Recommender Systems

Collaborative Filtering & Content-Based Recommending
Recommender Systems

• Systems for recommending items (e.g. books, movies, CD’s, web pages, newsgroup messages) to users based on examples of their preferences.

• Many websites provide recommendations (e.g. Amazon, NetFlix, Pandora).

• Recommenders have been shown to substantially increase sales at on-line stores.

• There are two basic approaches to recommending:
  – Collaborative Filtering (a.k.a. social filtering)
  – Content-based
Book Recommender

Machine Learning

User Profile

Red Mars
Foundation
Jurassic Park
Lost World
2001
Difference Engine

Neuromancer
2010
Personalization

• Recommenders are instances of personalization software.
• Personalization concerns adapting to the individual needs, interests, and preferences of each user.
• Includes:
  – Recommending
  – Filtering
  – Predicting (e.g. form or calendar appt. completion)
• From a business perspective, it is viewed as part of Customer Relationship Management (CRM).
Machine Learning and Personalization

• Machine Learning can allow learning a user model or profile of a particular user based on:
  – Sample interaction
  – Rated examples

• This model or profile can then be used to:
  – Recommend items
  – Filter information
  – Predict behavior
Collaborative Filtering

- Maintain a database of many users’ ratings of a variety of items.
- For a given user, find other similar users whose ratings strongly correlate with the current user.
- Recommend items rated highly by these similar users, but not rated by the current user.
- Almost all existing commercial recommenders use this approach (e.g. Amazon).
Collaborative Filtering
Collaborative Filtering Method

• Weight all users with respect to similarity with the active user.
• Select a subset of the users (neighbors) to use as predictors.
• Normalize ratings and compute a prediction from a weighted combination of the selected neighbors’ ratings.
• Present items with highest predicted ratings as recommendations.
Similarity Weighting

• Typically use Pearson correlation coefficient between ratings for active user, $a$, and another user, $u$.

$$
c_{a,u} = \frac{\text{covar}(r_a, r_u)}{\sigma_{r_a} \sigma_{r_u}}
$$

$r_a$ and $r_u$ are the ratings vectors for the $m$ items rated by both $a$ and $u$

$r_{i,j}$ is user $i$’s rating for item $j$
Covariance and Standard Deviation

- **Covariance:**
  \[
  \text{covar}(r_a, r_u) = \frac{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{m}
  \]

- **Standard Deviation:**
  \[
  \sigma_{r_x} = \sqrt{\frac{\sum_{i=1}^{m} (r_{x,i} - \bar{r}_x)^2}{m}}
  \]
Significance Weighting

• Important not to trust correlations based on very few co-rated items.

• Include *significance weights*, $s_{a,u}$, based on number of co-rated items, $m$.

\[
W_{a,u} = S_{a,u} C_{a,u}
\]

\[
s_{a,u} = \begin{cases} 
1 & \text{if } m > 50 \\
\frac{m}{50} & \text{if } m \leq 50 
\end{cases}
\]
Neighbor Selection

- For a given active user, $a$, select correlated users to serve as source of predictions.
- Standard approach is to use the most similar $n$ users, $u$, based on similarity weights, $w_{a,u}$
- Alternate approach is to include all users whose similarity weight is above a given threshold.
Rating Prediction

- Predict a rating, $p_{a,i}$, for each item $i$, for active user, $a$, by using the $n$ selected neighbor users, $u \in \{1,2,\ldots,n\}$.
- To account for users different ratings levels, base predictions on differences from a user’s average rating.
- Weight users’ ratings contribution by their similarity to the active user.

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u} (r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} w_{a,u}}$$
Problems with Collaborative Filtering

• **Cold Start**: There needs to be enough other users already in the system to find a match.

• **Sparsity**: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items.

• **First Rater**: Cannot recommend an item that has not been previously rated.
  - New items
  - Esoteric items

• **Popularity Bias**: Cannot recommend items to someone with unique tastes.
  - Tends to recommend popular items.
Content-Based Recommending

- Recommendations are based on information on the **content** of items rather than on other users’ opinions.
- Uses a machine learning algorithm to induce a profile of the users preferences from examples based on a featural description of content.
- Some previous applications:
  - Newsweeder (Lang, 1995)
  - Syskill and Webert (Pazzani et al., 1996)
Advantages of Content-Based Approach

• No need for data on other users.
  – No cold-start or sparsity problems.

