MOVIE RECOMMENDATION SYSTEM

FINAL PROJECT

USING MOVIELENS DATASET

BY:

CHANDAN GARG (CG3176)

JAY SHAH (JS5553)

NEERAJ RAMKUMAR (NR2728)

SALONI GUPTA AJAY KUMAR (SG3910)

1. Introduction

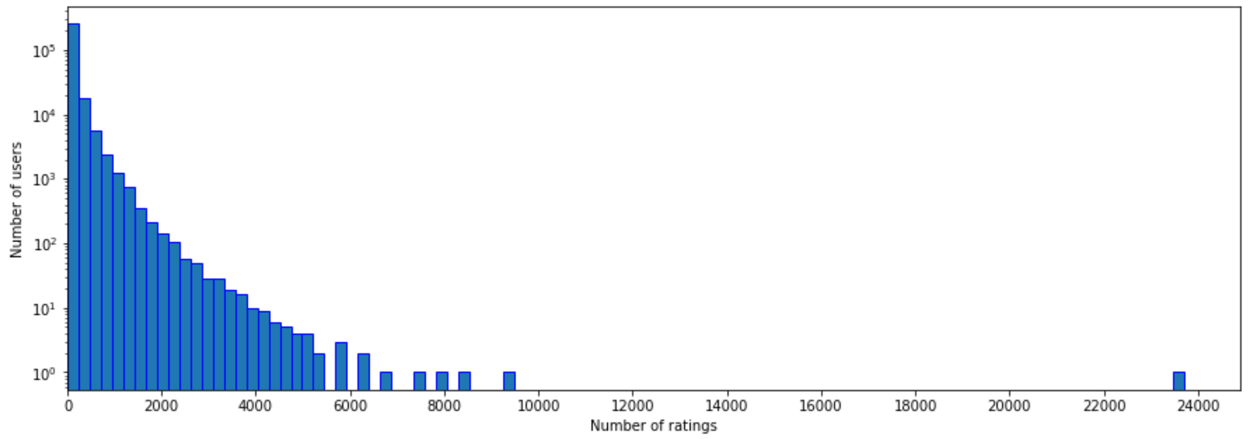
Recommender systems, in practice, are often complex systems which are built to cater to different segments of users. Different companies and different products prioritise different features such as addressing the problem of cold start, novelty, scalability, interpretability, and exploitation/exploration differently. However, one or more of these special attributes is often found in almost every recommender system that we use in daily life.

In this project, we improve upon our previous recommendation system; we will build an ensemble of more sophisticated models such as Factorization Machines, SVD followed by ANN and Neural Collaborative Filtering etc. These models are more generalised and flexible as compared to SVD Matrix Factorization and kNN. We will predict movie ratings provided by users in the MovieLens data set and recommend the top 10 movies to a user based on defined business objectives. Movies contain data about genres, tags dataset contain data about tags (short phrases) given by various users to the movies they have seen. A single movie can belong to multiple genres and has multiple tags.

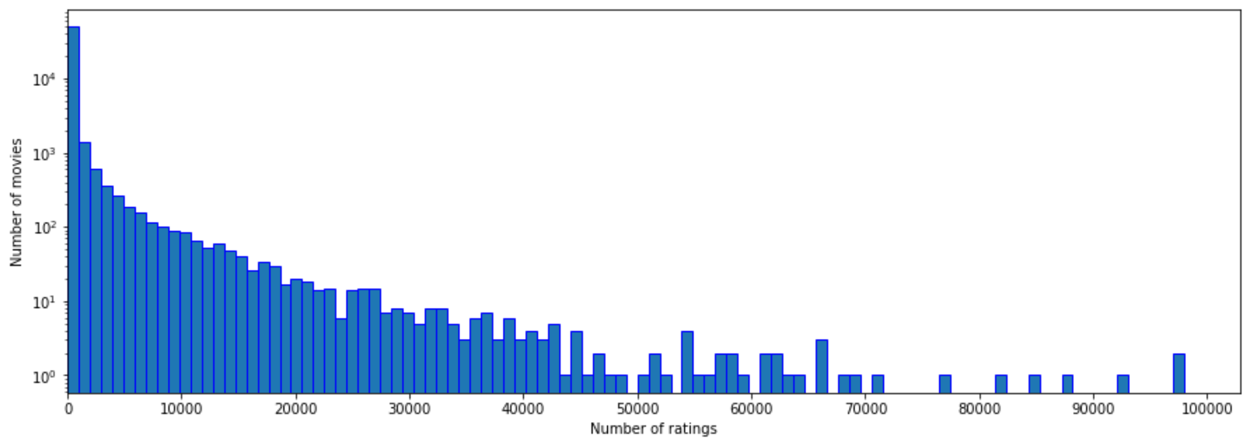
2. Exploratory Data Analysis

As discussed in the previous homework, we had found out that the total number of users in our MovieLens dataset are 283,228 and the total number of movies are 53,889.

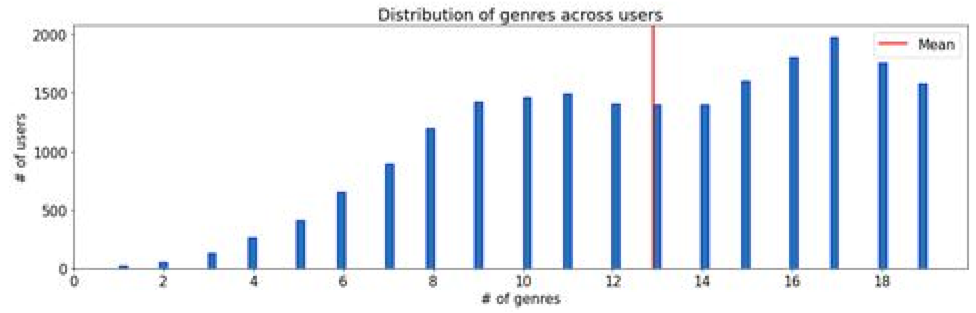
We had also observed a right skewed plot for the number of users versus number of ratings i.e. very few users have rated a very large number of movies.



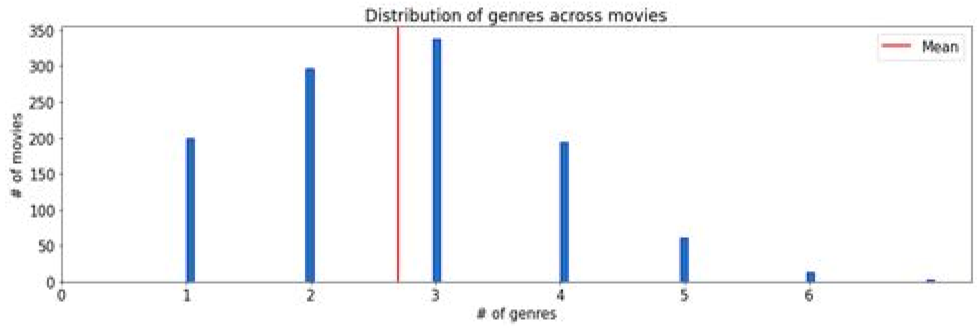
We observed a similar right skewed plot for the number of movies versus number of ratings i.e. very few movies have been rated large number of times.



Now, we also include metadata in the form of movie genres and tags. We found various interesting insights about how user and movie ratings distribute over genres:



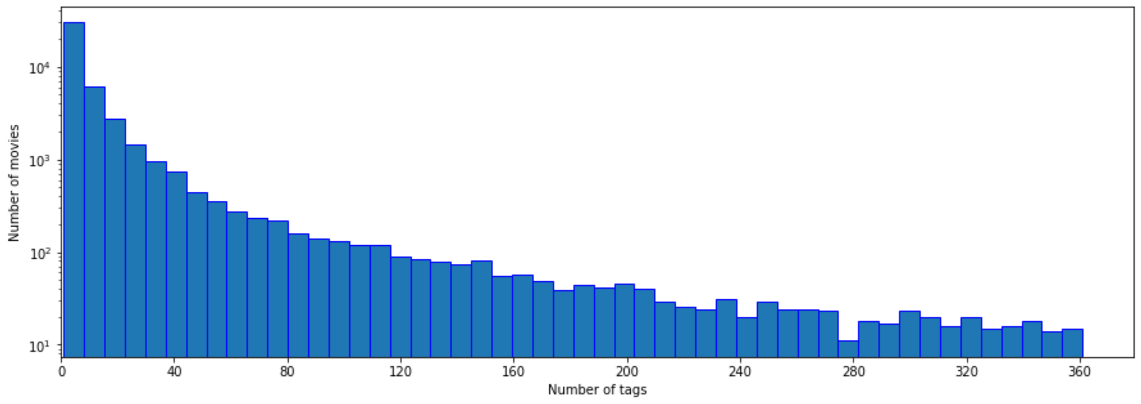
We observe a bi-modal distribution for # of genres vs # of users with modes at 9 genres and 17 genres respectively. On an average, a user has rated movies across 13 genres. We have 19 genres in total. This shows that our users have rated almost 70% of the total genres on an average indicating users exhibiting diversified interests.



