Personalized Book Recommendations

Team 8

Emily Strong
Manoj Karru
Nishant Gandhi

Project Objective

Compare deep learning approaches to personalized recommendation to classic and state-of-the-art techniques.

Process

Data Engineering

1. Data Wrangling
2. Exploratory Data Analysis
3. Feature Engineering

Model Development

1. Use libraries for classic algorithms
2. Implement custom algorithms for state-of-the-art and deep learning techniques
3. Train and hyper-parameter tune

Model Evaluation & Deployment

1. Evaluate each model for:
   a. Existing users
   b. Cold start users
2. Compare model performance
3. Deploy best model as notebook on AWS

Data Set

- This Dataset is scraped from public data on GoodReads.com.
- The Dataset is for the most popular 10k books on the site, with over 6 million user ratings includes book metadata and tags.
- For the Rating data we removed all ratings from users 1-5 to use for new user (cold-start) evaluation.
- For the remaining 53419 users, we performed a randomized 70-30 train-test split of their rating history.

Feature Engineering

Book Features

- One-hot encoded the author and language and used TF-IDF to embed the edition title, original title, and tags.
- Did non-negative matrix factorization to extract 25 latent features from these to model the topic of the work.
- With the scaled numeric features, this gives 36 books features.

User Features

- Calculated the average rating from each user in the training set and identified all of the books that the user had rated 4 or 5 stars
- Averaged the scaled numeric features and 500 components from the embedded features that explain 78% of their variance.
- Extracted 25 latent features from the averages to model the user’s book preferences.

Collaborative Filtering

Performs user-user or item-item comparisons to make recommendations. Two common algorithms:

1. KNN: Uses cosine similarity of the user's rating history with the ratings histories of all other users to select K most similar users
2. Matrix Factorization: Performs singular value decomposition for items based on their rating history

Both of these approaches do not natively support new users with no ratings history.

Example Input:

User ID	Book ID	Ratings
1	5	1
2	0	3
3	1	4

Logistic Regression

Trains on user features and the features of the content to predict the rating for each combination.

This is a **common technique in industry** due to its simplicity and its ability to accommodate new users and new content that don’t have a rating history but do have other features available.

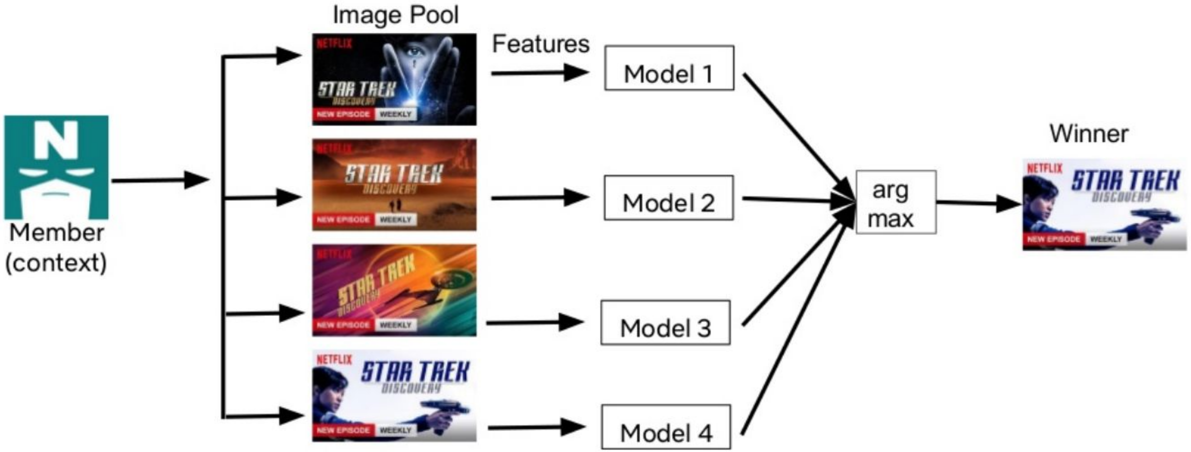
Example Input:

<table>
<tr><th>User Features</th><th>Content Features</th><th>Rating</th></tr>
<tr><td rowspan="3">User 1 Features</td><td>Book 1 Features</td><td>5</td></tr>
<tr><td>Book 2 Features</td><td>3</td></tr>
<tr><td>Book 3 Features</td><td>1</td></tr>
<tr><td rowspan="3">User 2 Features</td><td>Book 1 Features</td><td>3</td></tr>
<tr><td>Book 2 Features</td><td>2</td></tr>
<tr><td>Book 3 Features</td><td>1</td></tr>
</table>

Contextual Multi-Armed Bandits

Contextual Multi-Armed Bandits are a family of reinforcement learning algorithms.

They incorporate **state information** (e.g. user features) to predict rewards (e.g. ratings) for each content item, and then balance **exploring** to gain more information with **exploiting** to maximize the reward based on what is currently known.



Source: https://www.slideshare.net/JayaKawale/a-multiarmed-bandit-framework-for-recommendations-at-netflix

Contextual Multi-Armed Bandits

We use the bandit algorithm LinUCB which trains a ridge regression for each book, predicts the rating, and calculates an **upper confidence bound** based on the covariance and a tuning parameter. The item with the maximum upper confidence bound is selected.

When a new user has explicit features, personalized recommendations can be made. Otherwise default values can return default recommendations.

Example Input:

User Features	Book ID	Rating
User 1 Features	1	5
	2	3
	3	1
User 2 Features	1	3
	2	2
	3	1

Collaborative Denoising Autoencoders

Denoising autoencoders can be used for collaborative filtering by treating the missing ratings as noise.

CDAE is an algorithm that further improves the decoding by injecting user features to the input and adding bias to the output.

Predictions for new users can be made by inputting vectors of zeros as the user features and rating history, which returns the default values of the network.

Example Input:

User Features	Book 1	Book 2
User 1 Features	5	1
User 2 Features	0	3
User 3 Features	1	0

Wide and Deep Model

The wide and deep model is an ensemble method that trains a **feed-forward network** and a **generalized linear model** with L1 regularization that both feed into a **logistic regression**.

The logistic regression weights the inputs from each model to make a prediction.

It accepts the same inputs as a logistic regression, making it well-suited for new users and new content items when explicit features are available.

Example Input:

User Features	Content Features	Rating
User 1 Features	Book 1 Features	5
	Book 2 Features	3
	Book 3 Features	1
User 2 Features	Book 1 Features	3
	Book 2 Features	2
	Book 3 Features	1

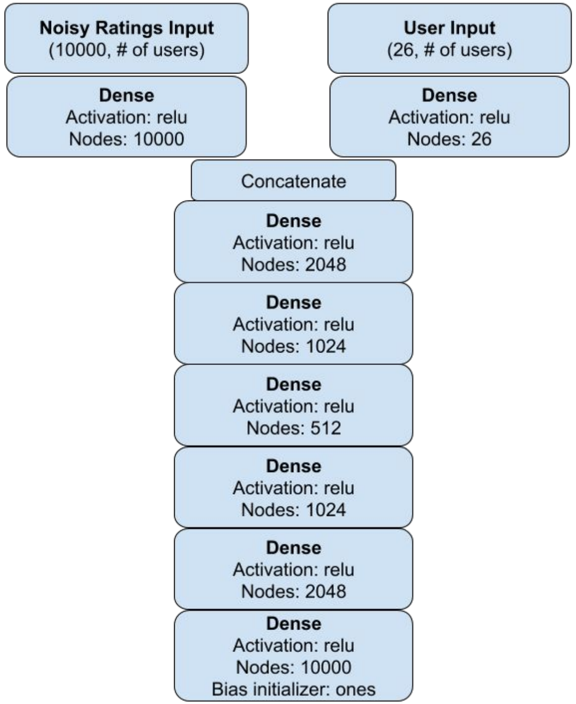
Model Evaluation

- All models are evaluated with:
    - Precision and recall for all ratings in the test set
    - Mean average precision for the top 10 recommendations per user from their books in the test set

