Movie Recommendation System

Abhinay Reddy Vongur
Rutgers University
abhinayreddy.v@rutgers.edu

Srujan Kashyap Dendukuri
Rutgers University
srujan.kashyap@rutgers.edu

Venkata Sri Ram Srujan Nidadavolu
Rutgers University
sriram.nidadavolu@rutgers.edu

ABSTRACT

With the abundance of information, there is a need for an effective filter which can be tailored to an individual. In this report, we explore the necessity of an effective recommended system and compare a neural network-based recommended system with various baseline models. For this purpose, the Movielens dataset was used, which consists of roughly 100,000 ratings; rated by 610 users from 1996 to 2018. In addition to comparing various models, we also provide direction for future extension and research.

KEYWORDS

Movie recommendation, Neural Collaborative Filtering, Joint neural Collaborative Filtering, SVD, KNN

1 INTRODUCTION

With the ever-growing need of providing user-centric services, the demand for personalized tools is more than ever. As such, many corporate giants like Amazon, Netflix, and Hulu rely on Recommended Systems to help filter information with regard to a user. Recommended Systems can be built to work on either Implicit Feedback or Explicit Feedback.

Implicit feedback does not rely on a user providing direct input with respect to their preferences but instead, collects information through user behavior, like browsing activity, purchase history, or the number of times a page was visited. Explicit feedback as the name suggests is the quantification of a user's preference with respect to an item such as movie ratings or product ratings. Explicit data is direct and considers a user's preferences but explicit data is prone to bias.

Implicit data on the other hand is relatively easy to collect and reduces the dependency on the user.Apart from the type of input, recommended systems can be broadly classified into Content-Based Filtering Systems and Collaborative Filtering Systems.

Content-Based Filtering Systems rely on the attributes of the items to provide recommendations. Therefore, the items that a user likes are tagged such as genre, director, or the cuisine of a restaurant or manufacturer of a product. Then, items with similar tags are recommended to the user. For this project, we have worked on collaborative filtering systems and hence will be discussed in detail.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Collaborative filtering systems rely on the history of user-item interactions and don't enforce the extensive knowledge of items themselves to provide recommendations.Collaborative filtering systems can be further classified as memory-based and model-based.

Memory-based approaches like KNNs [1] (k Nearest Neighbors) work directly with the user interactions without assuming any model which can explain these interactions. The term memory stems from the fact that the entire data needs to be available in memory for training. Memory-based models can be trained to work on user-user or item-item.

For each user, an interaction profile is built[5]. A user-user model tries to identify other users with the most similar interaction profile (simply termed as neighbors). Items popular among these neighbors are provided as recommendations to the user. For filtering, the similarity is calculated with each neighbor, and a weighted average of ratings is given to an item based on their similarity.

Item-item models have a similar notion, except here, items that are positively interacted with or rated by the user are collected and recommended. Just as in user-user models, we find similarities between items to filter and sort recommendations. In this project, we have implemented a User-based collaborative filtering model with KNN.

Model-based approaches assume there is a model capable of explaining interactions between users and items. We implemented a matrix factorization-based approach which will be discussed in detail in section VI.

The model-based matrix factorization approaches though perform well have their limitations as they use the inner product to model the interactions between users and the movies. This limits their ability to learn from non-linear data. Neural networks on the other hand can learn from non-linear data and don't rely on any specific technique for matrix factorization. This provides a generic approach to the matrix factorization problem.

Deep Learning-based recommender systems can be categorized generally into areas, Single Neural Network models and Deep Integration Models. Single Neural Network models rely only on deep learning techniques to learn while Deep Integration Models combine traditional techniques with Deep Learning techniques to learn

from the data.

Neural Collaborative Filtering[2] is an approach of the second category, it combines the traditional technique of matrix factorization with Multi-Layer Perceptrons to enable learning. Joint Neural Collaborative Filtering[3] is a variation of the Neural Collaborative Filtering approach. J-NCF uses a Joint neural network structure to learn representations of users/movies and learn interactions between them. J-NCF uses a non-linear kernel to model the user-movie interactions. The model we propose in this work is based on J-NCF.