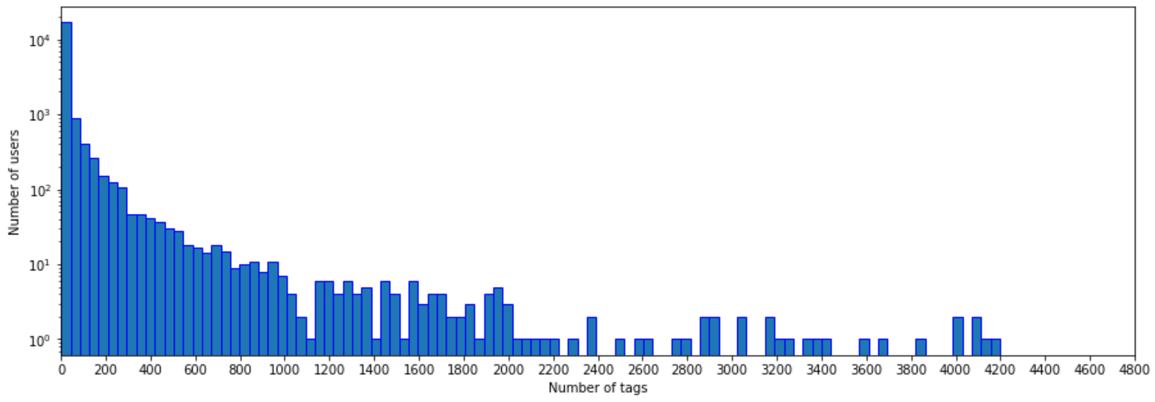
On an average, a movie belongs to approximately 3 genres out of 19 total genres. This is the reason that contributes to the fact that users have watched movies across a high number of genres even though our user vs number of movies distribution is right skewed.

We also use tags data for content based recommendation system which represent cleansed form of user reviews.

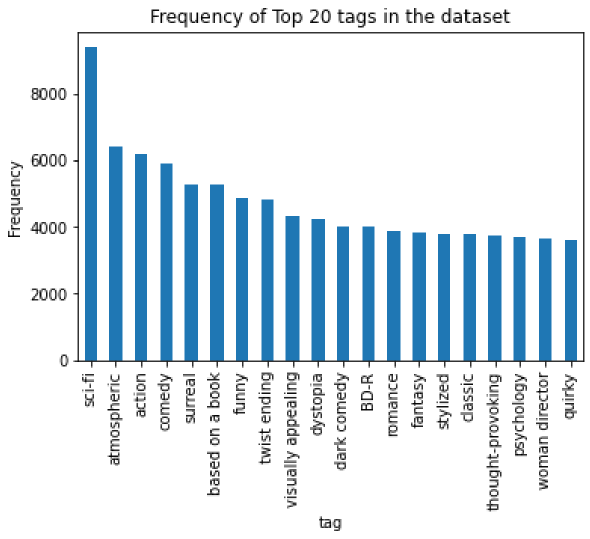
In number of tags vs number of movies plot, it shows a heavy tailed distribution which implies that certain popular tags have been assigned to a large number of movies.



Also, a large number of users (y-axis is on log scale) have reviewed movies instilling higher confidence in the quality of the tags data.



The plot below shows the frequency of the top 20 tags in the dataset. It is evident that these tags have a higher granularity of information as compared to genre data and also, the distribution of the movies across these tags is not skewed towards a few of them.



3. Data Sampling

We did an initial filtering of only those movies which had at least 1000 ratings and those users who had rated at least 10 movies. The motivation for this is that we want to build a data sample that has sufficient ratings across every movie. After this step, we obtained a dataset of 236649 users and 3914 movies.

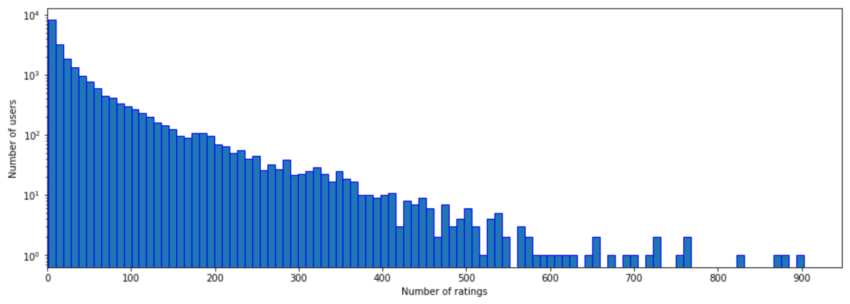
As discussed in the previous homework, we chose to sample our dataset using:

- Timestamp-based Sampling in combination with Random sampling

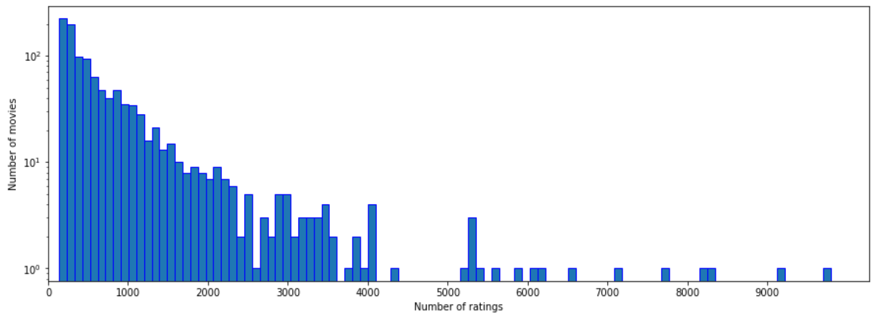
The motivation for sampling from the most recent ratings was that in a real-world situation, movies that have been rated recently are more likely to be seen given the current context of time and as they are more relevant to the current generation. We expect movies that have been rated in the past few years to better appeal to our users. So we chose movies that had at least 1000 ratings from 2005-2018.

To further sample our dataset, we used random sampling because we wanted to ensure that there is no bias against movies in our chosen time period. We first filtered the data to only give us user ratings that were given between 2005 and 2008. On this filtered dataset, we further chose the top 3000 most popular/highly-rated movies. The final layer of this sampling method used random sampling to create a sample dataset with approximately 21k users and 1100 popular movies.

We observe that the number of users versus number of ratings preserves a similar long tail distribution as explored in data analysis.



Similarly, we had preserved the trend for number of movies versus number of ratings as well.



Our dataset remains the same as in the previous homework.

4. Business Objective

The objective of our project is to improve the revenue of our platform through increasing the engagement of our users with the platform through providing them recommendations that they might be most interested in. Our focus remains serving users who have rated at least 3 movies with the top 10 movie recommendations based on the most recent relevant movies that the user has rated (not recommending those movies that the user has already rated). We want to consider those movies which have been rated after 2005 because most of our user base will consist of millenials .

In addition, we also aim to address the cold-start problem of a new movie. In simpler words, when a new movie is added to the database, which existing users do we recommend it to? Since new movies are released every other week, and are already in the public limelight at that point of time, they possess a great potential to bring benefits for the company. If the recommender system can leverage this advantage of their publicity, by suggesting it to the users who are most likely to watch it, the company can exponentially amplify the viewership base of its customers. We try to solve this problem through our content based recommendation system.

