Special Topics in Applied Mathematics I
Solution 2

XIA JIAHAN

February 6, 2026

1 Introduction to Recommendation Systems

In the information explosion era, recommendation systems (RS) are essential for filtering vast data and delivering personalized content across platforms like Amazon, Netflix, YouTube, and social media, enhancing user experience and business value by predicting preferences. They use machine learning to model similarities between items and user interests, powering two main types of suggestions: personalized homepage feeds (unique to each user) and related item recommendations (e.g., suggesting science apps when viewing a math app). This enables platforms like YouTube and the Google Play Store to anticipate what to show next. The core motivation is to help users navigate immense content libraries, where search alone falls short, by surfacing relevant, sometimes unexpected items and facilitating discovery.

A recommender system matches a user's query (context) to recommendable entities (items) by learning embeddings that place queries and items in a shared vector space for efficient similarity computation. It then performs candidate generation to quickly narrow a huge corpus to hundreds or thousands of candidates, scoring to rank this smaller set and select roughly ten items to display, and finally re-ranking to apply constraints such as removing content the user dislikes, boosting fresher items, and ensuring diversity and fairness. We will discuss each stage with examples from systems like YouTube.

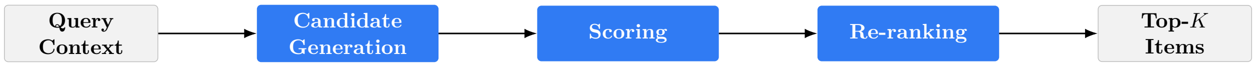

Figure 1: Overview of the Recommender System Process

2 Candidate generation

2.1 Overview

Candidate generation is the first stage of recommendation. Given a query, the system generates a set of relevant candidates. The following table shows two common candidate generation approaches:

Type	Definition	Example
content-based filtering	Uses *similarity between items* to recommend items similar to what the user likes.	If user A watches two cute cat videos, then the system can recommend cute animal videos to that user.
collaborative filtering	Uses *similarities between queries and items simultaneously* to provide recommendations.	If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn't seen any videos similar to video 1).

Both content-based and collaborative filtering embed items and queries (contexts) into a shared low-dimensional space $E = \mathbb{R}^d$. Candidate generation then retrieves items whose embeddings are most similar to the query embedding: a similarity function $s : E \times E \to \mathbb{R}$ ranks candidates by $s(q, x)$. Common choices include cosine similarity, dot product, and Euclidean distance.

Unlike cosine similarity, dot product is sensitive to embedding norms, so large-norm vectors can be preferred even when the angle is similar. This can affect recommendations as follows:

- Popular items often get large norms, so dot product can capture popularity but may cause popular items to dominate. A tempered variant is $s(q, x) = \|q\|^\alpha \|x\|^\alpha \cos(q, x)$ for $\alpha \in (0, 1)$.
- Rare items may be updated infrequently; if initialized with large norms they can be over-recommended. To avoid this problem, be careful about embedding initialization and use appropriate regularization.

2.2 Content-based filtering

Content-based filtering recommends items that are similar to what a user already likes by comparing item feature vectors. Figure 2 illustrates a simple (binary) feature matrix for apps; a user can be represented in the same feature space using explicit preferences and past installs. Given a similarity metric (e.g., dot product), the system scores each candidate item against the user vector and recommends the highest-scoring items, without using information from other users.

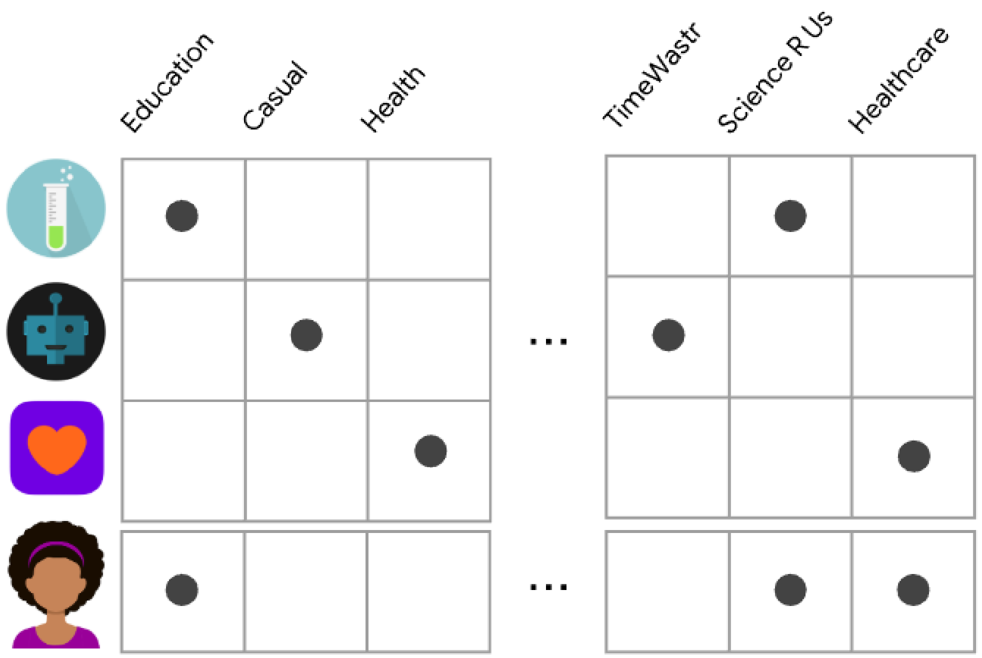

Figure 2: A toy feature matrix for apps (rows) and features (columns).

Content-based filtering has the advantage that it does not require data about other users, since recommendations are computed specifically for a given user, which makes it easier to scale to a large user base; it can also capture a user's particular interests and recommend niche items that few other users care about. However, it often relies on partially hand-engineered item features and substantial domain knowledge, so performance is limited by the quality of those features, and it can only recommend based on the user's existing interests, giving it limited ability to broaden the user's preferences beyond what they have already shown.

2.3 Collaborative filtering

To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B. Furthermore, the embeddings can be learned automatically, without relying on hand-engineering of features.