3 EXPLORATORY DATA ANALYSIS

3.1 Dataset Description

The dataset used for this project describes 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org), a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996, and September 24, 2018.

3.2 Data Preprocessing

The dataset was split into 80:20 ratio. For each user, 80% of their ratings were included in the Training set and 20% were included in the test set. The tags.csv contained the tags that users applied to different movies. The movies.csv file contained Movie Title and Genre information. We used this information to create vector representations for all the movies.

For each movie, we collected the title, genre tags, and tags given by the user and grouped them into a sentence that describes the movie. Then we trained a Word2Vec[8] model to learn the vector representations of the words. Next, we used the word2vec model to find vectors for all the words in the sentence of a movie and averaged all the vectors. We call this averaged vector, the vector representing the movie. This approach is based on the method described in [7].

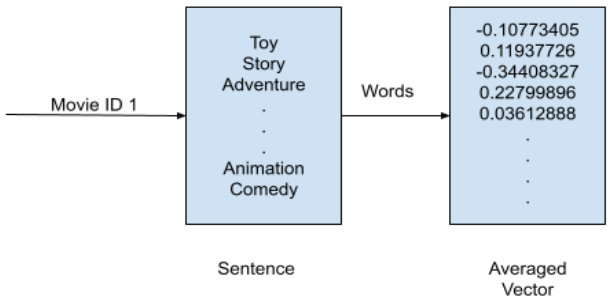

Figure 1: Movie Vector Generation

3.3 Analysis

The following plot shows the distribution of ratings by all the users.

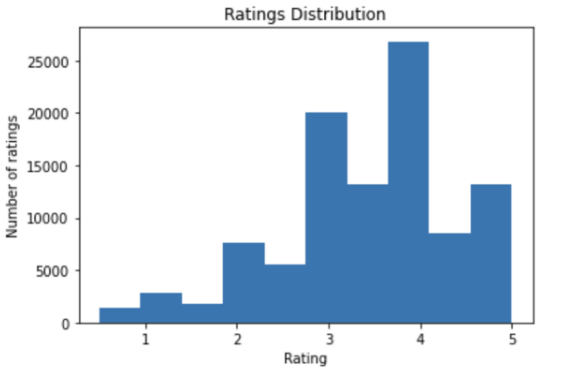

Figure 2: Histogram of ratings

Comedy and Drama constitute a majority of the dataset with 15.3% and 14.2% of the total movies respectively. Documentaries constitute the least with 0.018%

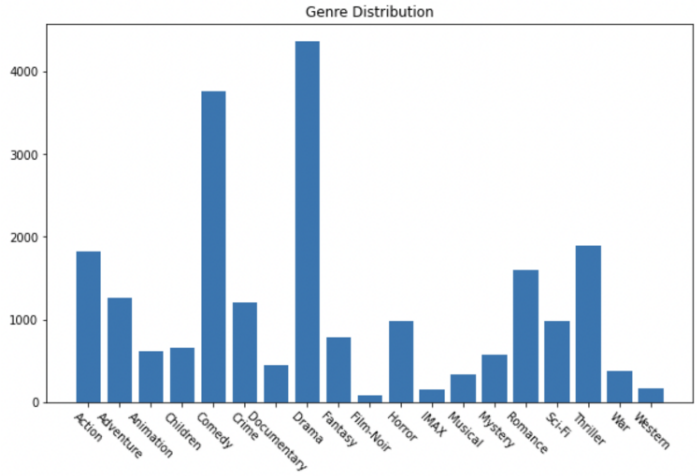

Figure 3: Distribution of Genres

Using the timestamp data, we observed how users rated movies of different genres. The graphs corroborate with years when popular movies of a particular genre were released.

