Domain Background

Implementing recommendation systems to encourage users to purchase or engage with online products more has been an ongoing area of interest for online businesses in a range of domains: from media to ecommerce to job boards and travel. On one side from an ecommerce business perspective the benefit of providing great recommendations is the potential to increase retention and revenue, and on the consumer side the benefit is that a genuinely good recommendation can help users discover more products that are useful to them and add value. Recommendations have historically come in a range of forms: from recommendations between friends, to advertisements, to special offers and more. With the vast amount of data that ecommerce websites have available there is an opportunity to build not only a high quality but scalable way to make personalised recommendations to users.

In this project I will use the Netflix data set of movie ratings to build and implement a recommendation engine to a web app that takes an input of movie ratings from a user and outputs a recommendation based on their ratings. The goal is to recommend movies that the user will then go on to play and enjoy. One thing that I will also show in the user interface is information as to why that movie is being recommended. As privacy issues are becoming increasingly scrutinised, I believe that helping users understand why they are being made a particular recommendation will help build trust in their use of a particular service and how their personal data is being used. I will also research and describe an architecture to be used to implement a recommendation system on a larger and more real-time scale.

My personal motivation to this is to understand the strengths and limitations of the most commonly used recommendation algorithms; to better understand the results that I am being returned on e-commerce websites day to day. I will also learn how to implement an algorithm into a production environment which is a skill that is highly in demand in the data science domain.

Problem Statement

The problem to be solved is to provide a useful recommendation of movies to a given user based on movies that they have already rated. The main approach to achieve this will be to predict how many stars a user is likely to rate a movie that they have never seen before. This way movies that are predicted to get a high rating can be recommended to the end user as suggestion in the order of highest rated movie.

The Netflix dataset of user ratings was orginally shared as part of a competition hosted in 2006. The competition was won by the Bellcore Pragmatic Chaos team who improved Netflix’s existing recommendation system called Cinematch by +10%

from its existing RMSE score which was 0.9514. Cinematch which was accurate within half a star 75 percent of the time[1]. In the competition the star ratings were predicted by teams for a group or users and then evaluated vs actual star ratings that had been held back in a qualifying test set. The final winning solution presented by Bellcore was ultimately not implemented by Netflix after cost-benefit analysis of the amount of work needed to implement the algorithm vs the gains of a +10% increase in RMSE. This will be something to take into consideration as I explore the pros, cons and architectures of implementing different methods.

In 2014 the two algorithms being used in production for rating prediction by Netflix were Restricted Boltzman Machines in the context of collaborative filtering which uses neural network architectures and Matrix Factorization[2]. These are two architectures that I will explore as part of a solution.

Making recommendations by predicting movie ratings is only one way to make predictions. Others include taking into account other factors such a recently viewed movies, title popularity, novelty, diversity and freshness or a movie which have since been built into the objectives of recommendation engines at Netflix.[3]

Datasets and Inputs

This data set was made public by Netflix for the 2006 competition.

The original data set can be found on archive.org in the form of .tar files[4]. The training set file is training_set.tar which contains 17,700 files, one per movie and each contains the following[5]:

- MovieID in the first row of the file range from 1 to 17,770 sequentially.
- CustomerIDs range from 1 to 2649429, with gaps. There are 480,189 users.
- Ratings are on a five star (integral) scale from 1 to 5.
- Dates have the format YYYY-MM-DD.

There is also a file called movie_titles.txt which describes each movie. It contains:

- MovieID do not correspond to actual Netflix movie ids or IMDB movie ids.
- YearOfRelease can range from 1890 to 2005 and may correspond to the release of corresponding DVD, not necessarily its theatrical release.
- Title is the Netflix movie title and may not correspond to titles used on other sites. Titles are in English.

There is also a test set called probe.txt which contains:

- A Movie ID per row

[1] https://electronics.howstuffworks.com/netflix2.html
[2] https://www.quora.com/How-does-the-Netflix-movie-recommendation-algorithm-work
[3] https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429
[4] https://archive.org/download/nf_prize_dataset.tar
[5] https://www.kaggle.com/netflix-inc/netflix-prize-data/home

- A list of Customer IDs

Each of these Customer IDs in the probe.txt file are in the training set but can be used to calculate the RMSE of predictions and compared against the RMSE of of Netflix's previous recommendation algorithm Cinematch with an RMSE score of 0.9514[6]

As the data set has been uploaded to Kaggle.com [7] I will be using this data set which has the movie_titles in a .csv file, and all of the individual files within training_set.tar have been combined into 4 files called combined_data_1.txt up until 4.

