
4. Predict item scores using weighted preferences.
5. Rank & recommend top items based on scores.

Content-based Filtering

- Analyzes **item features** (e.g., descriptions, image embeddings) and compares them to a user's **past preferences**.
- A user preference vector (v_u) is created by averaging the feature representations of liked items (L_u).
- New items (j) are ranked based on **cosine similarity** to the user preference vector (s_j), with the highest-scoring items recommended.



Mathematical Formulation

$$v_u = \frac{1}{|L_u|} \sum_{i \in L_u} x_i$$

$$\arg\max_{j \notin L_u} s_j \qquad s_j = \frac{v_u \cdot x_j}{\|v_u\| \|x_j\|}$$

Implementation Steps

1. Represent each item as a feature vector (text & image embeddings)
2. Compile a list of all "liked" items for a user
3. Calculate the cosine similarity between the preference vector and all other items.
4. Sort items by similarity.
5. Return top recommendations

Low-rank Matrix Completion

The goal is to complete a partially observed user-item rating matrix using a low-rank factorization approach.

Given a rating matrix $R \in \mathbb{R}^{m \times n}$, where m is the number of items and n is the number of users, the objective is to find two low-rank matrices $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$ such that:

$$\hat{R} = UV^T$$

where k is the latent factor dimension. The optimization problem is formulated as:

$$\min_{U,V} \sum_{(i,j) \in \Omega} (R_{ij} - (UV^T)_{ij})^2 + \lambda(||U||_F^2 + ||V||_F^2)$$

where:

- Ω is the set of observed entries in R
- λ is the regularization term to prevent overfitting
- $||\cdot||_F$ denotes the Frobenius norm

Implementation Steps:

1. Generate ratings matrix and create a mask for known ratings
2. Grid Search to find the optimal rank with minimal validation loss
3. Initialize low-rank matrices U and V using a projection layer
4. Use Adam optimizer for training
5. Compute final predictions for missing values, and recommend items with the highest predicted ratings for each user.

Further Optimizations:

- Instead of pure matrix factorization, train a Neural Network model that incorporates:
  - user embeddings (V), item embeddings (U)
  - A feedforward network for learning nonlinear interactions.
- Factorization Machines (FM) can model high-order interactions between users, items, and features.
- Graph Neural Networks (GNNs) can model user-item interactions dynamically.

Two-Tower

Instead of learning one large joint representation of users and items

- Use one NN (tower) to learn user reps and the other to learn item reps.
- Compare them with a similarity function.

<u>Implementation Steps & Mathematical Formulation</u>

1. Use CLIP to provide initial embeddings.
2. Compute weighted user embeddings:
   a. $$Z_u = \frac{\sum_{i \in R_u} r_{ui} \cdot Z_i}{\sum_{i \in R_u} r_{ui}}$$
3. User embeddings are passed through user tower for transformation:
   a. $$\tilde{Z}_u = f_U(Z_u; \theta_U)$$
4. Item embeddings are fed into item tower:
   b. $$\tilde{Z}_i = f_I(Z_i; \theta_I)$$

Loss function (MSE) ensures that user and item embeddings of items user likes become more similar.

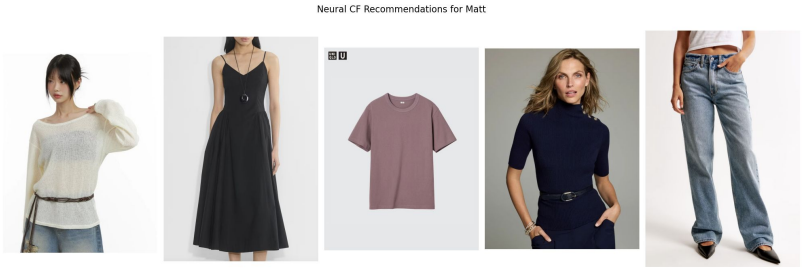
$$S(Z_u, Z_i) = \frac{\tilde{Z}_u \cdot \tilde{Z}_i}{\|\tilde{Z}_u\| \|\tilde{Z}_i\|}$$

$$\mathcal{L} = \frac{1}{N} \sum_{(u,i)} (S(\tilde{Z}_u, \tilde{Z}_i) - y_{ui})^2$$