5. Evaluation Metrics

To build a good recommendation system, we should evaluate our model predictions through some evaluation metrics. Additionally, to understand whether we have built a good recommendation system, we have created two baseline models and evaluated our recommendation system's metrics against this baseline. Since we are more interested in the relative order of recommendations than the actual predicted rating, we considered the following 2 metrics to evaluate our models:

1. Normalized Cumulative Discounted Gain (NDCG) : This is a utility-based metric that is particularly relevant for our system. Calculated by normalizing DCG by the idealized

DCG (IDCG) to get the ratio of our model's performance to the best possible performance, where IDCG is found by calculating DCG on the empirically ranked items for each user.

2. Precision/Recall : Precision at k is the proportion of recommended movies in the top-k set that are relevant. Recall at k is the proportion of relevant movies found in the top-k recommendations. Relevancy is defined as the top-k movies the user has rated.

Beyond accuracy metrics:

1. Coverage - We try to understand the catalog coverage and user coverage in our sample dataset.
    - User coverage@k = The proportion of users that have at least k items recommended from the unrated items with predicted rating > 3.5.
    - Catalog Coverage@k = The proportion of movies recommended to users out of all the movies for all users with predicted rating > 3.5.

2. Novelty - We use Expected Popularity Complement (EPC) [8] which is the weighted probability of the items in the recommendation not to be known from the users in training defined as

$$\mathrm{EPC} = \frac{\sum_{u \in U} \sum_{r=1}^{N} \frac{rel(u,i_r)*(1-pop(i_r))}{log_2(r+1)}}{\sum_{u \in U} \sum_{r=1}^{N} \frac{rel(u,i_r)}{log_2(r+1)}}$$

Popularity of a movie is calculated as a ratio of the times a movie has been rated so far to number of ratings of the most rated movie. EPC takes (1-pop(i_r)) as a proxy for non-popular (pop(i) belongs to the range [0,1])

$$pop(i) = \frac{|Rat(i)|}{max_{i \in I}|Rat(i)|}$$

We take top_k as a measure of rel(u,i_r). EPC measures novelty in some sense similar to nDCG by assigning more weight to a top recommended unpopular movie.

6. Intended User

The architectures of our recommendation system is designed as a two component pipeline model (more details in section 8.1). In both the architectures, the first pipeline is to give top 10 recommendations to active users; definition as per business rules. The second pipeline gives out user recommendations for a new movie.

The two pipelines, running in parallel, cater to the following 2 types of intended users, independently-

1. The intended users for the collaborative filtering part (Pipeline 0) of our recommendation systems will be people who have rated at least 3 movies after 2005 and are active users of our product. The users with movie ratings less than 3 are used in training data but these users are not recommended any movies. Models based on collaborative filtering recommendation run on a regular interval, say, every 2 days to recommend movies to these intended users.

2. In addition, the users who are most likely to watch a new movie that is added to the database are also intended users. These users are served by the content based recommendation model of our system (Pipeline 1).

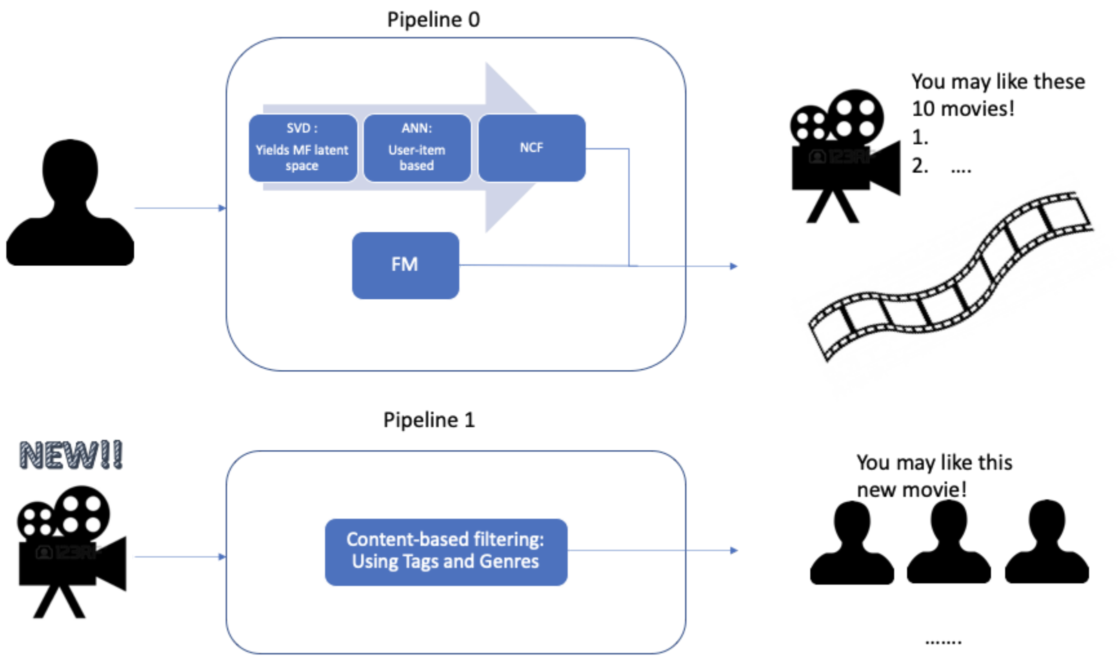
7. Business Rules

Business rules play a crucial part in the design of a recommender system. They help us implement rules that create a holistic recommendation system in tandem with machine learning models. We define the following business rules while designing our models:

- For users who rated less than 3 movies, we recommend the most popular movie from the genre that the user has already watched.
- If a user is recommended a set of movies in a particular week, we will not recommend any movie from that set the next week even if they are top recommended movies by our model. This is to minimize temporal repetitions
- If a user has watched 2 or more movies of a series, we recommend them the rest one by one.
- We give a user an option to “not recommend” a movie. If for a particular recommended movie, the user chooses “not recommend”, we do not recommend that movie as well as its nearest 10 neighbors in subsequent recommendations.
- To make sure our recommendations are not skewed towards only one or two genres, we cap recommendations from each genre to a particular number (cap to be decided with the product team)
- If we have age information of the users, we will recommend movies which are suitable for specific age groups. Also, if we have watchtime information about the users, we will recommend the movies which the user had left incomplete after watching more than 50% of its runtime.

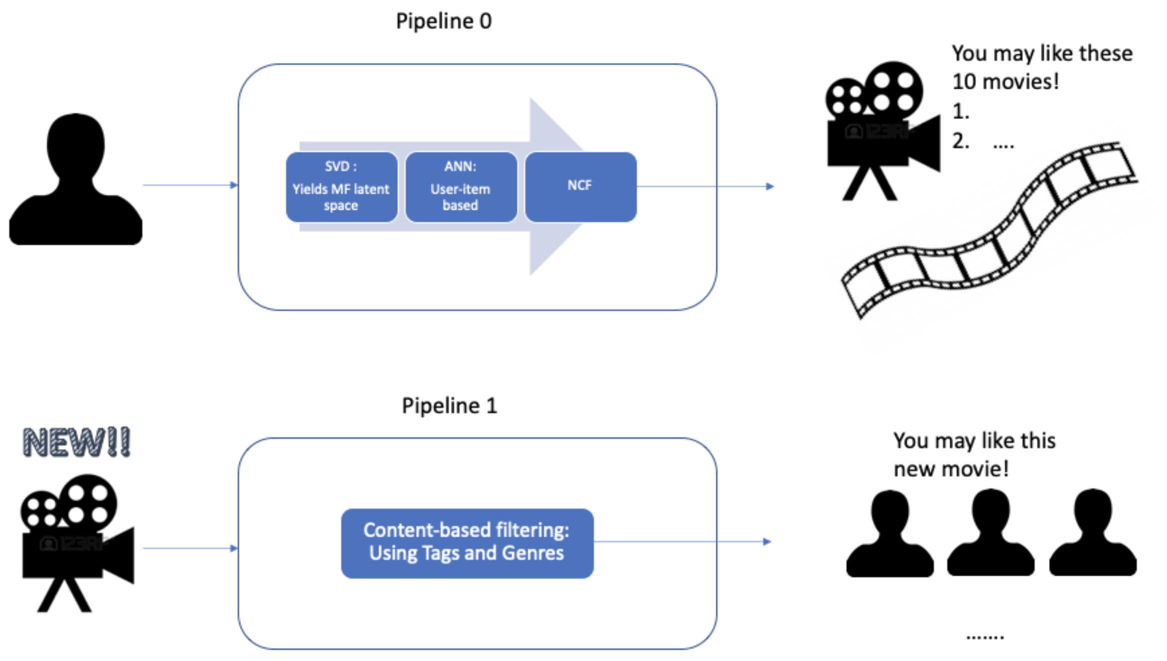
8. Recommendation Systems

8.1 Architecture



Pipeline 0
SVD : Yields MF latent space
ANN: User-item based
NCF
FM
You may like these 10 movies!
1.
2. ....
Pipeline 1
NEW!!
Content-based filtering: Using Tags and Genres
You may like this new movie!
....