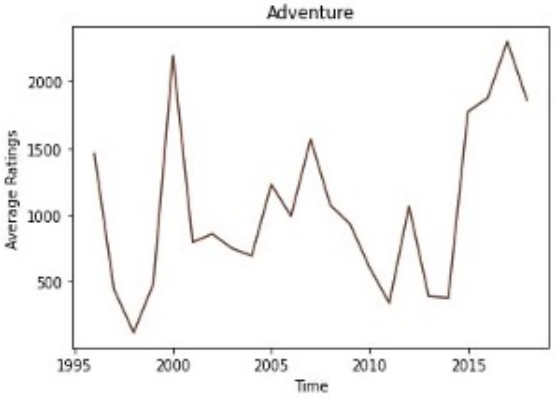

Figure 4: Number of ratings for Adventure genre over the years

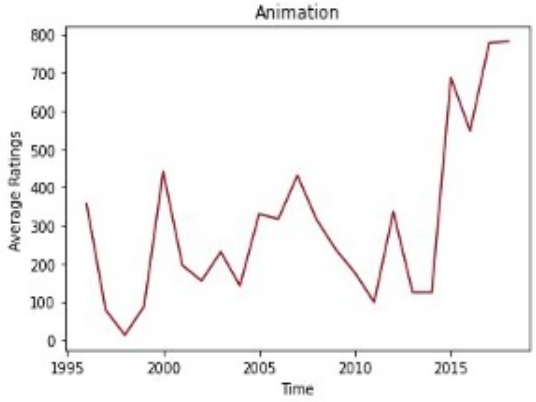

Figure 5: Number of ratings for Animation genre over the years

4 PROBLEM FORMULATION

The problem that we explore can be formulated as one of rating prediction where given n users and m items and if r_{ij} is the rating given by i^{th} user to j^{th} item. We predict ratings for items the user has not interacted with and provide Top-N recommendations based on the predicted ratings.

5 THE PROPOSED MODEL

In this project, we propose a deep learning model that can predict ratings and provide recommendations. The model is based on Joint Neural Collaborative Filtering[3], and learns from the explicit feedback data of users and movies. The model consists of three grouped networks in total. Two parallel networks to learn feature representations of users and movies which is called the Feature Learning Network and one network to learn interactions of users and movies called the Interaction Learning Network.

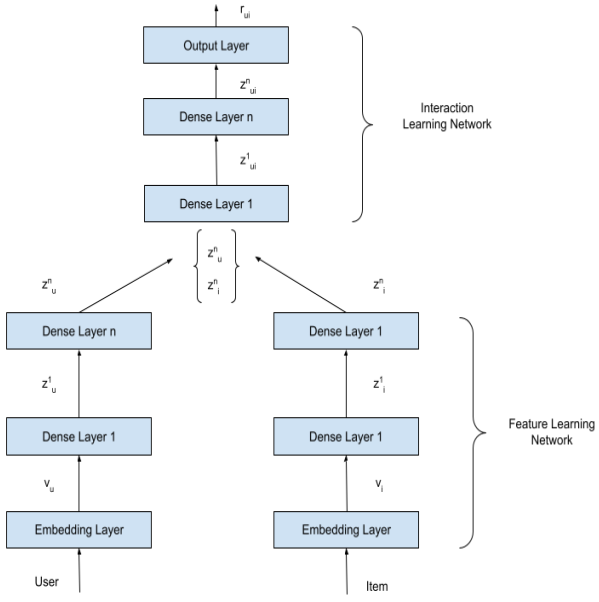

Figure 6: Network Architecture

The Feature learning network takes the identity of the user/movie as input. It then goes through an Embedding layer which generates a 100-bit embedding v_i/v_u for the user/movie. This embedding is then fed to Multi-Layer Perceptrons (MLP) which learn to extract latent feature vectors z_u^n/z_i^n for users/movies.

$$z_{ui}^0 = \begin{bmatrix} z_u^n \\ z_i^n \end{bmatrix}$$

These feature vectors of user and movie are then concatenated as shown above and passed to the Interaction Learning Network which tries to learn the interaction between users and movies through a series of dense layers. The output layer is a linear layer that gives rui the predicted rating user u will give to the movie i. We have used ReLU as the activation function in the dense layers.

$$f(z_{ui}^0) = r_{ui}$$

Since we are more focused on predicting ratings, we have adapted a point-wise learning approach. The point-wise learning involves calculating the distance between the predicted value and the true value, then using that to optimize the model parameters.We have used Mean Squared Error as the loss function in our experiments.

