Moviefy: A Movie Recommendation System

Authors: Ashwin Perti (Mentor)

Ayush Plawat

Department Of Information Technology

ABES Engineering College, Ghaziabad, India

Abstract:

Everywhere around the world people watch movies on a daily basis. They want to watch and view content according to their personal liking. This led to the advent of recommendation systems which are now an integral part of our lives. These recommendation systems suggest us content based on a variety of factors that is best suited for us. Following this trend, we have tried to develop a movie recommendation system using KNN on the MovieLens dataset.

The dataset is first explored and then wrangled to achieve the best possible results here.

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. This algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The main principle behind this system is the one that we have learned in our childhood i.e. calculate the distance between two given points.

In the context of the model, we use ratings and user reviews to classify movies that are similar and can be recommended to the user.

Keywords: ***NumPy, Pandas, Matplotlib, Scikit learn, fuzzywuzzy, KNN***

1. Introduction:

Recommendation systems are a type of data processing systems that suggests content to the users. They are mostly used by consumer facing businesses such as YouTube, Spotify, Netflix etc. Since the users are consuming content at an unprecedented rate than ever or like we say they are binge watching things. This led to the rise of effective recommendation systems that help platforms and businesses to not only grab customers but also retain them.
These systems collect information from users to make suggestions and also further improve themselves. Personalized suggestions are the exact term that the companies are trying to achieve right now

It means trying to know the characteristics and preferences of the user by collecting and analyzing historical behavior to know what kind of person the user is, what kind of behavior preference the user has and then make suitable suggestions.

To recommend movies, first collects the ratings for users and then recommend the top list of items to the target user. Then the ratings of users are filtered so as to keep only those that are relevant. The main idea is not only to recommend movie but to recommend good movies. So, we need movies that are not poorly rated and ratings by users who are active. This will lead to the system recommending good movies to the users according to his/her preferences.

A lot of data exploration and wrangling need to be done, so as to not only understand the data but to also extract that is meaningful and useful to us.

In this paper, the key research contents are to help users to obtain user-interested movie automatically in the massive movie information data using KNN algorithm and collaborative filtering algorithm, and to develop a prototype of movie recommendation system based on KNN collaborative filtering algorithm.

2. Related Work (1000-1500 words):

A lot of work has been done on recommender systems especially those that recommend movies but the most notable work on this subject was done in the Netflix Prize.

The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users or the films being identified except by numbers assigned for the contest.
The grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.
It was a big improvement in the Netflix’s recommendation system and was quite an achievement.

Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies. Each training rating is a quadruplet of the form <user, movie, date of grade, grade>. The user and movie fields are integer IDs, while grades are from 1 to 5 (integral) stars.

The data provided by Netflix looked as follows:

- Training set (99,072,112 ratings not including the probe set, 100,480,507 including the probe set)
    - Probe set (1,408,395 ratings)
- Qualifying set (2,817,131 ratings) consisting of:
    - Test set (1,408,789 ratings), used to determine winners

- Quiz set (1,408,342 ratings), used to calculate leaderboard scores.

The qualifying data set contains over 2,817,131 triplets of the form <*user, movie, date of grade*>, with grades known only to the jury. A participating team's algorithm must predict grades on the entire qualifying set, but they are only informed of the score for half of the data, the quiz set of 1,408,342 ratings.

For each movie, title and year of release are provided in a separate dataset. No information at all is provided about users. In order to protect the privacy of customers, "some of the rating data for some customers in the training and qualifying sets have been deliberately perturbed in one or more of the following ways: deleting ratings; inserting alternative ratings and dates; and modifying rating dates".

