VISVESVARAYA NATIONAL INSTITUTE OF TECHNOLOGY NAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Scalable Recommendation System using Hybrid Approach: Content and Collaborative Filtering

DEVAL MUDIA BT16CSE060
AYUSH SINGH BT16CSE098
CHAITYA CHHEDA BT16CSE016
SADNEYA PUSALKAR BT16CSE076

UNDER THE GUIDANCE OF
Dr. S.R. SATHE

INTRODUCTION

A recommendation system assists users in finding desired content from enormous corpora. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users, to ensure he/she consumes the content of specific interest and relevance.

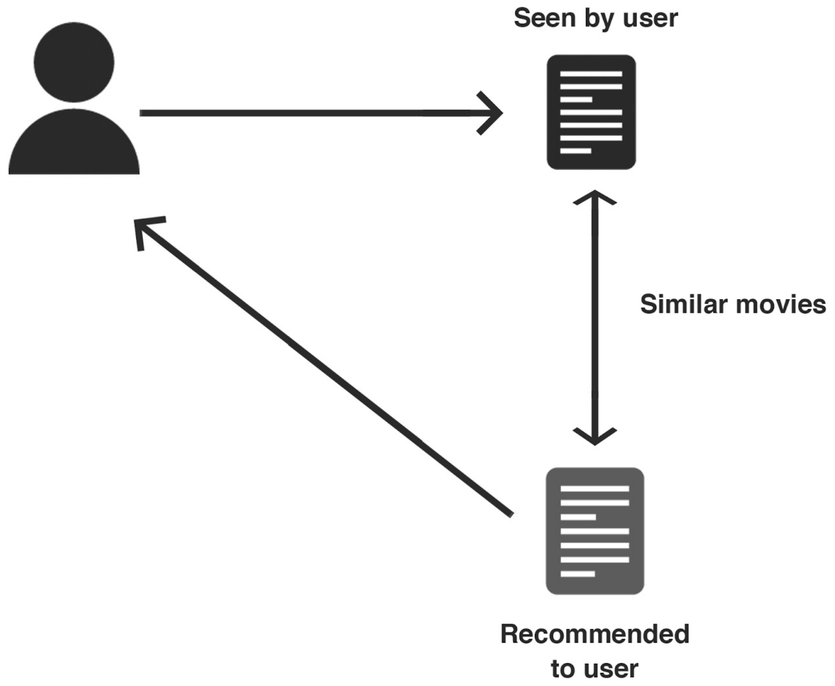
Companies like **Netflix** rely on targeted recommendations in order to serve a broad range of content to its users.

These recommendations are based on a user's previous watching history as well as content that similar users have watched.

CATEGORIES OF RECOMMENDER SYSTEMS

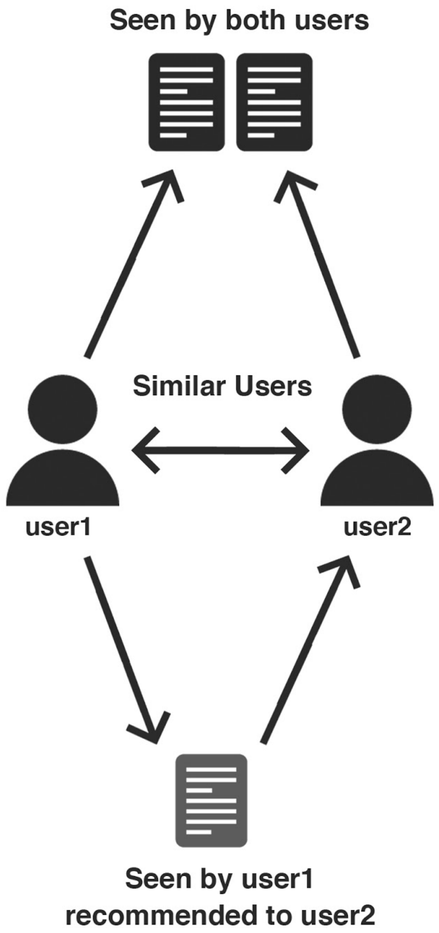
1. Content-based filtering

Makes recommendations based on the user's preference for certain type of items using **features** (meta-data) for example such as text and/or item attributes (no other user ratings are required) from past highly-rated items by that user.



2. Collaborative filtering

Makes recommendations based on the user's past **ratings** in addition to the other user's past ratings to find items of similar interest to the user.