Architecture 1 - Hybrid



Pipeline 0
SVD : Yields MF latent space
ANN: User-item based
NCF
You may like these 10 movies!
1.
2. ....
Pipeline 1
NEW!!
Content-based filtering: Using Tags and Genres
You may like this new movie!
....

Architecture 2 - Series only

We tried 2 architectures with different 'Pipeline 0's for discovering recommendation sets for users and the same 'Pipeline 1' for recommending new movies to the relevant users.

Architecture 1: Pipeline 0 is the ensemble model formed out of powerful collaborative-filtering frameworks such as Singular Value Decomposition (SVD), which generates User and Movie vectors in a latent space. This is followed by Approximate Nearest Neighbours (ANN) algorithm, which helps us to find the nearest neighbours based on User and Movie latent vectors, which finally produces a reduced dataset that is utilised by a Neural Collaborative Filtering (NCF) model.

We call SVD → ANN → NCF as our 'Series' model.

This Series model is finally used in hybrid with a Factorization Machine (FM) model. The final hybrid model is formed through a weighted average of predictions from the Series and the FM model.

NCF is an extremely powerful, but slower to train algorithm. To speeden up the training time, we decreased the train dataset size with minimal compromise on its quality (see plots in 8.3.1), through the usage of user-movie based ANN. FM is able to capture the global, user and movie biases along with interactions between variables, in the same model while the user-user based ANN is better able to improve the user coverage. We synergised on the strengths of all these models, and hence created a hybrid Collaborative Filtering model.

Architecture 2: Pipeline 0 consists of only the 'Series' model.

Pipeline 1 in both the architectures is the implementation of a Content-based filtering algorithm which uses movie metadata to recommend a new movie to the users who would be interested in watching it.

8.2 Baseline Models

The baseline model from the previous assignment was built using the mean rating of our dataset as the predicted ratings for all the movies in the entire data sample. Since this gives the same rating to all user-movie pairs, there is no notion of ranking. Hence, we were unable to give meaning to the NDCG metric as well as coverage for this model.

In order to solve this problem, we have now built 2 new baseline models.

First: Bias Baseline model, built using the inbuilt BaselineOnly basic algorithm in the surprise library. It predicts a baseline estimate for a given user and item [4].

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

Where μ global average of all items, b_u is the user bias which is the average score for a user across all his items minus μ, b_i is the item bias which is the average score across all users that have interacted with it, minus μ.

Second: Collaborative Filtering Model, built through a Probabilistic Matrix Factorization approach. Here, we assume a Multivariate Gaussian Distribution on the user and movie locations for our CF Model. We model the user and movie locations as follows:

For N_1 users and N_2 objects, generate

$$\textbf{User locations:} \quad u_i \sim N(0, \lambda^{-1}I), \quad i = 1, \ldots, N_1$$

$$\textbf{Object locations:} \quad v_j \sim N(0, \lambda^{-1}I), \quad j = 1, \ldots, N_2$$

Given these locations the distribution on the data is

$$M_{ij} \sim N(u_i^T v_j, \sigma^2), \quad \text{for each } (i,j) \in \Omega\,.$$

We are able to solve for each user location and movie location individually. Thus, an EM algorithm or ALS approach can be avoided. We can calculate the user and movie locations as follows:

$$
\begin{aligned}
u_i &= \left(\lambda\sigma^2 I + \sum_{j\in\Omega_{u_i}} v_j v_j^T\right)^{-1} \left(\sum_{j\in\Omega_{u_i}} M_{ij} v_j\right) \\
v_j &= \left(\lambda\sigma^2 I + \sum_{i\in\Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\sum_{i\in\Omega_{v_j}} M_{ij} u_i\right)
\end{aligned}
$$

However, we can't solve for all user and movie locations at once to find the MAP solution. Therefore, we have used a coordinate ascent algorithm to find the optimal user and movie locations.

Model	**NDCG (k=10)**	**Precision (k=10)**	**Recall (k=10)**
Bias Baseline	0.174	0.056	0.062
Probabilistic Matrix Factorization	0.107	0.037	0.034

Since we care more about order of recommendations as compared to the predicted rating, we treat NDCG as our primary metric for this use case and precision/recall at k as a secondary metric.

We have the plots for Coverage, NDCG, Precision and Recall later in this report.

8.3.1 Pipeline 0 - Collaborative Filtering Algorithms

We used the following algorithms in a series model with or without hybrid
(Architecture 1 and 2 respectively):

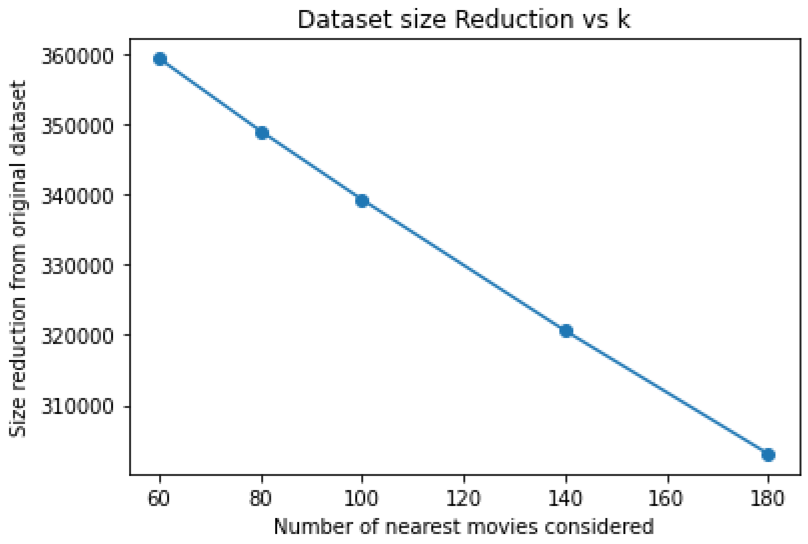
1. Singular Vector Decomposition (SVD) - A model-based CF algorithm using ALS - We used the famous SVD algorithm, which is optimized with Stochastic Gradient Descent, to build this model. SVD is a famous algorithm that was used heavily in the Netflix challenge; Netflix is a popular movie recommender system. Since we are limiting our data to approx 20k users and 1k movies, scalability is not the main concern for this project. If we are to use the full original dataset, we would be making use of Spark, which would work faster with large datasets. We have used the same hyperparameters which we optimized in the previous HW.

2. Approximate Nearest Neighbors (ANN) - We used the Annoy package from Spotify to implement ANN. Annoy uses random projections by building up n trees. At every intermediate node in the tree, a random hyperplane is chosen, which divides the space into two subspaces. This hyperplane is chosen by sampling two points from the subset and taking the hyperplane equidistant from them. n is tuned to the business need, by looking at what tradeoff you have between precision and performance.

We have built a user-item based ANN model, which finds out the k nearest items to a given user. We filter users by taking those that have a minimum of k ratings, and then use ANN to generate their nearest items. We look for the items among these k which are also present in the original user-item pairs present in train data, and only use them for training that user.

We finalized our hyperparameter k depending on the final dataset size that we found was necessary for our input to the Neural Collaborative Filtering model.

As shown below, the dataset size reduction is linearly decreasing with increasing k.



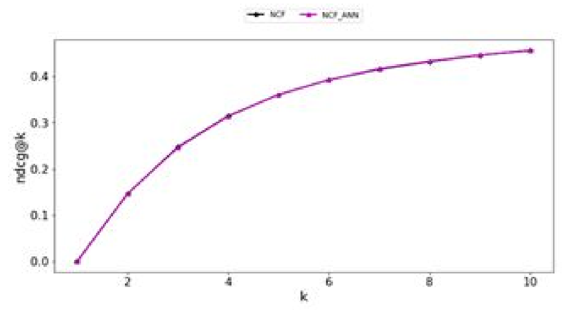
We choose our final k to be 100 based on hyperparameter tuning with our NCF model to obtain optimal results.

