Personalized Movie Recommendation System

Shujia Liang, Lily Liu, Tianyi Liu
Stanford University

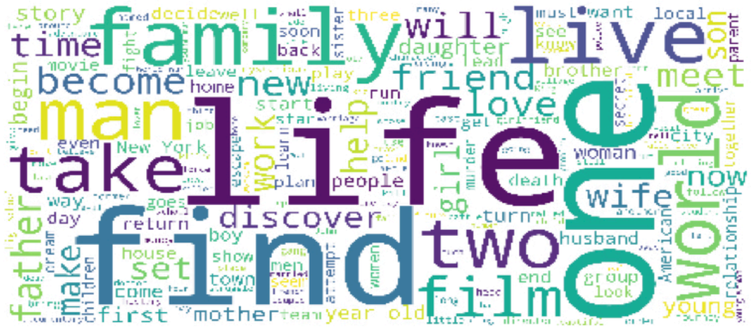
Background and Guidelines

• Predict users' rating for movies

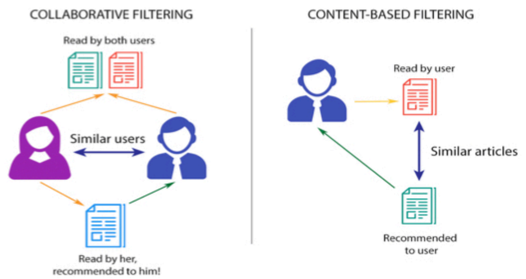
We predict users' rating for movies using techniques in two broad categories: content-based filtering and collaborative filtering. We found that collaborative filtering performs better than content-based filtering.

• Dataset and Features: Facts

A. Over 26 millions of ratings made by 180 thousands of users.
B. Features such as budget, genre, cast, director, etc., of over 45,000 movies
C. Obtained from Kaggle MovieLens data set [1].



• Personalize Recommendation System



Collaborative Filtering is a recommendation method without "understanding" the features of movies themselves. It requires large amount of user behaviors, and predicts users' behavior based on their similarity to other users. Specifically, k-Nearest Neighbors (KNN) and matrix factorization (MF) were implemented.

Content-based Filtering is a recommendation method based on both movie features and user behaviors. It compares movie features to the features of individuals' previous choices, and the closest ones are recommended. Specifically, term-frequency/inverse-document-frequency (TF-IDF) and an extended word2vec model, doc2vec, were implemented.

• Future work

We can combine collaborative filtering and content-based filtering: linearly combine ratings predicted by the two methods, incorporate features from one method into another. We would also add in a time aspect in our model to capture users' change of preferences over time.

• Result summary

Methods	TF-IDF	doc2vec	KNN	Matrix factorization
Test rmse	1.052	0.927	0.908	0.904

Collaborative filtering (KNN and Matrix factorization) generally performs better than content-based filtering

• Reference

1. https://www.kaggle.com/rounakbanik/the-movies-dataset/version/7
2. G. Salton. Automatic Text Processing. Addison-Wesley (1989)
3. https://radimrehurek.com/gensim/models/doc2vec.html
4. Y. Koren.Factor in the Neighbors: Scalable and Accurate Collaborative Filtering.
5. Y. Koren, R. Bell, C. VolinskyMatrix Factorization Techniques for Recommender Systems. Computer. 42. 8 (2009)

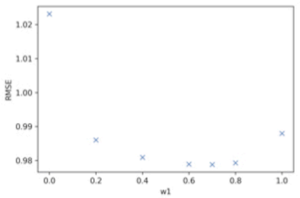
Content-based Filtering

• Method 1: TF-IDF

TF-IDF is a measure of the frequency of words [2]. Frequency of word **t** in document **d** weighted by the inverse of frequency of **t** in all documents is

$$TF - IDF(t) = TF(t,d) * IDF(d) = TF(t,d) log \frac{|D|}{1 + |d : t \in d|}$$

where |D| is the length of document and the denominator indicates the number of documents where t appears.



Rating predictions were computed by

$$\hat{r_{ui}} = \mu_u + \frac{\sum_j sim(i,j)(r_j - \mu_u)}{\sum_j sim(i,j)}$$

sim(i, j) is a weighted sum of two similarity matrices. Optimal weight for description and tagline is 0.7, and weight for actor/director names and keywords is 0.3. Results on test set shown on the right.

Each movie is described by two vector. One is TF-IDF of words in description and tagline, the other is TF-IDF of words in actor/director names and keywords. Similarity between each movie is calculated using cosine similarity. $(u_i \cdot u_j)/(|u_i||u_j|)$

Result is two matrices representing similarity between all pairs of movies.