3. Hybrid filtering

Combines both of the content-based and collaborative-based approaches [1]. There are several strategies for hybrid recommenders; however, the most popular are ***weighted recommenders*** which combine the scores outputted by different recommender algorithms.

PROJECT AIM

The focus of this project is to implement a hybrid recommender system that is based on the combination of content-based filtering and collaborative filtering methods.

The scalable **collaborative filter** is based on **Singular Value Decomposition (SVD)** and supports adding new users by a fold-in technique.

In **content-based filtering** each item is represented by a set of features or attributes in the vector space model which characterize that item and use **cosine similarity** to calculate the similarity scores between the items.

The content filtering and collaborative filtering are **combined** by taking the **weighted average** of the content recommendation and collaborative recommendation.

METHODOLOGY AND IMPLEMENTATION

1. CONTENT FILTERING

A popular method in content-based filtering is using the **item features** as the knowledge base and the user's profile or preference as an information retrieval query which then produces a recommendation to satisfy that query.

Features

	title	cast	director	keywords	genres
0	Toy Story	[Tom Hanks, Tim Allen, Don Rickles]	John Lasseter	[jealousy, toy, boy]	[Animation, Comedy, Family]

This metadata information that will be merged to form a dump:

['jealousy toy boy friendship friend rivalry boynextdoor newtoy toycomestolife tomhanks timallen donrickles johnlasseter Animation Comedy Family']

Then we can represent this information in the Vector Space Model by considering each feature of a movie as a term. Using sklearns' (python library) CountVectorizer, we form a count matrix which a matrix of these token/term count.

Document Term Matrix

	intelligent	applications	creates	business	processes	bots	are	i	do	intelligence
Doc 1	2	1	1	1	1	0	0	0	0	0
Doc 2	1	1	0	0	0	1	1	0	0	0
Doc 3	0	0	0	1	0	0	0	1	1	1

Then we simply compute the cosine similarity between a given movie and all the movies represented in the count matrix.

1.1 Challenge

The previous method although returns similar movies to our input movie, but it does not take into consideration criteria like popularity of the movie.

For example, **Batman and Robin** will have high cosine similarity to **The Dark Knight** because cast and crew is same but it was a terrible movie that shouldn't be recommended to anyone.

Therefore movies suggested should be similar to a particular movie that a user liked but still takes into consideration the popularity of the movies so that bad movies are not recommended to any user based on his/her taste.

1.2 Solution

To ensure popular movies with good critical response are returned, we take the top 25 movies based on cosine similarity scores from previous method and use IMDB's *weighted rating* formula:

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right)$$

where,

- v is the number of votes for the movie
- m is the minimum votes required to be listed in Top 250 chart
- R is the mean rating of the movie as a number from 1 to 10
- C is the mean vote across the whole report (currently 7.0)

to construct a chart of the most popular movies from those 25 movies.

2. COLLABORATIVE FILTERING

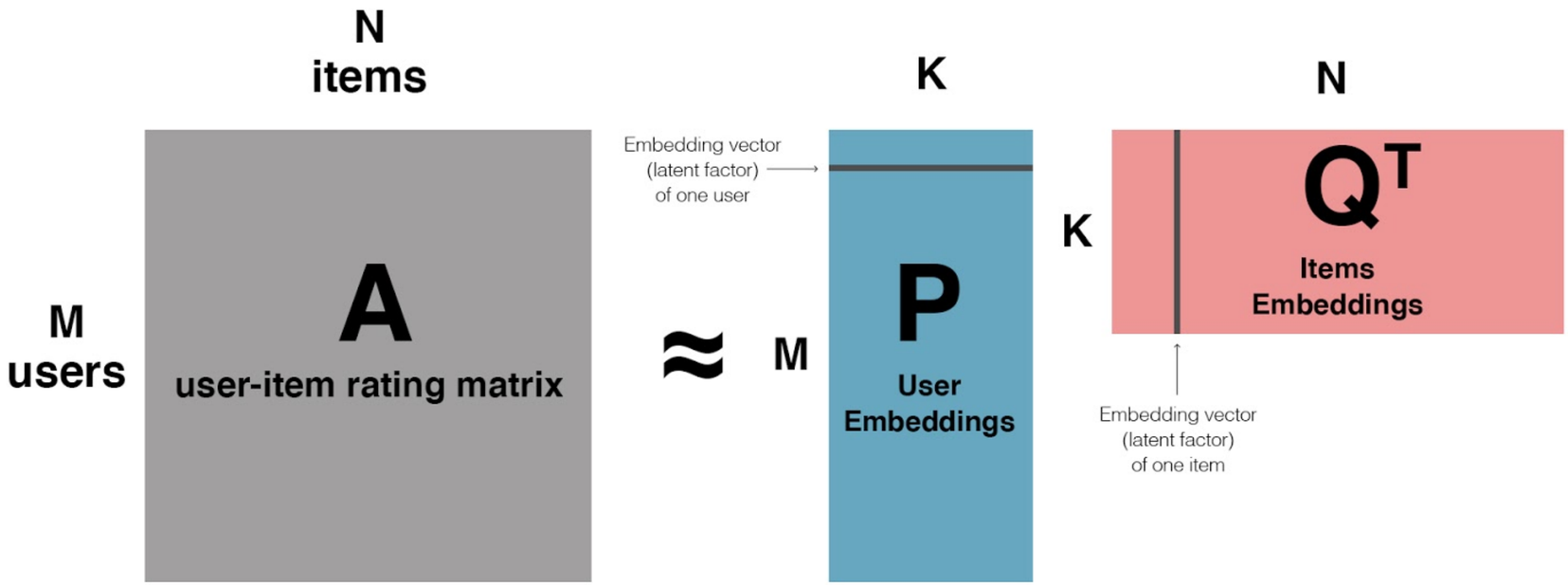
Collaborative filtering deduces recommendations based on the **user's past ratings** and other like-minded user's past ratings to find items of similar interest to the user