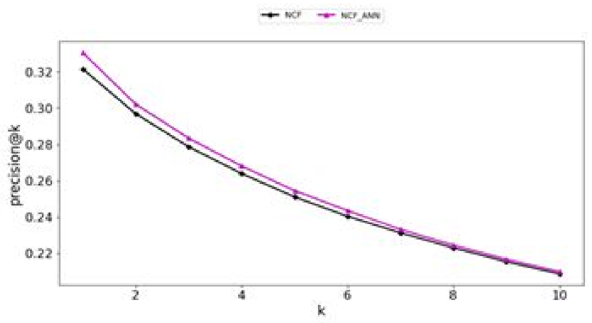
3. Neural Collaborative Filtering - NCF is able to learn any generic relationship between the vectors in latent space. NCF replaces the user-item inner product of MF models with a neural architecture that can learn an arbitrary function from data. NCF is generic and can express and generalize matrix factorization under its framework. [0] We use min-max standardization on the ratings since this model uses sigmoid activation function in the last layer. Since our ratings are standardized in the range 0-1, we chose binary cross-entropy as the loss function for the neural network. We trained NCF on the full dataset as well as the subset of the dataset obtained by running ANN model on 100 nearest neighbours based on user-item similarity from SVD latent spaces. For users who had less than 50 movies, we retained all the data for them. For the rest of the users, we reduced the data using ANN. This was done in order to not lose data for users who have already rated a lesser number of movies while removing lower rated movies of the prolific users. We observe that it is extremely time efficient to train on a subset as compared to the full dataset with a minor tradeoff on nDCG (shown in plots below for the

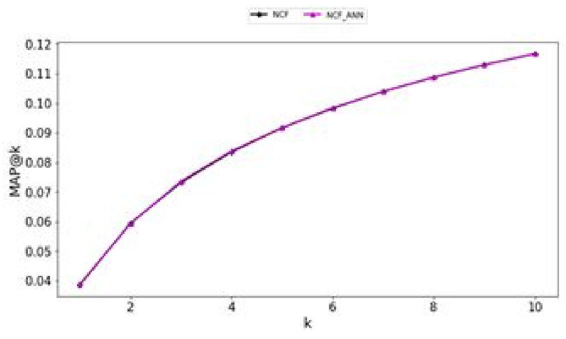
hidden layers with the best results). For both the datasets, the nDCG is higher for shallower networks as compared to relatively deeper networks. This could be attributed to the fact that deeper networks could be overfitting on the training data and hence worse performance on the validation data. The hidden layer structure on which both the methods performed the best have approximately equal nDCG, precision, recall and MAP as shown in plots below. It would be interesting to investigate if the data reduction through ANN also acted as a proxy for regularisation.

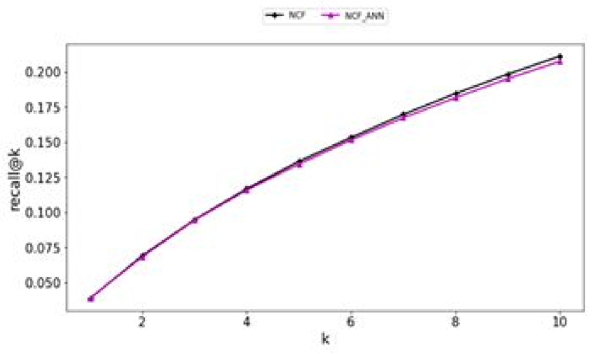
Hidden layers	nDCG (ANN followed by NCF)	nDCG(NCF on full data)
[16, 8, 4]	0.45	0.45
[32, 16, 8, 4]	0.27	0.29
[8,4]	0.37	0.39

* used microsoft's repository on recommenders for NCF









This whole pipeline of SVD + ANN + NCF is used in hybrid with FM in architecture 1 while used as a standalone series model in architecture 2.

4. Factorization Machine : Factorization machines (FMs) are a state of the art technique to problems including multiple entity types. They are similar to other collective approaches in that they allow for multiple relationships between multiple entities and are useful in capturing the

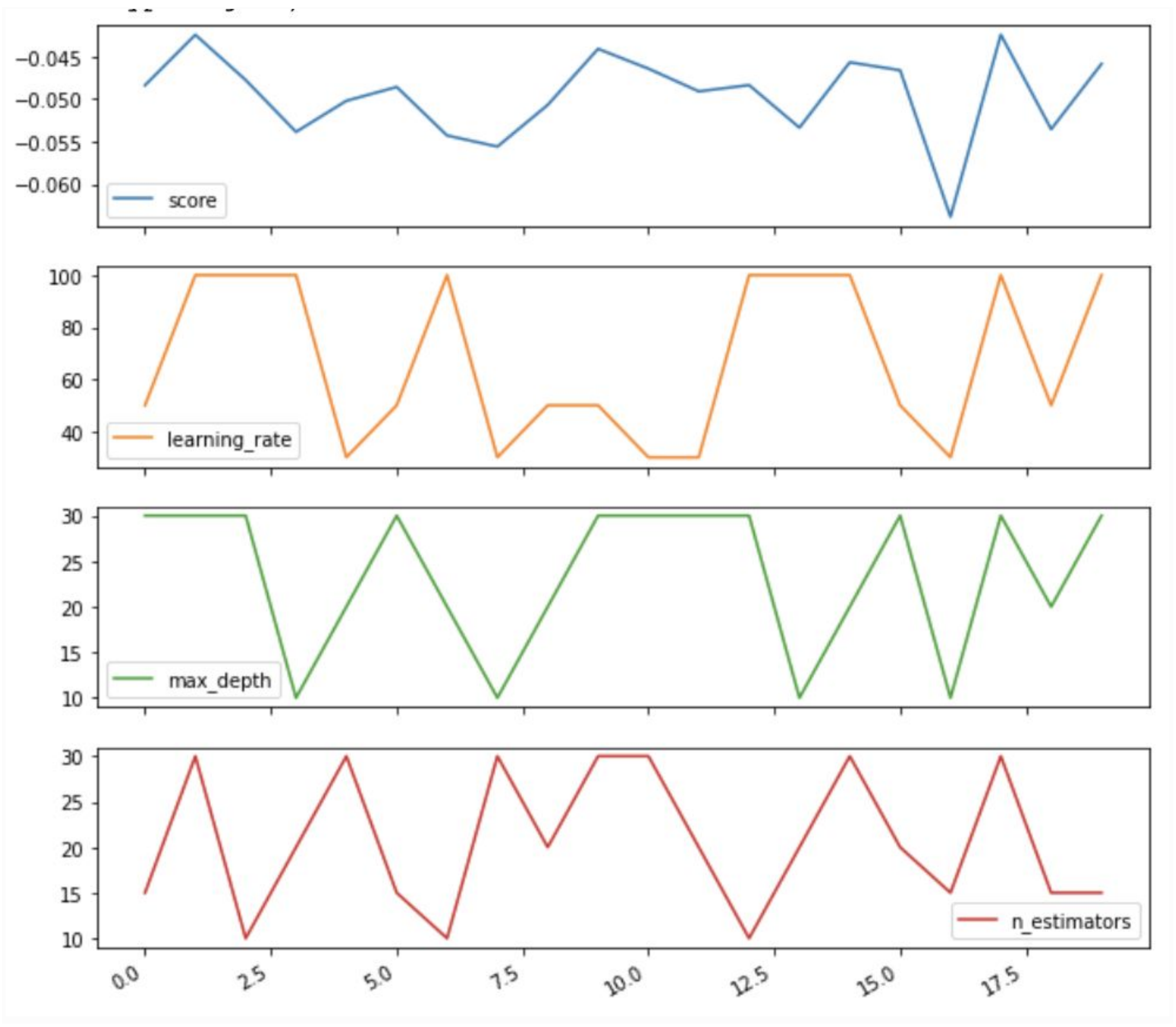
interaction effects between the variables. FMs use a clever feature representation to cast factorization as a regression, classification, or ranking problem.

We have used the LightFM package to implement the FM technique. The model learns embeddings (latent representations in a high-dimensional space) for users and items in a way that encodes user preferences over items. When multiplied together, these representations produce scores for every item for a given user; items scored highly are more likely to be interesting to the user.