I will also aim to use using the imdbpy package [8] to enhance the movie_title.txt file with additional attributes such as director, genre etc.

Overall the data set contains movie ratings from over 480,000 users and over 100M ratings.

Solution Statement

Some main types of Solutions to Recommendation Systems include[9]:

1. **Recommending the most popular items**

This requires no personalisation but looks at the most popular movies e.g by average rating, and could segment these into different movie types e.g by genre etc. As this includes no predictions of algorithms it won't be appropriate for the task at hand but I will order top ranked movies as part of my exploratory analysis of the dataset.

Classifier

For example k-means could be used for this to create a classification system for each user type and then predict ratings for each of those user types. This requires features to be known about the movie data set e.g genre, director, actors, etc that aren't available in this data set.

Content Based

This works on understanding how 'similar' one movie is to another.

Collaborative Filtering

User to User - We take a user and then find similar users to them, and look at what those users have also liked in the past. This would be useful to build a recommender that said 'users who watched similar films to you also liked y films'

[6]https://en.wikipedia.org/wiki/Netflix_Prize
[7] https://www.kaggle.com/laowingkin/netflix-movie-recommendation/data
[8] https://github.com/alberanid/imdbpy
[9] https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/

Item to Item - Here we look at items that are similar to a particular item and then recommend them to the user. This would be useful for example to show a user ‘you just liked x movie, how would you like to watch y movie’

Below is also a list of algorithms for recommendation systems suggested on the netflix blog[10]

- Linear regression
- Logistic regression
- Elastic nets
- Singular Value Decomposition
- Restricted Boltzmann Machines
- Markov Chains
- Latent Dirichlet Allocation
- Association Rules
- Gradient Boosted Decision Trees
- Random Forests
- Clustering techniques from the simple k-means to novel graphical approaches such as Affinity Propagation
- Matrix factorization

Benchmark Model

The first benchmark model will be a model that takes the average score across all users for each movie, and each movie will be predicted as having this average rating. Calculating the RMSE for this prediction will then get a benchmark RMSE score.

The second benchmark score will be against the RMSE score of the Netflix algorithm at the time of the competition which was 0.9514.

Evaluation Metrics

The evaluation metric will the the RMSE (root mean squared error) between the predicted review score vs the scores in the test set. The lowest RMSE achieved by the teams that entered the Netflix challenge was 0.8567[11]. I will evaluate the different models that I try using RMSE and discuss the pros and cons of each. The root mean squared error is calculated by summing the squared difference between the predicted and actual value for each observation, dividing it by the number of observations and taking the square root of this value.

[10]https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5

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

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

RMSE definition [12]

Project Design

Data Preparation

For the data preparation stage I will take the combined_data.txt files and transform them so that the dataset set is in a table format of:

- Movie ID
- Customer ID
- Date
- Rating

I will also need to join together the 4 files and at some point join the movie_titles.txt file to present the name of the recommended movie.

I will also use the imdbpy package to see if I can get additional features for the 17,770 movies in the data set including:

- Genre
- Director

Exploratory analysis

Here I will calculate:

- Distribution of review over movies
- Overall ranking of movies
- Sparsity of the data set

Non Personalised recommender systems

Here I will segment by the additional features I can get from IMDB and order these recommendations for each feature by the highest average rating to create a recommender based on highest rating.

I will use the probe.txt data set as the test set.

Benchmark Model

I will Calculate the raw average rating and RMSE of this benchmark model

Improved Recommender systems

[12] http://statweb.stanford.edu/~susan/courses/s60/split/node60.html

Collaborative Recommender Systems

User-User

I will look to build a user to user collaborative recommender using with the aim of finding similar users to predict movie recommendations. It will use the following techniques:

- Comparison with top N neighbours (around 30)
- Weighted Average
- User mean or z-score normalisation
- Cosine similarity over normalized ratings

Item-Item

This form of recommendation is analogous to "people who rate item X highly, like you, also tend to rate item Y highly, and you haven't rated item Y yet, so you should try it".[13] The steps I will take include:[14]

- Normalising the ratings per user
- Computing similarity between pairs of items i.e correlation between rating vectors
- Predict user-item rating by finding the weighted sum of “item neighbours” or linear regression to estimate the rating

Advanced recommender System

- **Restricted Boltzmann Machine**

The Restricted Botlzmann machine for collaborative filtering is a shallow neural network that can be used to predict movie ratings. I will investigate and apply this neural network to the movie ratings data.