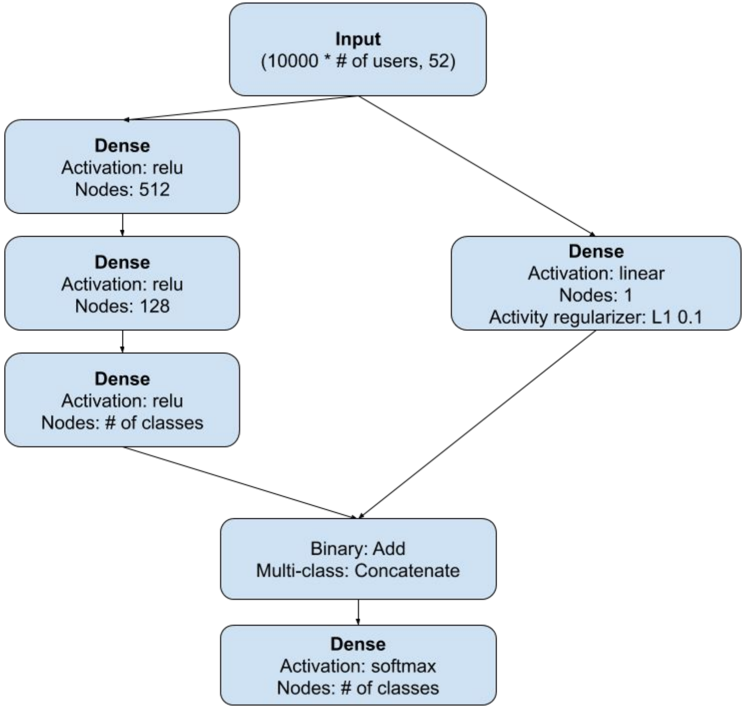
- Most of the algorithms were designed for binary prediction (eg clicks) so we tested both prediction of 5 star rating and prediction of like/dislike with a 4 or 5 rating assumed to be a like.

- For the models trained on the raw ratings, we also post-prediction converted the predicted values to the binary scheme and calculated a binary mAP to see how well they correspond to the like/dislike models.

Results



CDAE Best Architecture



Wide and Deep Best Architecture

Results

Algorithm	Test mAP [*binary mAP*]	Test Precision	Test Recall	Cold Start mAP	Cold Start Precision	Cold Start Recall
Collaborative Filtering (KNN)	0.0841 [0.6033]	0.5071	0.4214	N/A	N/A	N/A
Collaborative Filtering (SVD)	0.0949 [0.5845]	0.5376	0.4738	N/A	N/A	N/A
Logistic Regression	0.646	0.759	0.897	0.7492	0.9442	0.5519
LinUCB	0.6075	0.7478	0.9154	0.3967	0.7178	0.6149
CDAE	0.6628	0.7989	0.0378	0.52	0.7344	0.1638
Wide & Deep	0.5929	0.7604	0.9023	0.6279	0.5580	0.9547

For all models trained on 5 star ratings, the binary mAP was comparable to the mAP for the same model trained on likes/dislikes. This suggests that both methods are modeling the same underlying relationships.

Discussion

- Deep learning models did not outperform logistic regression and multi-armed bandits
- Results suggest that all models are capturing high-level “like/dislike” preferences without the granularity of predicting exact ratings
- Of the deep learning models, Wide & Deep did not have the bias issues CDAE seems to have

Demo

https://ec2-52-91-237-109.compute-1.amazonaws.com:8888/notebooks/Demo.ipynb