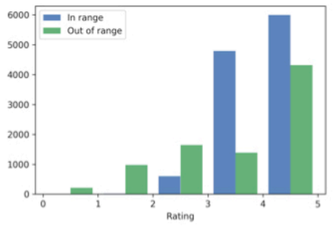
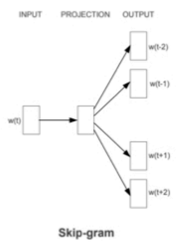


Figure: Histogram of predictions by TF-IDF similarity matrix, with "in range" equivalent to ±0.75

• Method 2: Doc2vec

In order to capture the similarity effect of synonyms and understand the context of words, we used doc2vec [3]. Doc2vec is an extension of word2vec that added a document-unique feature vector representing the entire document. All punctuations and words with no insightful information (a, the, and, of, etc.) were removed. We used the skip-gram model, which predicts the context of word from a given input word. We also introduced noise words into the model to improve the prediction robustness. We experimented with various learning rates, and arrived at the best rate of 0.025, which produces a test RMSE of 0.927.



Skip-gram

Movies most similar to Skyfall:

Similar Movies	Sim Score
Octopussy	0.8358
Transporter	0.8298
Safe house	0.8287
Unlocked	0.8106
Undercover Man	0.8062
Push	0.8033
Sniper 2	0.8007
Patriot Games	0.8005
2047 Sights Death	0.7952
Interceptor	0.7951

Test RMSE under different learning rate

Learning Rate	RMSE
0.020	0.927403
0.025	0.927003
0.030	0.927009
0.035	0.927358
0.040	0.927994
0.05	0.929483
0.1	0.938572
0.2	0.950744
0.4	0.970215

Collaborative Filtering

• Method 1: K-Nearest Neighbors

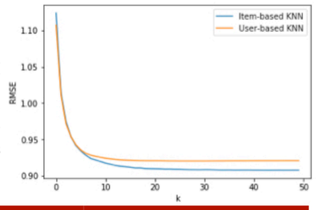
Both item-to-item and user-to-user collaborative filtering methods were implemented. In the user-based approach, we predicted user u's rating of movie i using a weighted sum of movie i's rating from the k nearest users based on their ratings' similarity score, biased by their own rating preference:

$$\hat{r_{ui}} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v)(r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

where $N_i^k(u)$ is the k nearest neighbors of user u's ratings.
Similarly, in item-based approach we adjusted for the movie i's average rating [4]:

$$\hat{r_{ui}} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j)(r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

For both item-based and user-based KNN, we searched for the best k value. The right figure shows that RMSE stabilized at k =20, but it does not monotonically decrease. The best k in terms of RMSE is 43 for item-to-item collaborative filtering and 28 for user-to-user collaborative filtering.

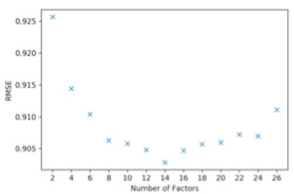


	Item-to-item	User-to-user
Optimized k value	43	28
Training RMSE	0.2900	0.2830
Test RMSE	0.9079	0.9203

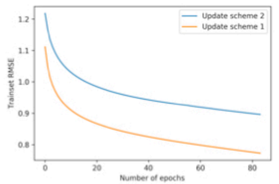
The differences between training and test RMSE suggest some degrees of overfitting, yet increasing k up to 300 did not reduce this gap.

• Method 2: Matrix Factorization

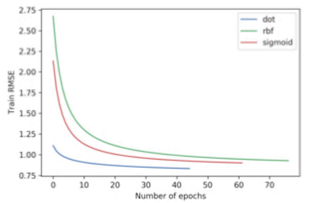
Users and movies are in the same vector space. Each entry of user vector p_u indicates a user's interest in certain characteristics of a movie. Each entry of movie vector q_i indicates the extent of a movie's characteristic. Estimate rating by $\mu + b_u + b_i + q_i^T p_u$ where μ, b_u, b_i are baselines for the movies population the specific movie, and user [5]. Learn all parameters by minimizing a regularized loss using gradient descent.
We tried different number of factors: dimension of p_u and q_i . Different update scheme: update all factors at once (scheme 1) or update one factor at a time (2). And different kernel for $q_i^T p_u$



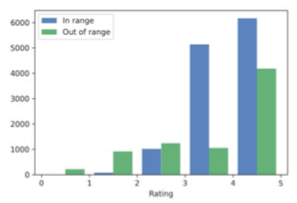
Each vector should have 14 features with regularization factor λ=0.01 to achieve lowest RMSE



Scheme with all factors are updated simultaneously converges faster.



Linear dot kernel performs better than other kernel types.



Histogram for prediction by matrix factorization with "in range" equivalent to ±0.75