• Able to recommend to users with unique tastes.

• Able to recommend new and unpopular items
  – No first-rater problem.

• Can provide explanations of recommended items by listing content-features that caused an item to be recommended.
Disadvantages of Content-Based Method

• Requires content that can be encoded as meaningful features.
• Users’ tastes must be represented as a learnable function of these content features.
• Unable to exploit quality judgments of other users.
  – Unless these are somehow included in the content features.
LIBRA
Learning Intelligent Book Recommending Agent

• Content-based recommender for books using information about titles extracted from Amazon.
• Uses information extraction from the web to organize text into fields:
  – Author
  – Title
  – Editorial Reviews
  – Customer Comments
  – Subject terms
  – Related authors
  – Related titles
LIBRA System

Amazon Pages

LIBRA Database

Information Extraction

Rated Examples

Machine Learning

Learner

User Profile

Predictor

Recommendations
1. ~~~~~
2. ~~~~~~~
3. ~~~~~~

...
Sample Amazon Page

Age of Spiritual Machines
Sample Extracted Information

Title: <The Age of Spiritual Machines: When Computers Exceed Human Intelligence>
Author: <Ray Kurzweil>
Price: <11.96>
Publication Date: <January 2000>
ISBN: <0140282025>
Related Titles: <Title: <Robot: Mere Machine or Transcendent Mind>
   Author: <Hans Moravec> >
   ...
Reviews: <Author: <Amazon.com Reviews> Text: <How much do we humans…> >
   ...
Comments: <Stars: <4> Author: <Stephen A. Haines> Text:<Kurzweil has …> >
   ...
Related Authors: <Hans P. Moravec> <K. Eric Drexler>…
Subjects: <Science/Mathematics> <Computers> <Artificial Intelligence> …
Libra Content Information

- Libra uses this extracted information to form “bags of words” for the following slots:
  - Author
  - Title
  - Description (reviews and comments)
  - Subjects
  - Related Titles
  - Related Authors
Libra Overview

- User rates selected titles on a 1 to 10 scale.
- Libra uses a naïve Bayesian text-categorization algorithm to learn a profile from these rated examples.
  - Rating 6–10: Positive
  - Rating 1–5: Negative
- The learned profile is used to rank all other books as recommendations based on the computed posterior probability that they are positive.
- User can also provide explicit positive/negative keywords, which are used as priors to bias the role of these features in categorization.
Bayesian Categorization in LIBRA

- Model is generalized to generate a **vector** of bags of words (one bag for each slot).
  - Instances of the same word in different slots are treated as separate features:
    - “Chrichton” in author vs. “Chrichton” in description
- Training examples are treated as **weighted** positive or negative examples when estimating conditional probability parameters:
  - An example with rating $1 \leq r \leq 10$ is given:
    - **positive** probability: $(r - 1)/9$
    - **negative** probability: $(10 - r)/9
Implementation

- Stopwords removed from all bags.
- A book’s title and author are added to its own related title and related author slots.
- All probabilities are smoothed using Laplace estimation to account for small sample size.
- Lisp implementation is quite efficient:
  - Training: 20 exs in 0.4 secs, 840 exs in 11.5 secs
  - Test: 200 books per second
Explanations of Profiles and Recommendations

- Feature strength of word $w_k$ appearing in a slot $s_j$:

$$\text{strength}(w_k, s_j) = \log \frac{P(w_k \mid \text{positive}, s_j)}{P(w_k \mid \text{negative}, s_j)}$$
Libra Demo

http://www.cs.utexas.edu/users/libra
Experimental Data

- Amazon searches were used to find books in various genres.
- Titles that have at least one review or comment were kept.
- Data sets:
  - Literature fiction: 3,061 titles
  - Mystery: 7,285 titles
  - Science: 3,813 titles
  - Science Fiction: 3.813 titles
Rated Data

- 4 users rated random examples within a genre by reviewing the Amazon pages about the title:
  - LIT1 936 titles
  - LIT2 935 titles
  - MYST 500 titles
  - SCI 500 titles
  - SF 500 titles
Experimental Method