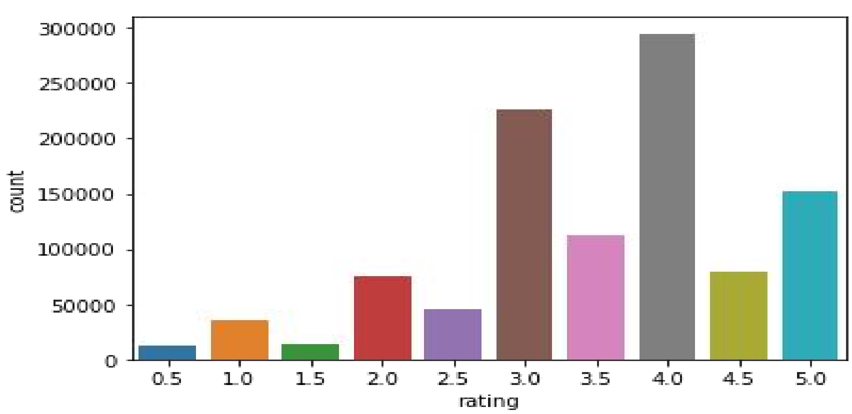
The training set is such that the average user rated over 200 movies, and the average movie was rated by over 5000 users. But there is wide variance in the data—some movies in the training set have as few as 3 ratings, while one user rated over 17,000 movies.

The most accurate algorithm in 2007 used an ensemble method of 107 different algorithmic approaches, blended into a single prediction.

3. Comparative Analysis of Existing Work/Proposed Methodology/Design (500-1000 words):

The proposed methodology is explained below:

We want to recommend movies to the user that are similar to this taste or match his preferences, but are also high rated. In short, we want to recommend good movies. Here is bar graph depicting the ratings in the dataset.



There are also Null values in the dataset shown below that we have to remove.

	movieId	title	genres	userId	rating	timestamp
1061817	131241	Ants in the Pants (2000)	Comedy|Romance	NaN	NaN	NaN
1061818	131243	Werner - Gekotzt wird später (2003)	Animation|Comedy	NaN	NaN	NaN
1061819	131248	Brother Bear 2 (2006)	Adventure|Animation|Children|Comedy|Fantasy	NaN	NaN	NaN
1061820	131250	No More School (2000)	Comedy	NaN	NaN	NaN
1061821	131252	Forklift Driver Klaus: The First Day on the Jo...	Comedy|Horror	NaN	NaN	NaN
1061822	131254	Kein Bund für's Leben (2007)	Comedy	NaN	NaN	NaN
1061823	131256	Feuer, Eis & Dosenbier (2002)	Comedy	NaN	NaN	NaN
1061824	131258	The Pirates (2014)	Adventure	NaN	NaN	NaN
1061825	131260	Rentun Ruusu (2001)	(no genres listed)	NaN	NaN	NaN
1061826	131262	Innocence (2014)	Adventure|Fantasy|Horror	NaN	NaN	NaN

The following are snippets of code that are used to do the same.

```
# Filtering Data

popularity_thres = 50
popular_movies = list(set(df_movies_cnt.query('count >= @popularity_thres').index))
df_ratings_drop_movies = df_ratings[df_ratings.movieId.isin(popular_movies)]
print('Shape of original ratings data: ', df_ratings.shape)
print('Shape of ratings data after dropping unpopular/low rated movies: ', df_ratings_drop_movies.shape)
```

```
Shape of original ratings data:  (1048575, 4)
Shape of ratings data after dropping unpopular/low rated movies:  (943006, 4)
```

```
# Filtering data
ratings_thres = 50
active_users = list(set(df_users_cnt.query('count >= @ratings_thres').index))
df_ratings_drop_users = df_ratings_drop_movies[df_ratings_drop_movies.userId.isin(active_users)]
print('Shape of original ratings data: ', df_ratings.shape)
print('shape of ratings data after dropping both unpopular movies and inactive users: ', df_ratings_drop_users.shape)
```

```
Shape of original ratings data:  (1048575, 4)
shape of ratings data after dropping both unpopular movies and inactive users:  (857707, 4)
```

```
# pivot and create movie-user matrix
movie_user_mat = df_ratings_drop_users.pivot(index='movieId', columns='userId', values='rating').fillna(0)

# create mapper from movie title to index using movie ID
movie_to_idx = {
    movie: i for i, movie in
    enumerate(list(df_movies.set_index('movieId').loc[movie_user_mat.index].title))
}
# transform matrix to scipy sparse matrix
movie_user_mat_sparse = csr_matrix(movie_user_mat.values)
```

This is not the entire code just the part that helps us grasp the approach to wrangling the data.

