Cold-Start Recommendations Across E-Commerce Platforms

Problem Statement

- **Goal:** Optimize personalized recommendations for users visiting new e-commerce websites with minimal purchase history.
- **Challenge:** The cold-start problem limits recommendation effectiveness, leading to poor user experience, reduced engagement, and lower conversion rates.

Why This Problem Matters

- **User Frustration:** Irrelevant recommendations decrease user satisfaction.
- **Business Impact:** Affects customer retention and revenue.
- **Cross-Platform Relevance:** Users switch between platforms (e.g., Amazon to eBay), losing valuable interaction data.

Success Metrics

- **Precision@k, Recall@k, NDCG:** Measure relevance of recommendations.
- **Click-Through Rate (CTR):** User engagement with recommendations.
- **Computational Efficiency:** Latency and memory usage.

Constraints & Data Requirements

- **Constraints:**
  - Data availability across platforms
  - Real-time recommendation performance
  - User privacy compliance
- **Required Data:**
  - User interactions (orders, wishlists, browsing)
  - Product metadata (titles, images, reviews)
  - User-generated content (ratings, preferences)

Technical Approach

- **Model:** GraphSAGE with Metadata Integration
- **Features:**
  - Pre-trained BERT embeddings for text metadata
  - Image embeddings (e.g., CLIP)
  - Graph topology (user-item interactions)
- **Objective Function:** Optimize ranking loss for cold-start scenarios

Algorithm & Implementation

- **GraphSAGE Aggregation:** Modified to handle heterogeneous features
- **Libraries:**
  - PyTorch Geometric (GraphSAGE)
  - Hugging Face Transformers (BERT embeddings)
  - Scikit-learn (Preprocessing)
- **Web Extension:** Real-time recommendations across e-commerce sites

Validation Methods

- **Evaluation Metrics:** Precision@k, Recall@k, NDCG
- **Cross-Validation:** Simulate cold-start by hiding interactions
- **Baseline Comparison:** Collaborative filtering, standard GraphSAGE

Initial Results

- **Custom Dataset (76 images):**
  - HDBSCAN identified 3 clusters (sweaters, sweatpants, dresses/skirts)
  - Low CPU usage (<5%), fast execution (<1 min)
- **Fashion-MNIST (1,000 images):**
  - 33 clusters detected with meaningful groupings
  - Minimal resource consumption, completed in minutes

Current Limitations

- **Outliers:** Patterned clothing not clustered well
- **Scalability:** Challenges with large datasets
- **Metadata Mismatch:** Inconsistent data formats across sites

Next Steps

- **Optimize Clustering:** Fine-tune HDBSCAN parameters
- **Improve Metadata Handling:** Better standardization across platforms
- **Real-Time Performance:** Enhance web extension efficiency
- **Expand Datasets:** Include more diverse product data

1. Heterogeneous Graph Neural Networks (HeteroGNN)
    - **Why It's a Good Fit:** Handles multi-type data (users, products, brands) and multi-relation graphs.
    - **How It Works:**
        - Graph Construction: Users, products, brands as nodes; interactions as edges.
        - Node Features: BERT for text, CLIP for images, price normalization.
        - Aggregation: Learns from diverse node and edge types.
    - **Application to Our Problem:**
        - Build a cross-platform graph with nodes representing users, products, and brands from different e-commerce sites.
        - Use product metadata (image embeddings, descriptions, prices) as node features.
        - Learn user preferences from Store1 and predict relevant products in Store2.
    - **Scalability:** Mini-batch training with neighbor sampling.

2. Factorization Machines (FM)

- **Why It's a Good Fit:** Integrates metadata into collaborative filtering efficiently.
- **How It Works:**
  - User-Item Matrix enriched with brand, price, description, image embeddings.
  - Predicts likelihood of user-item interactions.
- **Application to Our Problem:**
  - Represent each user-item interaction with features like user behavior, product metadata (brand, price, image embeddings).
  - Factorization Machines will learn feature interactions, allowing us to make predictions even if the user has no prior history on the current website.
  - Ideal for quick recommendations based on metadata similarity across platforms.
- **Scalability:** Fast training/inference, easy to update.
- **Challenges:** Limited non-linear relationship modeling, requires feature engineering.

3. Meta-Learning (MAML-Based Recommender)

- **Why It's a Good Fit:** Adapts quickly to new platforms with minimal data.
- **How It Works:**
  - Meta-Training: Learns from multiple platforms (Amazon, eBay).
  - Meta-Adaptation: Fine-tunes for new sites with minimal interactions.
  - Incorporates metadata for cold-start scenarios.
- **Application to Our Problem:**
  - Train the model on datasets from known e-commerce platforms (meta-learning phase).
  - When a user visits a new store, fine-tune the model using limited interactions, leveraging metadata like images, brand, and descriptions.
- **Scalability:** Optimized for few-shot learning, quick adaptation.
- **Challenges:** Complex implementation, meta-training required.

Algorithm Comparison Summary

Algorithm	Strengths	Challenges	Best Use Case
Heterogeneous GNN	Handles multi-type data, scalable	Complex graph design, dynamic updates	Rich metadata with complex relationships
Factorization Machines	Simple, fast, scalable	Limited to linear interactions	Real-time recommendations with metadata
Meta-Learning (MAML)	Quick adaptation for cold-start users	Complex to implement, requires meta-training	Cross-platform personalization with minimal data

Questions & Discussion

- How can we improve real-time cross-platform recommendations?
- Strategies for handling data privacy across websites?
- Ideas for integrating additional metadata types?