Utility Matrix

		Items					
		1	*2*	...	*i*	...	*m*
Users	*1*	5	3		1	2	
	2		2				4
	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		
	a	3	5		?	1	

utility matrix u: *user* $\times$ *item* $\rightarrow$ *ratings* where each cell r_{ui} corresponds to the rating of user u for item i, and it is required to find the rating of current user a which is r_{ai}

To establish recommendations, the collaborative filtering systems use latent factor models that transform both items and users to the same latent factor space. To create these embeddings we have used Matrix Factorisation.



The embeddings are learned such that the product PQ^T is a good approximation of the feedback matrix A. The (i,j) entry of $P.Q^T$ is simply the dot product $\langle U_i, V_j \rangle$ of the embeddings of user i and item j, which we want to be close to $A_{i,j}$.

The prediction r_{ui} is set as $\hat{r}_{ij} = \mu + b_i + b_u + q^T_i p_u$

Where,
μ is the global mean,
b_i is the item bias,
b_u is the user bias,
q is the item matrix,
p is the user matrix

To estimate all the unknown, we minimize the following regularized squared error:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui}^2 + \lambda (b_i^2 + b_u^2 + || q_i ||^2 + || p_u ||^2)$$

The minimization of this error function is performed by a very straightforward stochastic gradient descent.

2.1 Challenge

When a new user arrives, his/her user embedding has very few ratings. To recommend movies to the new user we cannot compute the matrix factorization all over again since it is an expensive computation.

2.2 Solution

Our scalable collaborative filter is based on incremental approach and supports updates while retaining the recommendation quality. So when new users enter the system they can be "folded in" directly and receive their recommendations instantaneously with good accuracy that is comparable to the base model.

2.2.1 Folding in a new user

We use the error function mentioned above and the gradient descent algorithm. But this time we keep the item matrix fixed and try predicting values only for the new user embedding instead of the complete user matrix, thereby reducing computation drastically.

$$bu \leftarrow bi + \gamma (eui - \lambda bu)$$
$$\mathrm{p}u \leftarrow \mathrm{p}i + \gamma (eui - \lambda \mathrm{p}u)$$

4. RESULTS AND EVALUATION

The class of metrics we decided to use for evaluation are the predictive accuracy metrics. These metrics measure how accurate or close the predicted ratings are to the real user ratings

Mean Absolute Error (MAE)

The MAE measures the average deviation between the predicted and real user rating.

$$\text{MAE} = \frac{\sum_{u,i} \left| p_{u,i} - r_{u,i} \right|}{n}$$

Root Mean Square Error (RMSE)

The RMSE is similar to MAE; however, it places more emphasis on large errors.

$$\text{RMS} = \sqrt{\frac{\sum_{u,i} (p_{u,i} - r_{u,i})^2}{n}}$$

4.1 Calculating the RMSE value for the testing dataset

The training data of rating is used to train the collaborative model to predict the rating for the given movie and the user and content model use movie data to calculate the rating.

The RMSE is calculated for the testing data, i.e, 25% of the total rating data.