• 10-fold cross-validation to generate learning curves.
• Measured several metrics on independent test data:
  – Precision at top 3: % of the top 3 that are positive
  – Rating of top 3: Average rating assigned to top 3
  – Rank Correlation: Spearman’s, $r_s$, between system’s and user’s complete rankings.
• Test ablation of related author and related title slots (LIBRA-NR).
  – Test influence of information generated by Amazon’s collaborative approach.
Experimental Result Summary

• **Precision at top 3** is fairly consistently in the 90’s% after only **20 examples**.
• **Rating of top 3** is fairly consistently above 8 after only **20 examples**.
• All results are always significantly better than random chance after only **5 examples**.
• **Rank correlation** is generally above 0.3 (moderate) after only **10 examples**.
• **Rank correlation** is generally above 0.6 (high) after **40 examples**.
Precision at Top 3 for Science
Rating of Top 3 for Science
Rank Correlation for Science
User Studies

• Subjects asked to use Libra and get recommendations.
• Encouraged several rounds of feedback.
• Rated all books in final list of recommendations.
• Selected two books for purchase.
• Returned reviews after reading selections.
• Completed questionnaire about the system.
Combining Content and Collaboration

• Content-based and collaborative methods have complementary strengths and weaknesses.
• Combine methods to obtain the best of both.
• Various hybrid approaches:
  – Apply both methods and combine recommendations.
  – Use collaborative data as content.
  – Use content-based predictor as another collaborator.
  – Use content-based predictor to complete collaborative data.
Movie Domain

• **EachMovie Dataset** [Compaq Research Labs]
  – Contains user ratings for movies on a 0–5 scale.
  – 72,916 users (avg. 39 ratings each).
  – 1,628 movies.
  – Sparse user-ratings matrix – (2.6% full).

• **Crawled Internet Movie Database (IMDb)**
  – Extracted content for titles in EachMovie.

• **Basic movie information:**
  – Title, Director, Cast, Genre, etc.

• **Popular opinions:**
  – User comments, Newspaper and Newsgroup reviews, etc.
Content-Boosted Collaborative Filtering

EachMovie → Web Crawler → Movie Content Database → Content-based Predictor → Active User Ratings → Collaborative Filtering → Recommendations

IMDb → User Ratings Matrix (Sparse) → Full User Ratings Matrix
Content-Boosted CF - I

User-ratings Vector

Training Examples

Content-Based Predictor

Pseudo User-ratings Vector

- User-rated Items
- Unrated Items
- Items with Predicted Ratings
Content-Boosted CF - II

- Compute pseudo user ratings matrix
  - Full matrix – approximates actual full user ratings matrix
- Perform CF
  - Using Pearson corr. between pseudo user-rating vectors
Experimental Method

• Used subset of EachMovie (7,893 users; 299,997 ratings)

• Test set: 10% of the users selected at random.
  – Test users that rated at least 40 movies.
  – Train on the remainder sets.

• Hold-out set: 25% items for each test user.
  – Predict rating of each item in the hold-out set.

• Compared CBCF to other prediction approaches:
  – Pure CF
  – Pure Content-based
  – Naïve hybrid (averages CF and content-based predictions)
Metrics

• Mean Absolute Error (MAE)
  – Compares numerical predictions with user ratings

• ROC sensitivity [Herlocker 99]
  – How well predictions help users select high-quality items
  – Ratings ≥ 4 considered “good”; < 4 considered “bad”

• Paired t-test for statistical significance
Results - I

CBCF is significantly better (4% over CF) at (p < 0.001)
Results - II

CBCF outperforms rest (5% improvement over CF)
Active Learning
(Sample Section, Learning with Queries)

• Used to reduce the number of training examples required.

• System requests ratings for specific items from which it would learn the most.

• Several existing methods:
  – Uncertainty sampling
  – Committee-based sampling
Semi-Supervised Learning
(Weakly Supervised, Bootstrapping)

• Use wealth of unlabeled examples to aid learning from a small amount of labeled data.

• Several recent methods developed:
  – Semi-supervised EM (Expectation Maximization)
  – Co-training
  – Transductive SVM’s
Conclusions

• Recommending and personalization are important approaches to combating information over-load.
• Machine Learning is an important part of systems for these tasks.
• Collaborative filtering has problems.
• Content-based methods address these problems (but have problems of their own).
• Integrating both is best.