We have used the WARP loss function for our implementation. It maximises the rank of positive examples by repeatedly sampling negative examples until rank violating one is found. It is a useful loss function to optimize the top n recommendations.

Parameters:
We used the Hyperopt package for hyper parameter tuning by bayesian methods as it is significantly faster than brute force grid search methods. The best parameters are:
'epoch': 30, 'k': 10, 'n_components': 15 as can be seen from the graph.

The graph shows the loss at all the trials of parameter combinations.



8.3.2 Pipeline 1 - Content-based Filtering Algorithm

The Content-based filtering algorithm aims to serve the cold-start movie problem for our system.

The algorithm is as follows:

- Create item/movie profile using movie metadata
- Creating movie embeddings using Doc2Vec
- Getting similar items/movie to a new movie using cosine similarity between the movie embedding vectors
- Mapping back similar movie to the users who have watched them
- Recommending the new movie to the above users found

We created movie metadata using a combination of all the genres that a movie belongs to, and all the tags given to movies by users. For every movie, we created a document which was the concatenation of genres and tags.

Before using the genre and tag metadata document, we applied a set of basic cleaning techniques to each word, such as stemming of each word, converting to lowercase, removing special characters, etc. This is a necessary prerequisite before generating the document embeddings.

To generate embeddings, we used Doc2Vec, which projects documents into a latent space, thus creating movie vectors. It is also known as Paragraph Vector. Doc2Vec is supposed to be an extension to Word2Vec such that Word2Vec learns to project words into a latent d-dimensional space whereas Doc2Vec aims at learning how to project a document[1].
We have implemented the Distributed Memory version of the Doc2Vec model using the gensim library.

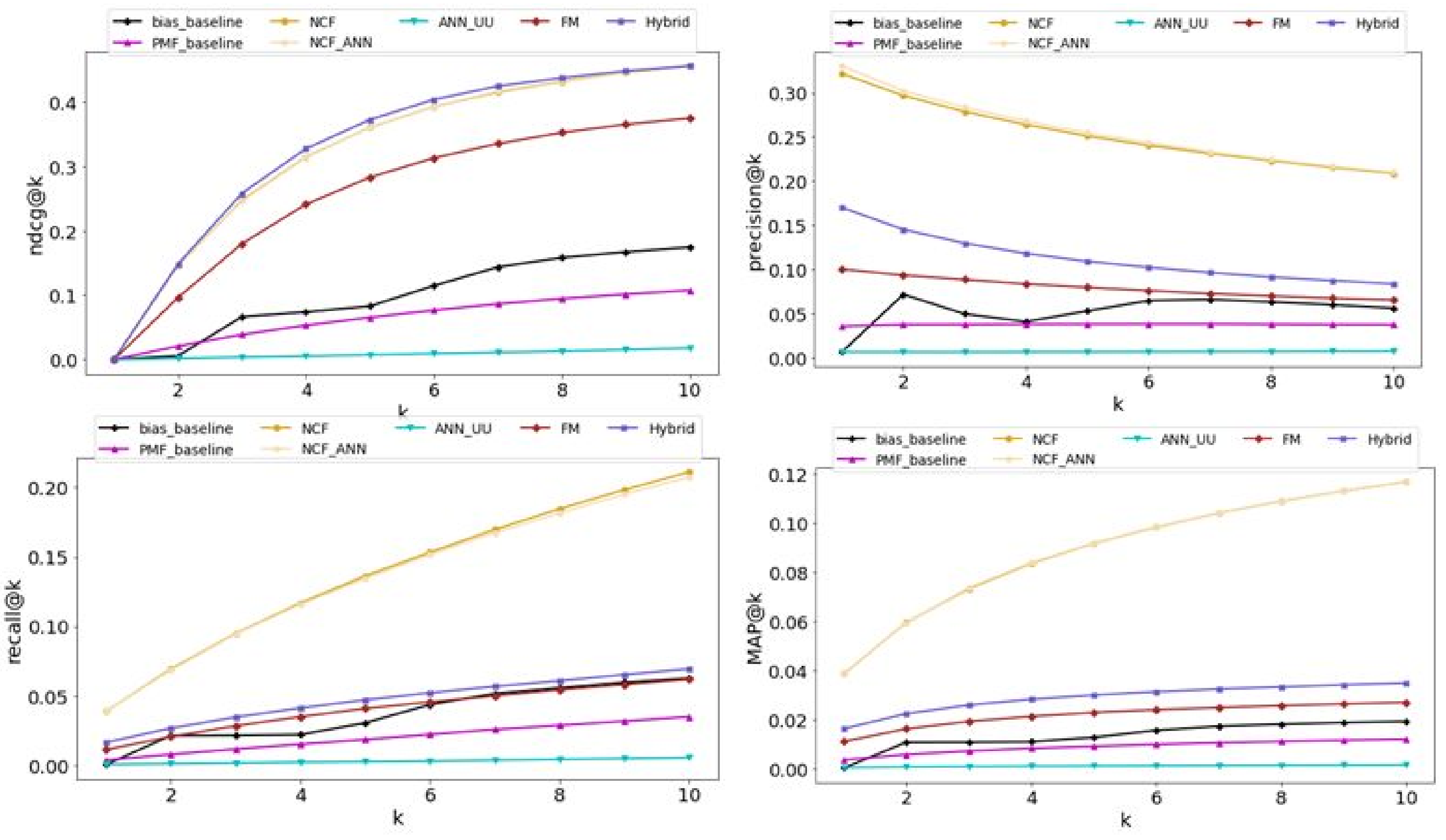
After learning the embeddings for all the movies in our dataset, we used cosine similarity measure to find the top-k similar movies to a particular movie.

11. Results

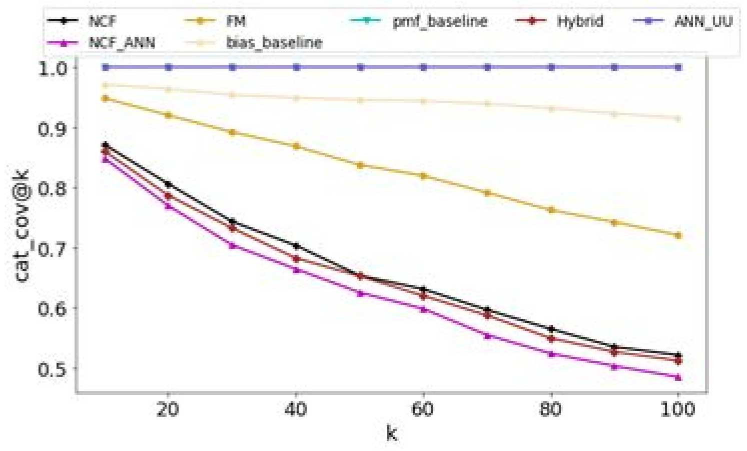
Results for all of the models we implemented for recommendation set of size 10:

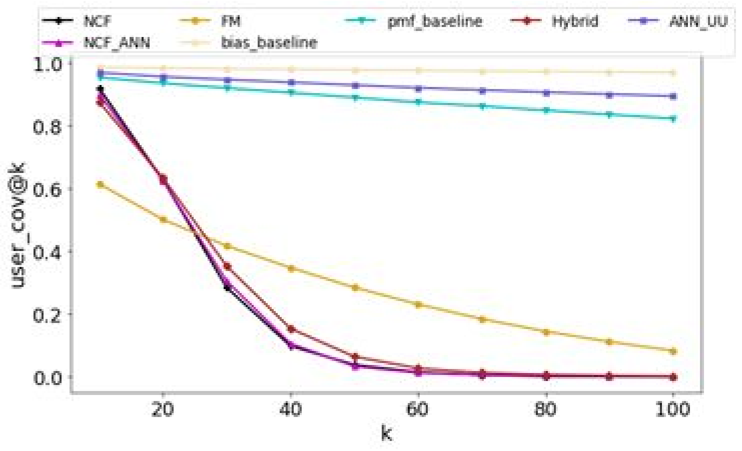
Models	nDCG @k=10	`Precision @ k = 10	Recall @ k = 10	Catalog Coverage	User Coverage	MAP @k=10	Novelty @ k =10
Bias Baseline	0.174	0.056	0.062	0.91	0.96	0.019	0.151
PMF Baseline	0.107	0.037	0.034	1	0.82	0.012	0.175
NCF	0.456	0.2	0.2	0.48	0.001	0.11	0.119
NCF-ANN	0.457	0.21	0.2	0.52	0.0014	0.11	0.113
FM	0.374	0.065	0.061	0.72	0.084	0.02	0.12
Hybrid	0.456	0.083	0.07	0.51	0.002	0.034	0.117
ANN-User -User	0.018	0.007	0.005	1	0.89	0.0015	0.2

nDCG, precision, recall, MAP (test data)