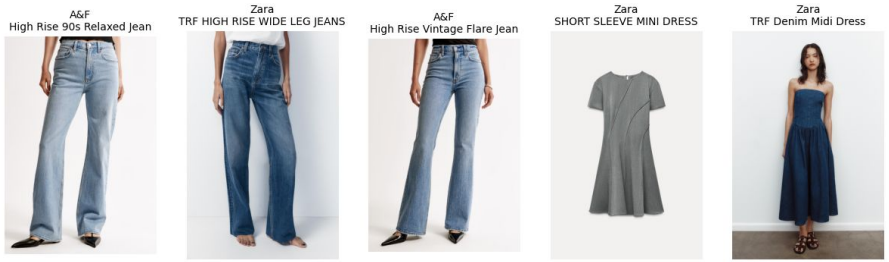
Initial Results

Matthew’s Recommendations

Collaborative Filtering

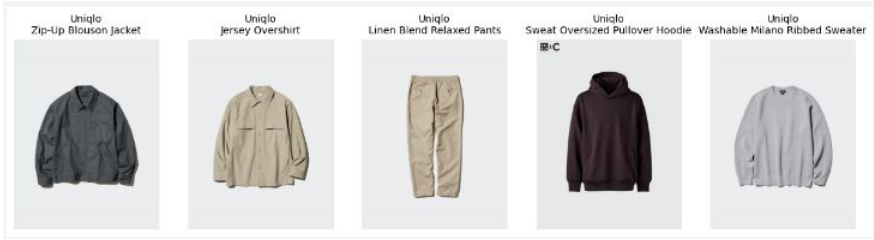


Content Filtering



Low-rank

Two Tower

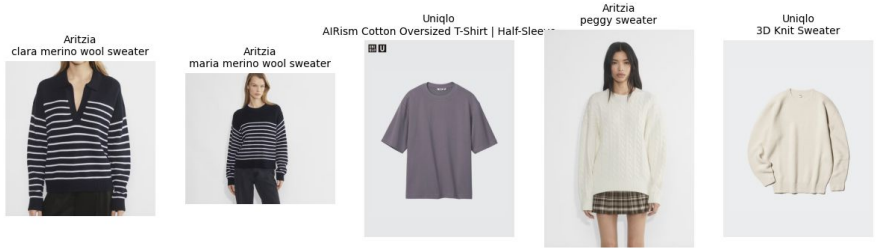


Laura’s Recommendations

Collaborative Filtering

Neural CF Recommendations for Laura

Content Filtering



Low-rank

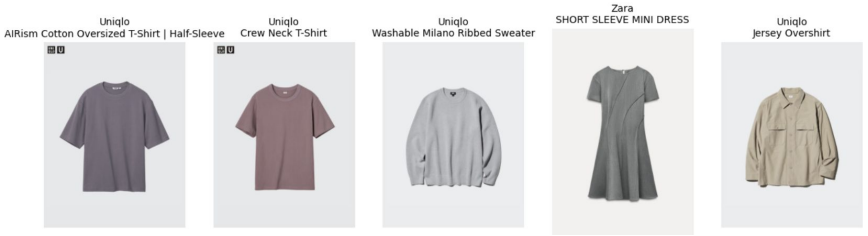
Two Tower

Vivian’s Recommendations

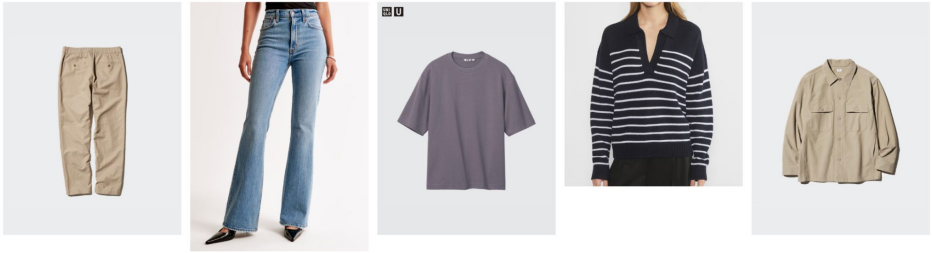
Collaborative Filtering

Neural CF Recommendations for Vivian

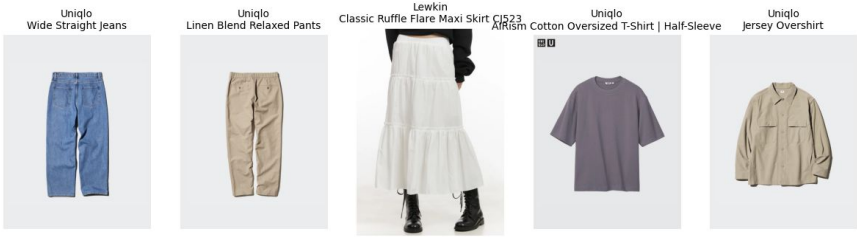
Content Filtering



Low-rank



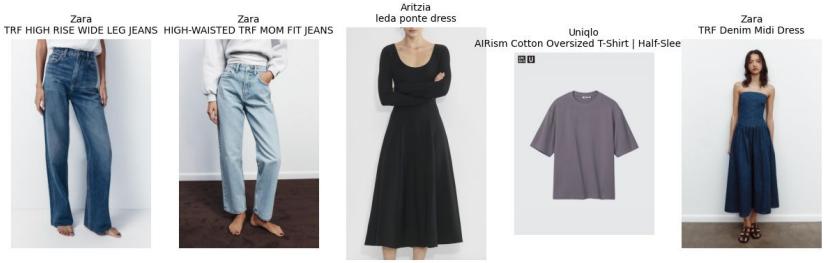
Two Tower



Megan’s Recommendations

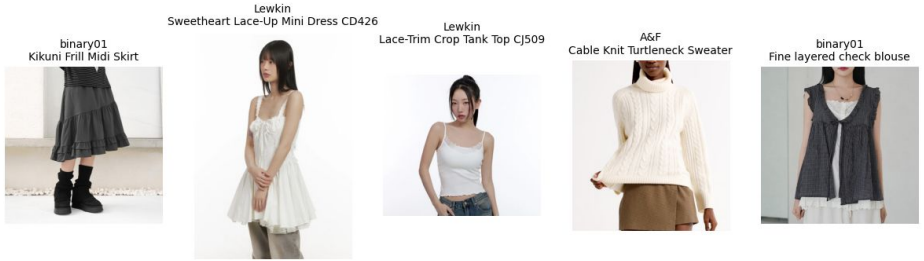
