Artificial Intelligence Applied to the Web

Chapter 4 Part 4 - Recommendation Systems II

Diego Cornejo, Felipe Hernández and Juan Velásquez

University of Chile
Departament of Industrial Engineering

Spring 2025

Outline

Content-Based Recommendation

Introduction

- Unlike Collaborative Filtering (CF), this approach uses information **about the items** themselves.
- The basic idea is to recommend items based on their characteristics and the user's preferences.
- **Example:** If you know a user likes fantasy novels, you can recommend a new fantasy novel to them.

Content-Based Recommendation

Core Requirements

- Two key pieces of information are needed:
  - *Item Descriptions:* Information about the item's features, such as genre, keywords, or author. This is the "content".
  - *User Profile:* A model that describes the user's past interests and preferences.
- The main task is to determine which items best **match** the user's profile.

Content Representation

Simple Approach: Item Features

- The simplest way to describe items is with an explicit list of features, like a database table.
- Each item is represented by its characteristics (genre, author, keywords, etc.).

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

Content Representation

User Profiles

- A user's profile is built using the same features to find matches.
- This profile can be created in different ways:
  - **Explicitly:** by asking the user for their preferences.
  - **Automatically:** by deriving preferences from items the user has liked.

Example User Profile for "Alice":

Title	Genre	Author	Type	Price	Keywords
…	Fiction, Suspense	Brunonia Barry, Ken Follett	Paperback	25.65	detective, murder, New York

Content Representation

Measuring Similarity

- The system recommends items by evaluating how "similar" a new item is to the user's profile.
- Similarity can be measured in different ways. For example, given an unseen book B, the system could check whether the author of the book is in the list of Alice's authors.
- Another way is to calculate the similarity or overlap of the involved keywords, relying on the Dice coefficient:

$$\frac{2 \times \left|\text{keywords}(b_i) \cap \text{keywords}(b_j)\right|}{\left|\text{keywords}(b_i)\right| + \left|\text{keywords}(b_j)\right|}$$

- This formula calculates the similarity between two books, b_i and b_j, based on their shared keywords.

Content Representation

From Keywords to Vectors

- Instead of simple feature lists, the standard approach is to use a list of relevant keywords from the document's text.
- The main idea is that these keywords can be generated **automatically**.
- This transforms the document's content into a mathematical representation, typically a vector.

The Vector Space Model

Problems with a Simple Approach

- A naïve approach is a *Boolean vector*, where a "1" means a word is present and a "0" means it's absent.
- This simple method has obvious problems:
  - It assumes every word has the same importance.
  - It creates a bias towards longer documents, as they naturally have a higher chance of matching a user's profile.

The Vector Space Model
Solution: TF-IDF

- To solve these issues, documents are described using *TF-IDF*.
- This is a standard technique from Information Retrieval that encodes documents as vectors in a multi-dimensional space.
- The coordinates of the vector are a product of two key measures:
  - **Term Frequency (TF)**
  - **Inverse Document Frequency (IDF)**

The Vector Space Model

TF-IDF

- TF: Measures how often a specific term appears in a single document (important words are assumed to appear more frequently).

$$\mathrm{TF}(i,j) = \frac{\mathrm{freq}(i,j)}{\mathrm{maxOthers}(i,j)}$$

- IDF: Reduces the weight of terms that appear very frequently across **all** documents in the collection (common words, such as “the”, “is”, “a”, are not useful for distinguishing between documents.).

$$\mathrm{IDF}(i) = \log\left(\frac{N}{n(i)}\right)$$

The Vector Space Model

The Final Vector

- The combined *TF-IDF weight* for a term in a document is the product of its TF and IDF scores:

$$\text{TF-IDF}(i, j) = \text{TF}(i, j) * \text{IDF}(i)$$

- The final document is represented not as a simple list of words, but as a **vector of these calculated TF-IDF weights**.

Improving the Vector Space Model

Stop Words and Stemming

- TF-IDF vectors are often very large and sparse. Several techniques are used to make them more compact and remove noise.
- *Remove Stop Words:*
  - This involves removing common words (e.g., “a”, “the”, “on”) that appear in nearly all documents and offer little descriptive value.
- *Use Stemming:*
  - This process replaces variants of a word with their common root (e.g., “stemming” -> “stem”, “went” -> “go”).
  - It helps reduce the vector size and improves matching.