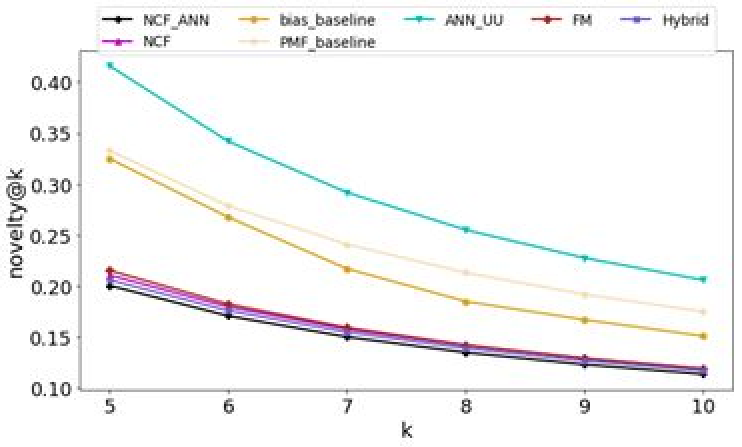


User and Catalog coverage





Novelty



We have observed the following insights from the above table and plots:

1. In general, the models are able to perform better than both the baselines except user-user based ANN which is why we removed it from both the architectures. In order to test the trade-off between recommendation set size and accuracy metrics, we plot all metrics vs k for k=1 to 10. As expected, nDCG, recall and MAP increases as we increase the recommendation set size k while precision decreases with k.

2. Both Hybrid and Series architecture have comparable performance in terms of the primary metric (nDCG) but the latter outperforms the hybrid model significantly in terms of precision, recall and MAP. The FM model has much lower precision-recall as compared to SVD-ANN followed by NCF. Even for the best weight distribution[5] for average in hybrid model optimized for nDCG, other metrics like precision-recall could not

do more than marginally better than the FM model alone. One possible solution could be to optimize the weight distribution on a combination of metrics as against nDCG alone. Hence, we finally consider Series architecture for production.

3. In general, NCF based models in both cases, trained on entire data as well as on filtered data through SVD-ANN(series architecture) gave better results on all accuracy metrics. An explanation for NCF based models outperforming all others is their ability to learn complex relationships between user and item latent spaces while all other models assume some sort of fixed functional form of variables or relationships between them which might not be true on real data.

4. When we test the quality of results beyond accuracy metrics, the user and catalog coverage plots show a very interesting insight. The models with higher accuracy (NCF based and FM) tend to show a steeper declining trend as compared to lower accuracy models (baselines and ANN with user-user similarity). This is in line with the fact that the low accuracy models may be providing more random recommendations, they are able to get higher user as well as catalog coverage by chance.

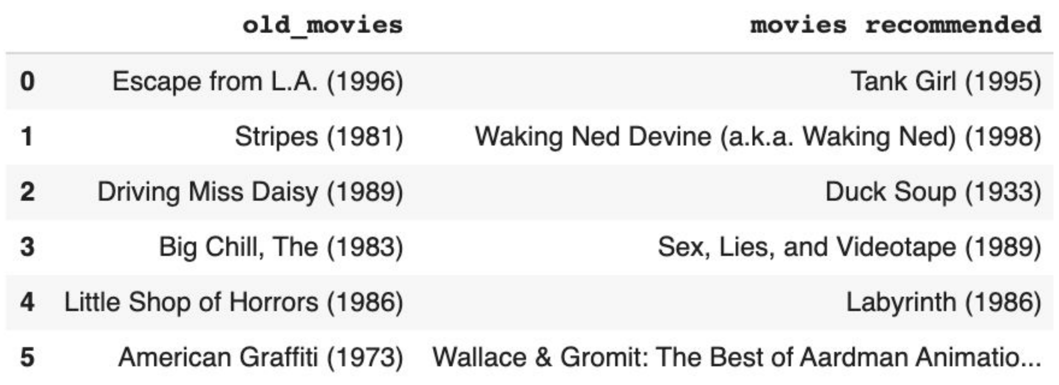
5. We have defined novelty in terms of weighted mean of unpopularity of items recommended. We observe that the user-user based ANN model is able to provide the most novel recommendations followed by NCF based models. In general, the novelty@10 is on the lower side because we didn't optimize on any proxy for novelty. This problem could be alleviated by using Greedy re-ranking algorithm which takes the weight of diversity into consideration for the objective function.

6. Interestingly, through the content based recommendation model, we are able to recommend movies from the same franchise and same starcast (see example below). Genres alone could not have provided this relevance at such granularity, hence underscores the importance of tags data.

Based on various trials across multiple models over extensive hyperparameter spaces, we have analysed the advantages and disadvantages of each of them. Since we care the most about nDCG, we suggest our manager use the Series model for production and the content based model for new movies with a caution that these models have a scope of improvement in terms of coverage and novelty as explained above and in the future improvements section.

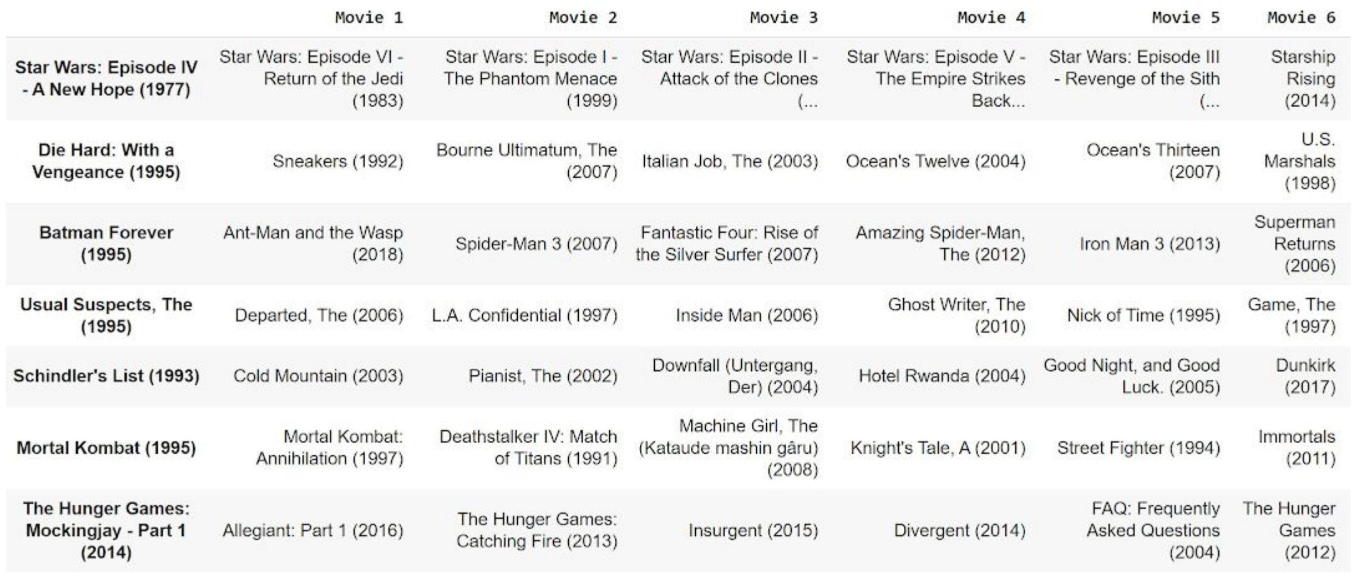
Following is an example of recommendations for a particular user from our Series Model:

old_movies are the movies seen by a user (in train data); movies recommended are from our Series model. This user seems to be interested in old movies and the model also recommends older movies to this user:



	old_movies	movies recommended
0	Escape from L.A. (1996)	Tank Girl (1995)
1	Stripes (1981)	Waking Ned Devine (a.k.a. Waking Ned) (1998)
2	Driving Miss Daisy (1989)	Duck Soup (1933)
3	Big Chill, The (1983)	Sex, Lies, and Videotape (1989)
4	Little Shop of Horrors (1986)	Labyrinth (1986)
5	American Graffiti (1973)	Wallace & Gromit: The Best of Aardman Animatio...