For each movie rated by a user in a testing dataset, a rating is predicted by our hybrid by taking a weighted average of values from collaborative and content models.

RMSE value is calculated for all predicted ratings against the actual rating in the dataset.

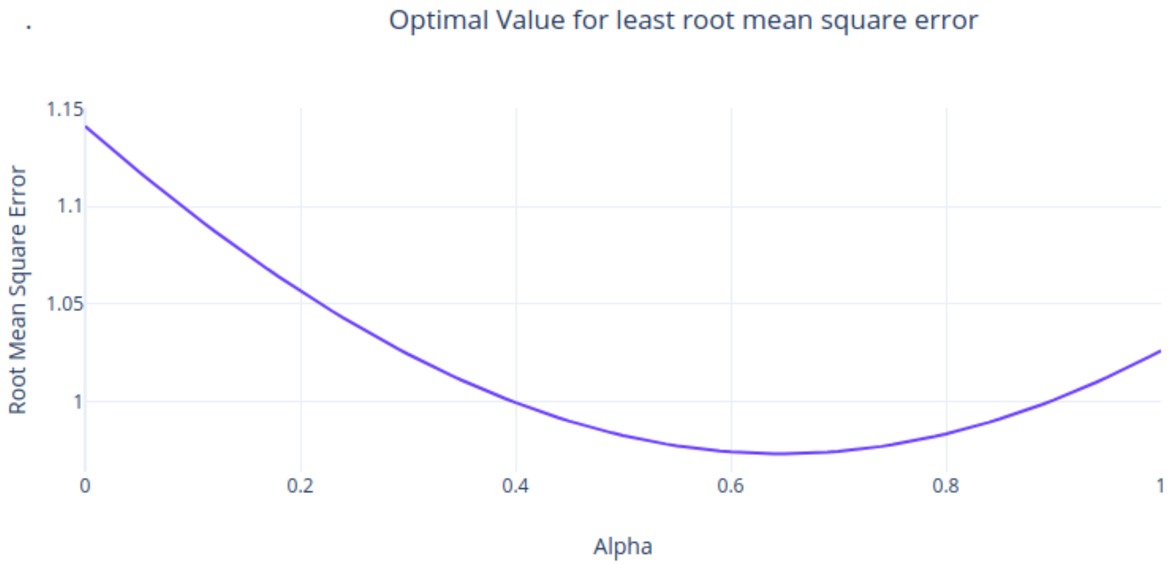
The RMSE and MAE values calculated for our hybrid model and the standard collaborative and content model alone are summarized:

Algorithm	RMSE	MAE
Collaborative Filtering	1.012	0.745
Content Filtering	1.296	1.082
Hybrid Filtering	**0.973**	**0.733**

4.2 Finding optimal value of *alpha*

In hybrid filtering the weight given to the rating predicted by the collaborative model is taken as *alpha* and the rating predicted by the content model is taken as (1 - *alpha)*.

To calculate the optimal value of *alpha* to get the least RMSE value, its value is varied from 0 to 1 and corresponding RMSE value is calculated for the testing dataset.



We get the minimum RMSE value for *alpha* as 0.65.

Significance

The *alpha* score of 0.65 implies that the score from the collaborative model has relatively more weightage than the content model in the hybrid model. This optimal *alpha* value may vary depending on the item for which the rating is calculated.

Discussion

The RMSE of the hybrid model is 0.973 this means that on average, its value is deviated from the actual value by 0.973 stars, for a 1-5 scale.

The MAE of the hybrid model is 0.733 this means that on average, it’s wrong by a value of 0.733 stars, for a 1-5 scale.

The RMSE value is less for the hybrid model than for the collaborative and content model individually. Hence by combining the results from both, we get better results.

5. CHALLENGES

Using subset of Data

The dataset Movie-Lens 25M has 25 million ratings for 62,000 movies by 162,000 users. For finding recommendations using collaborative filtering, we used a subset of this data because of computation power limitation right now.

Pivoting from NEO4j (Graph database) to Hybrid Filtering

The earlier proposed method of finding recommendations using **Random Walk** on the graph formed by NEO4j proved to be inefficient and less accurate than the hybrid filtering methodology.

FUTURE WORK

Parallelization

The collaborative score and the content score are calculated independently hence computation for these scores can be done parallelly.

Also, the cosine similarity value which is used in content filtering has to be calculated for many movies at a time. This calculation can be done simultaneously for multiple movies at the same time.

Development of a working web application

Support Folding in the new user as well as a new item simultaneously

THANK YOU