Improving the Vector Space Model

Other Improvements

- *Size Cutoffs:*
  - To reduce noise, some systems use only the top n most informative words (e.g., the top 100 or 128 keywords).
  - This requires finding a balance: too few keywords can miss important features, while too many can add noise and worsen accuracy.
- *Use Phrases as Terms:*
  - Multi-word phrases (e.g., "United Nations") can be more descriptive than single words and can be encoded as additional dimensions in the vector space.

Improving the Vector Space Model

Limitations

- A major limitation of the keyword-based approach is its *lack of semantic understanding*. It does not consider the context of the words.
- **Example:** A review stating, *"there is nothing on the menu that a vegetarian would like."*
  - The word "vegetarian" would likely receive a high weight.
  - This could lead to an unintended and incorrect recommendation for a user who is actually interested in vegetarian restaurants.

Content Base Recommendation

Similarity-Based Retrieval

- While Collaborative Filtering recommends what **similar users** liked, Content-Based recommendation suggests items that are **similar to what the current user liked in the past**.
- Based on an existing user profile, the system's goal is to predict whether a user will like new items they haven't seen yet.
- The most common techniques to achieve this rely on representing items (documents) using the **vector-space model**.

Similarity-Based Retrieval

k-Nearest Neighbors (kNN)

- A straightforward method to predict if a user will like a new document is to check if they liked similar documents in the past.
- This requires two things:
  - A history of the user's "like/dislike" ratings.
  - A similarity measure to compare documents, most commonly **cosine similarity**.

k-Nearest Neighbors (kNN)

How it Works: “Voting”

- The prediction for a new, unseen item is based on letting the k most similar items (that the user has already rated) “vote”.
- **Example:** If 4 out of the 5 most similar items were “liked” by the user, the system predicts the new item will also be liked.
- This simple method has several possible variations, such as:
  - Weighting the “votes” based on the degree of similarity.
  - Using a minimum similarity threshold to be considered a neighbor.

k-Nearest Neighbors (kNN)

Application: Short-Term vs. Long-Term Interests

- The kNN method is particularly effective for modeling a user's *short-term interests*.
- By only considering recently rated items, it can quickly adapt and recommend relevant follow-up content (e.g., news stories about a recent event).
- This can be combined with other models (like probabilistic classifiers) that capture a user's *long-term preferences* over months of activity.
- A common strategy is to first try the short-term model (kNN) and, if no recommendations are found, fall back to the long-term model.

k-Nearest Neighbors (kNN)

A Practical Example

Let's find a movie recommendation for Alex.

- 1. User Profile: The system knows Alex has "liked" two movies:
  - *Star Wars*: Keywords are [sci-fi, space opera, action].
  - *Blade Runner*: Keywords are [sci-fi, dystopian, classic].
- 2. A New Item: A new movie, *Dune*, becomes available. Its keywords are [sci-fi, space opera, epic].

k-Nearest Neighbors (kNN)

A Practical Example

- 3. Find Neighbors: The system compares *Dune* to Alex's profile. It finds the "nearest neighbors" (most similar items) based on keyword overlap:
  - *Star Wars* is a very close neighbor (shares sci-fi, space opera).
  - *Blade Runner* is also a neighbor (shares sci-fi).
- 4. Recommend: Since Alex "liked" the nearest neighbors, the system predicts he will also like *Dune* and recommends it.

k-Nearest Neighbors (kNN)

Advantages and Disadvantages

- *Advantages:*
  - Relatively simple to implement.
  - Adapts quickly to recent changes in a user's interests.
  - Can make reasonable predictions with a relatively small number of initial ratings.
- *Disadvantages:*
  - Prediction accuracy can be lower than more sophisticated techniques.
  - Naïve implementations can be computationally intensive and slow when searching for neighbors in a large dataset.