Collaborative Filtering

Content Filtering



Low-rank

Two Tower



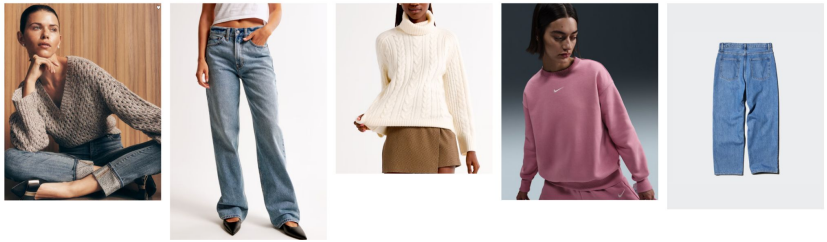
Made up Persona: Sophia

Collaborative Filtering

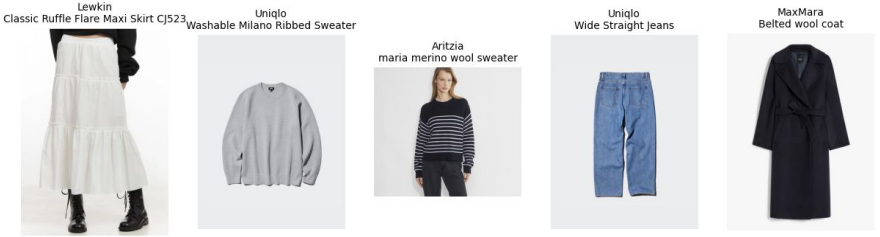
Neural CF Recommendations for Laura

Content Filtering

Low-rank



Two Tower



Next Steps

- **Continuously add data to our dataset** (more clothes/users)
  - Develop an automated pipeline to scrape images from retail websites and generating rating vectors for synthetically generated users
- **Add additional layer of optimization to our models**
  - Figure out how to incorporate user feedback from initially outputted recommendations
  - Reinforcement learning?
- **Combine/narrow down to ~2 models that best fit our needs**