We also experimented with a slightly different model by adding the content information of the movies as an input to the Interaction Learning Network. We generated a vector representation for each movie. This movie vector was then given as the additional input to the interaction learning network. The results of these two models will be discussed in the results section.

6 EXPERIMENTS

To evaluate our proposed model, we first built two baseline models. The first model is a memory-based approach where we used KNN to build a user-based collaborative filtering system. KNN works by first building an interaction matrix, which is just a pivot table with users as rows and movies as columns [7]. When a recommendation needs to be made for a particular user, k neighbors are identified on the basis of the interaction matrix. The neighbors are identified using a similarity measure; in our case, it was cosine similarity.

Cosine Similarity of two users A, B is given as

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{m} A_i B_i}{\sqrt{\sum_{i=1}^{m} A_i^2}\sqrt{\sum_{i=1}^{m} B_i^2}}$$

Where i iterates over all of the movies rated by the users. The intuition is to recommend movies watched by other users whose tastes are as similar as our target user. After identifying the neighbors, their Top-N movies are picked and a weighted average is calculated on the basis of similarity. Then the recommendations are filtered by these predicted ratings and provided to the target user. Though KNN is fairly simple to implement, KNN suffers from the shortcomings of any other memory-based model that is to store and utilizes the entire interaction matrix for prediction. Since this approach does not assume any model that can explain this data, it is harder to find similar users if two users have not seen any movies in common. To overcome this challenge, we explored a model-based approach.

In this approach, we develop a popular matrix factorization-based model known as the Singular Value Decomposition (SVD).

Model	RMSE	MAE	PRECISION	RECALL	NDCG	F1 Score
KNN	0.94	0.72	0.61	0.30	0.65	0.402
SVD	0.87	0.69	0.60	0.29	0.71	0.39
SVD with grid search	0.85	0.65	0.62	0.31	0.71	0.413
Deep Learning Model	0.79	0.60	0.64	0.32	0.70	0.426
Deep Learning Model with content	0.90	0.70	0.59	0.29	0.69	0.388

Table 1: Metrics for different Models

As described in section II, SVD assumes an underlying model [6] and tries to explain the data. KNN suffered in part due to the sparse nature of the interaction matrix. SVD overcomes this by decomposing it into three sub matrices which are of lower dimensions and are dense in nature.

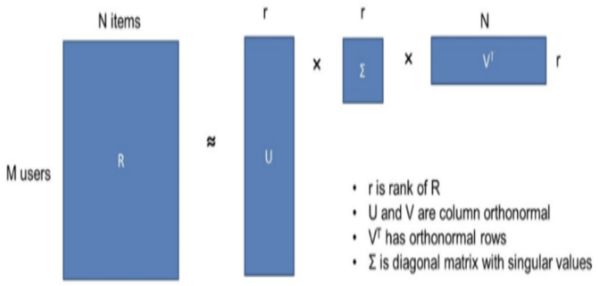

Figure 7: SVD Decomposition

As shown in figure 1, the utility matrix is decomposed into U_{nxr} a user-latent feature matrix, $r \times r$ a diagonal matrix that defines the strength of each latent feature and V_{rxm} a movie-latent feature vector. To provide recommendations for a user i, the corresponding vector in the U matrix is picked and the dot product is calculated with feature vectors of movies they did not watch yet. The results of these dot products are sorted to provide Top-N recommendations. In SVD, users and items are mapped to an r-dimensional latent space. SVD explains how much a user is aligned with a set of latent features and how much a movie fits within those features. An advantage of SVD over KNN is that even if two users have no movies in common, it is still possible to find the similarity based on the latent vectors.
SVD has the property of having minimal reconstruction error which translates to minimal RMSE. However, SVD is not without any drawbacks. For one, SVD cannot explain the model that it assumes and, therefore there is no deeper understanding of the recommended items. Secondly, SVD does not map well to non-linear data which is where our neural network-based model comes in.

For the purpose of experiments, we have used two layers in the feature learning network and three layers in the interaction learning network.