Probabilistic Methods

Recommendation as Classification

- This approach views content-based recommendation as a classic **text classification problem**.
- The goal is to build a model that can automatically classify a new, unseen document into one of several predefined categories.
- For recommendations, the categories are simple: **"like"** and **"dislike"** (or "hot" and "cold").

Probabilistic Methods

The Naïve Bayes Classifier

- A prominent technique for this task is the *Naïve Bayes classifier.*
- Basic setup:
  - Documents are represented by *Boolean feature vectors* (a "1" if a keyword is present, "0" if absent).
  - The model calculates the probability of a document belonging to the "like" class given the words it contains: $P(\text{like} \mid \text{text})$.

$$P(Y|X) = \frac{\prod_{i=1}^{d} P(X_i|Y) \times P(Y)}{P(X)}$$

- Core Assumption: It "naïvely" assumes that the occurrences of words in a document are **conditionally independent** of each other.

Probabilistic Methods

Why Naïve Bayes Works

- The assumption of word independence is technically incorrect (e.g., “Hong” is not independent of “Kong”).
- However, the classifier has been shown to work surprisingly well in practice.
- Even if the calculated probability scores are not perfectly accurate, the final classification (the choice between “like” or “dislike”) is often correct.

Probabilistic Methods (Naïve Bayes)

A Practical Example

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	**1**
2	0	0	1	1	**0**
3	1	1	0	0	**1**
4	1	0	1	1	**1**
5	0	0	0	1	**0**
6	1	1	0	0	**?**

Probabilistic Methods (Naïve Bayes)

A Practical Example

$$
\begin{aligned}
P(X|\text{Label}=1) = {} & P(\text{recommender}=1|\text{Label}=1) \times \\
& P(\text{intelligent}=1|\text{Label}=1) \times \\
& P(\text{learning}=0|\text{Label}=1) \times \\
& P(\text{school}=0|\text{Label}=1)
\end{aligned}
$$

$$
= \frac{3}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \approx 0.149
$$

Probabilistic Methods

Advantages and Disadvantages

- Advantages:
  - Generally provides good accuracy.
  - The model is easy to update with new data.
  - It is computationally fast for both learning and prediction.
- Disadvantage (Cold-Start Problem):
  - Like most learning techniques, it still requires an initial set of training data (past user ratings) to build a reasonable user profile.

Probabilistic Methods

Beyond Boolean: The Multinomial Model

- The simple Boolean (presence/absence) representation loses potentially important information.
- A more advanced approach, the *multinomial model*, takes the **count** of how many times a term appears in a document into account.
- This model generally leads to significantly better classification results, especially for longer documents, because it can properly weight the evidence from frequently occurring keywords.

Content-Based Methods

Limitations

1. Shallow Content Analysis
   - Relying on keywords alone may not be sufficient to judge an item's true quality.
   - The analysis often misses key aspects like:
     - For web pages: aesthetics, usability, or timeliness.
     - The difference between a well-written vs. a poorly written article using the same keywords.
   - Content may be too short to extract meaningful features (e.g., jokes).
   - Content may not be text-based and is therefore difficult to analyze automatically (e.g., images, audio, video).

Content-Based Methods

Limitations

2. Overspecialization

- These systems have a tendency to recommend **"more of the same"**.
- They only suggest items that are very similar to what the user has already liked, leading to obvious and uninteresting recommendations.
- This limits the user's ability to discover new, unexpected items. The goal is to increase **serendipity**.
- **Example:** A news recommender suggesting multiple articles that all cover the exact same event the user just read about.

Content-Based Methods

Limitations

3. The Ramp-Up (Cold-Start) Problem
   - Although they don't need a large community, CB systems still require an initial set of ratings from a new user to build their profile.
   - Users are often unwilling to rate a large number of items (e.g., 20-50) before the service becomes useful.
   - *Possible Solutions:*
     - Ask the user to provide an initial list of keywords or topics of interest.
     - "Reuse" information from other personalized applications the user has already configured.

Artificial Intelligence Applied to the Web

Chapter 4 Part 4 - Recommendation Systems II

Diego Cornejo, Felipe Hernández and Juan Velásquez

University of Chile
Departament of Industrial Engineering

Spring 2025