As seen in the last snippet the matrix is converted into sparse matrix. The solution to representing and working with sparse matrices is to use an alternate data structure to represent the sparse data.

The zero values can be ignored and only the data or non-zero values in the sparse matrix need to be stored or acted upon.

4. Analysis of Existing Results/Implementation Results:

The recommendation systems results are displayed below:

- The first recommendation is on a movie called ‘V for Vendatta’.

```
Input movie: V for Vendetta
Possible matches for: ['V for Vendetta (2006)']

Recommendation system starts
Recommendations for V for Vendetta:
1: Kill Bill: Vol. 2 (2004), with distance of 0.45811667820588053
2: Kill Bill: Vol. 1 (2003), with distance of 0.45113614263931523
3: Lord of the Rings: The Return of the King, The (2003), with distance of 0.44164255679675435
4: Casino Royale (2006), with distance of 0.44065041434654284
5: Prestige, The (2006), with distance of 0.439073634334578
6: 300 (2007), with distance of 0.43626464974891654
7: Iron Man (2008), with distance of 0.43219217884683925
8: Sin City (2005), with distance of 0.42203184608146216
9: Dark Knight, The (2008), with distance of 0.4187473948012579
10: Batman Begins (2005), with distance of 0.3800680734223937
```

- The second recommendation is for the Chinese movie called Shanghai Triad.

```
Input movie: Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
Possible matches for: ['Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)']

Recommendation system starts
Recommendations for Shanghai Triad (Yao a yao yao dao waipo qiao) (1995):
1: Amateur (1994), with distance of 0.812136512046904
2: Chungking Express (Chung Hing sam lam) (1994), with distance of 0.8052131703346157
3: Farinelli: il castrato (1994), with distance of 0.7994004038621397
4: Farewell My Concubine (Ba wang bie ji) (1993), with distance of 0.7874038762523473
5: Madness of King George, The (1994), with distance of 0.7848752995357182
6: Eat Drink Man Woman (Yin shi nan nu) (1994), with distance of 0.7830370983465151
7: Mrs. Parker and the Vicious Circle (1994), with distance of 0.7810640790405625
8: Richard III (1995), with distance of 0.7629395484286996
9: Queen Margot (Reine Margot, La) (1994), with distance of 0.7624432396973219
10: To Live (Huozhe) (1994), with distance of 0.7587464776779457
```

The results are displayed along with the distance to the point(movie) we provided. The system doesn’t recommend movies below a custom popularity threshold that was put in the model. This allow only good movies to be recommended to the users which match his/her preferences.

5. Conclusion and Future Scope:

The further enhancement of recommendation systems lies in using multiple engines on top of one another as used in the Netflix Prize. This doesn't that a recommendation system such as ours is obsolete, it is quite usable for personal recommendations.

The recommendation systems nowadays are all hybrid in nature and use various statistical method to not only make recommendations but also concur results that are most applicable or useful to the user.

The key here lies in finding parameters that a user consciously and unconsciously uses to find movies to watch and then designing a model as close to it as possible.

It is also quite possible that in the not so far future recommendation system become better than our own ability to classify movies which will lead to some great advancements in the consumer businesses of many corporations.

References:

[1] https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[2] C. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.

[3] Cui, Bei-Bei. (2017). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences. 12. 04008. 10.1051/itmconf/20171204008.

[4] https://machinelearningmastery.com/sparse-matrices-for-machine-learning/

[5] A Novel Scheme for Movie Recommendation System using User Similarity and Opinion Mining by Nagamanjula R, A. Pethalakshmi. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-4S2 March, 2019.

[6] https://en.wikipedia.org/wiki/Netflix_Prize

[7] A personalized movie recommendation system based on collaborative filtering, by V. Subramaniyaswamy, R. Logesh, M. Chandrashekhar, Anirudh Challa, V. Vijayakumar.

[8] Mukherjee, R., Sajja, N. & Sen, S. A Movie Recommendation System – An Application of Voting Theory in User Modeling. User Model User-Adap Inter 13, 5–33 (2003). https://doi.org/10.1023/A:1024022819690