For evaluation, we have used the following metrics

i. Precision: Precision measures, of the Top-N, recommended items, and how many of them are actually relevant to the user.

$$\text{Precision} = \frac{|\text{Relevent in Recommended}|}{|\text{Recommended}|}$$

Precision of 60% (SVD with Grid Search) implies 60% of recommended items are relevant to the user.

ii. Recall: Recall is the proportion of relevant items found in the returned Top-N set. This is a metric that we want to maximize. From Table I, we can observe that the recall score for KNN is 30% which implies that 30% of the relevant items are recommended to the user in the Top-N set.

$$\text{Recall} = \frac{|\text{Relevent in Recommended}|}{|\text{Relevant}|}$$

iii. RMSE: RMSE measures how well a model fits the data. The lower the RMSE value, the closer the model fits the actual data. So this is a value we want to minimize.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y} - y)^2}{n}}$$

Where $\hat{y}$ is the predicted value and y is the actual value.

iv. MAE: Mean Absolute Error is the mean of the absolute values of individual predicted errors over the test set.

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y} - y|}{n}$$

v. F1 Score: F1 score is simply the weighted average of Precision and Recall. It considers both false positives and false negatives.

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

vi. NDCG: NDCG is perhaps the most important metric in evaluating various models. NDCG stands for the Normalized Discounted Cumulative Gain. DCG in particular penalizes when highly relevant items are placed low in the Top-N recommended list. For evaluation, we consider items in the predicted set which are common to the test set along with their ranking in the prediction set. We divide the DCG with iDCG which is the ideal ordering.

$$NDCG = \frac{DCG}{iDCG}$$

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{log_2(i+1)}$$

rel_i is the relevance of the item i, which is one if the item is present in both the test set and recommended set else it is zero. iDCG naturally assumes a relevance of 1 for all items. An ideal algorithm would yield an NDCG value of 1 since it would be equal to iDCG. We want NDCG score to be closer to 1. The results in

table 1 show that SVD and SVD with Grid Search have the highest NDCG scores.

7 RESULTS

The following table compares the various models built on different metrics.

As seen in table 1, the Deep Learning Model outperforms the standard baseline models on most metrics. The performance of neural network model with movie vectors as input is not as expected. As seen in the analysis, the content of the movie can be informative in providing recommendations. So, this can be one area where the model be tuned and extended.

8 CONCLUSIONS AND FUTURE WORK

As discussed in the report we have explored methods of including contextual information like genre and movie descriptions into our neural network. However, the modifications did not yield significant results. We aim to explore methods to improve our Neural Network-Based Model. Partly, deep learning-based approaches from the vastness of dimensions especially when the data itself is sparse. There are many parameters that are needed to be estimated and this would relay heavy computational costs.

9 ACKNOWLEDGEMENT

We are immensely grateful to our Professor, Yongfeng Zhang, and our TA, Zhiqing Hong for their constant support and guidance which was instrumental in the completion of this project.

REFERENCES

[1] David Adedayo Adeniyi, Zhaoqiang Wei, and Yongquan Yang. 2016. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics 12, 1 (2016), 90–108.

[2] He, Xiangnan and Liao, Lizi and Zhang, Hanwang and Nie, Liqiang and Hu, Xia and Chua, Tat-Seng "Neural Collaborative Filtering." arXiv, 2017. https://arxiv.org/abs/1708.05031.

[3] Chen, Wanyu and Cai, Fei and Chen, Honghui and de Rijke, Maarten "Joint Neural Collaborative Filtering for Recommender Systems." arXiv, 2019. https://arxiv.org/abs/1907.03459.

[4] "Movie2Vec: Feature Extraction From Wikipedia Plot Summaries". https://movie2vec.wordpress.com/2016/05/14/movie2vec-feature-extraction-from-wikipedia-plot-summaries/.

[5] https://www.aurigait.com/blog/recommendation-system-using-knn/.

[6] https://www.aurigait.com/blog/recommendation-system-using-matrix-factorization/.

[7] http://www.diva-portal.org/smash/get/diva2:1464572/FULLTEXT01.pdf.

[8] https://radimrehurek.com/gensim/models/word2vec.html