- **Matrix Factorisation**

The last type of recommender system I will implement is a Matrix Factorization based Recommender System. This type of recommender system uses a Singular Value Decomposition (SVD) factorized matrix of the original similarity matrix to build recommender system. This method works by estimating a range of user dimensions to describe tastes in movies, and then also estimates these for each movie. This way a recommendation is made by matching the ‘tastes’ of a user vs their profile.

Evaluation

For the Restricted Boltzmann Machine and Matrix Factorisation, as they are used to predict star ratings I will calculate the RMSE of the models that I create and compare vs the benchmark model.

[13] https://en.wikipedia.org/wiki/Item-item_collaborative_filtering
[14]

Implementing into Website

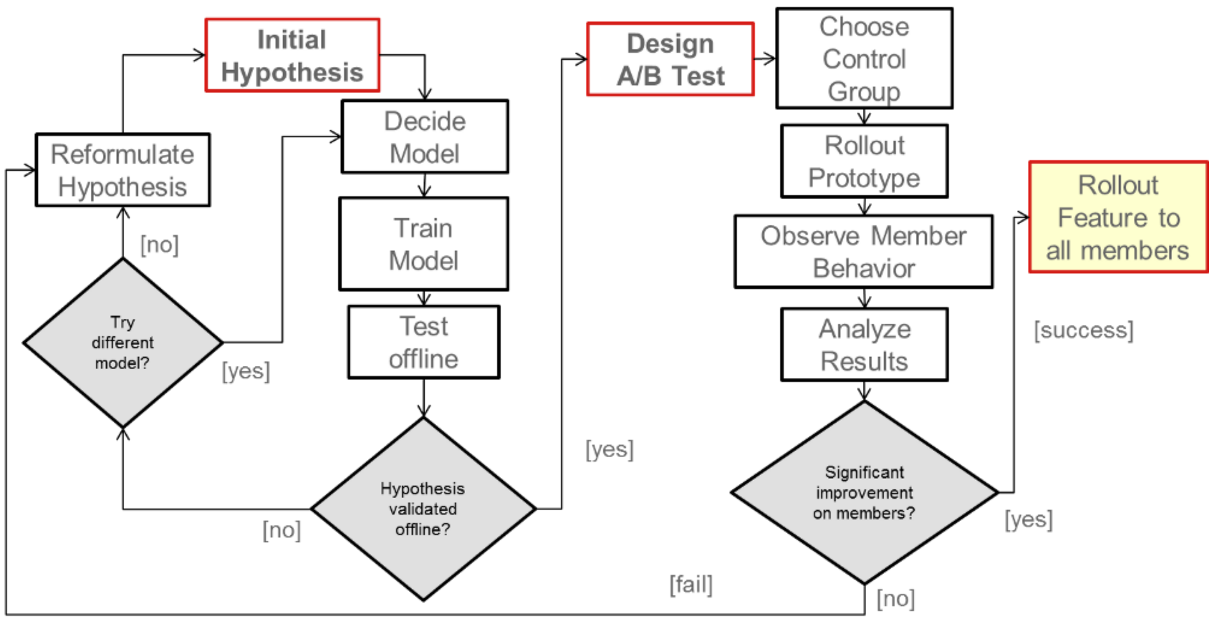
Here I will build a simple web app using django hosted on heroku to display the output of the 4 algorithms. The item-item recommender will ask for 1 movie input and return a list of recommendations. The other models will ask a user to specify a range of movies and return a list of recommendations based on what other users also liked.

Discussion of implementation of a recommender on a web/application.

Finally I am going to research architectures as to how implement a live recommendation system at scale for up to date recommendations.

Summary

The diagram below shows the approach that Netflix take to release a new recommendation system into production. My project will focus on the left-hand side of the workflow from initial hypothesis, decision of the model to use, training and testing of the model performance 'offline' i.e on a static test set. In practice Netflix then goes through the second step of designing running an A/B test to test the effect of the model on key business and user metrics before rolling out a new feature to their audience.



Netflix process for improved machine learning algorithms into production[15]

[15] https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5