Following is an example of movies recommended by the Content Based Model. Movies on the Extreme Left (Index) are the query movies and movies shown in the columns are the movies recommended corresponding to the query movie.



	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
Star Wars: Episode IV - A New Hope (1977)	Star Wars: Episode VI - Return of the Jedi (1983)	Star Wars: Episode I - The Phantom Menace (1999)	Star Wars: Episode II - Attack of the Clones (...	Star Wars: Episode V - The Empire Strikes Back...	Star Wars: Episode III - Revenge of the Sith (...	Starship Rising (2014)
Die Hard: With a Vengeance (1995)	Sneakers (1992)	Bourne Ultimatum, The (2007)	Italian Job, The (2003)	Ocean's Twelve (2004)	Ocean's Thirteen (2007)	U.S. Marshals (1998)
Batman Forever (1995)	Ant-Man and the Wasp (2018)	Spider-Man 3 (2007)	Fantastic Four: Rise of the Silver Surfer (2007)	Amazing Spider-Man, The (2012)	Iron Man 3 (2013)	Superman Returns (2006)
Usual Suspects, The (1995)	Departed, The (2006)	L.A. Confidential (1997)	Inside Man (2006)	Ghost Writer, The (2010)	Nick of Time (1995)	Game, The (1997)
Schindler's List (1993)	Cold Mountain (2003)	Pianist, The (2002)	Downfall (Untergang, Der) (2004)	Hotel Rwanda (2004)	Good Night, and Good Luck. (2005)	Dunkirk (2017)
Mortal Kombat (1995)	Mortal Kombat: Annihilation (1997)	Deathstalker IV: Match of Titans (1991)	Machine Girl, The (Kataude mashin gâru) (2008)	Knight's Tale, A (2001)	Street Fighter (1994)	Immortals (2011)
The Hunger Games: Mockingjay - Part 1 (2014)	Allegiant: Part 1 (2016)	The Hunger Games: Catching Fire (2013)	Insurgent (2015)	Divergent (2014)	FAQ: Frequently Asked Questions (2004)	The Hunger Games (2012)

Recommendations from content based model

Shown below are the movies similar to “The Hunger Games: Mockingjay - Part 1 (2014)” and the distance between them.

```
similar_movies("The Hunger Games: Mockingjay - Part 1 (2014)",10)
```

```
[('Allegiant: Part 1 (2016)', 0.8808690905570984),
 ('The Hunger Games: Catching Fire (2013)', 0.8771207928657532),
 ('Insurgent (2015)', 0.8687823414802551),
 ('Divergent (2014)', 0.8524252772331238),
 ('FAQ: Frequently Asked Questions (2004)', 0.8320668339729309),
 ('The Hunger Games (2012)', 0.8226839303970337),
 ('Aeon Flux (2005)', 0.8107877969741821),
 ('Hulk (2003)', 0.7983793020248413),
 ('Ultraviolet (2006)', 0.7959374189376831),
 ('V for Vendetta (2006)', 0.7780249118804932)]
```

12. Attempts to improve Hybrid model using User-User ANN

We believed that our hybrid collaborative model could benefit with the addition of a parallel independent user-user ANN model. We implemented user-based ANN, as use-user similarity based models bring an increased novelty (see novelty plot) and serendipity quotient, and thus increased catalog coverage. Such models can help users discover new interests, which might otherwise not be discovered in a user-item based model.

In isolation, the system may not know the user is interested in a given item, but the user-user based might still recommend items that the user potentially likes, because similar users are interested in that item [6].

The steps to building the model were as follows:

1. We created User and Movie vectors in a shared latent space using our SVD model from the previous assignment. This step is a prerequisite for our next step.
2. We used Spotify's "Annoy" package to retrieve the k nearest users to a given user. We chose this because it is one of the most popular and fastest libraries among many other Approximate Nearest Neighbour methods.
3. We created our own prediction function by following this algorithm:

(1) $$\boxed{\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|}} \quad \forall u \in \{1 \ldots m\}$$

(2) $$s_{uj} = r_{uj} - \mu_u \quad \forall u \in \{1 \ldots m\}$$

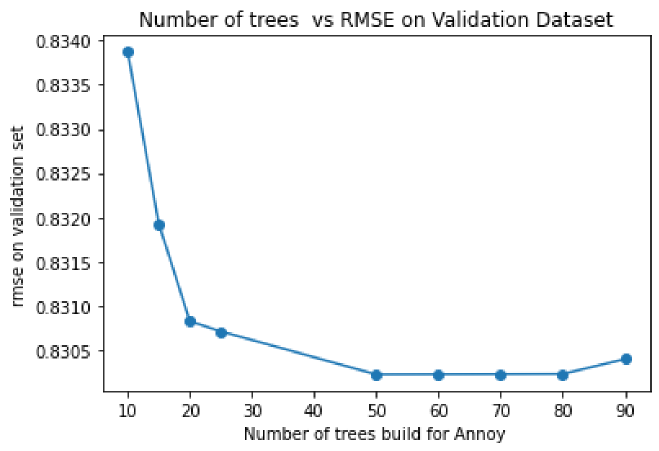
(3) $$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u,v) \cdot s_{vj}}{\sum_{v \in P_u(j)} |\text{Sim}(u,v)|}$$

$$= \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{Sim}(u,v)|}$$

where $P_u(j)$ is the *peer set* of *k* closest neighbors to user *u* that have a rating for item *j*

Here, we defined the Sim(u,v) as the cosine similarity found using 1 - (dist/2) where dist is the distance between u and v vectors.

Hyperparameter tuning on k, where k is number of trees:



We got a best RMSE value at k=50. Therefore we used k=50 for training our Annoy ANN model to find user-user similarities.
We got the highest novelty score among all models for the User ANN Model.
However, we noted that the NDCG value was noticeably lower than both of the Baseline models discussed in Section 8.2.
Below is the summary for comparison:

Model	**NDCG (k=10)**
ANN user-user similarity based	0.020644
Bias Baseline	0.174
Probabilistic Matrix Factorization	0.107

Since the models that we implemented in Pipeline 0 all gave an NDCG value better than our baseline models, the ANN user-user based was clearly not giving comparable performance. Since NDCG is one of our primary metrics in the system, we decided to exclude this ANN user-based model from our Hybrid model in Pipeline 0.

13. Future Improvements

We believe that both our models could be improved in many different ways, to yield a better and more robust recommendation system for movies. Here are some of our thoughts on possible future improvements:

1. Our recommender system has a low novelty score. We could improve it by incorporating weighted average of novelty score and a ranking metric (like nDCG) as evaluation metric while optimizing hyperparameter.
1. We could build a collective matrix factorization model to incorporate the genres of the movies into our model. We could then have control over the diversity of the genres of the movies being recommended to the user.
2. We can combine our existing rating dataset with additional data about user preferences, gathered through user surveys. We can ask them specific questions like age, favorite actor etc. to recommend a more personalized movie.
3. We could combine existing movie rating dataset with iMDB data to obtain metadata about movies which could be potentially used to add features and improve our models.
4. Implementing Reinforcement learning to solve temporal duplication as well as cold-start problems. [7] adopts the “explore & exploit” strategy to improve the recommending performance continuously, and therefore goes quickly through the cold-start stage.

References

0. He, Xiangnan, et al. "Neural collaborative filtering." *Proceedings of the 26th international conference on world wide web*. 2017
1. https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da
2. https://pypi.org/project/annoy/
3. https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401
4. https://surprise.readthedocs.io/en/stable/basic_algorithms.html
5. Suriati, S., Meisyarah Dwiastuti, and T. Tulus. "Weighted hybrid technique for recommender system." *J. Phys. Conf. Ser*. Vol. 930. 2017
6. https://developers.google.com/machine-learning/recommendation/collaborative/summary
7. Qiao, Runtao, Shuhan Yan, and Beijun Shen. "A reinforcement learning solution to cold-start problem in software crowdsourcing recommendations." *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2018.
8. https://dl.acm.org/doi/abs/10.1145/2487575.2487656