Angelene Arito
DSC540 – 910
Final Project Write-up
6/12/19

Abstract

Objective: Recommender systems play an enormous role in content marketing and content selection in today's world and can be beneficial for both businesses and users. For a business, having a good recommender system in place as well as ways of predicting behavior can work together to create specialized marketing content that ultimately drives revenue. From a user standpoint, having a recommender system in place can help sort through the massive amount of content that is presented ta user every day on various platforms. The goal of this paper is to focus in on the film recommender space and take various movie features and utilize recommender systems to ultimately predict movie ratings, which can be big tells in terms of a film's overall success.
Methods: Content-based and collaborative-based filtering were used to engineer supplemental features to other movie metadata features in order to feed models and predict average movie rating. A few types of models were tested to ensure performance could be optimized. Among the regression machine learning methods chosen are random forests, gradient boosting, and neural networks.
Conclusion: Predicting movie rating based on content alone is insufficient, which stems from its objectivity. When paired with collaborative-based filtering features, however, the performance improves greatly and machine learning ensemble methods like random forests and gradient boosting can actually be used to create effective predictive models for rating.

1. Introduction

Customized content is ubiquitous and recommender systems are everywhere in today's world – product suggestions on Amazon, customized playlists generated on Spotify, movie recommendations on Netflix, etc. It's also seeming commonplace for companies to constantly iterate on their recommender system to enhance user experience or market certain content – even to the point of creating competitions around it (Netflix Prize in 2009). Over the course of this project, the goal was to incorporate recommender systems into helping to predict a target variable – in this case movie rating. From a business perspective, being able to predict the rating of a movie as well as having a recommender system tailored to specific needs can be powerful in driving revenue; from a user perspective recommender systems and things like rating predictions can help in sorting through the enormous amount of content we consume on a daily basis and choose more efficiently and effectively.

2. Literature Review

2.1 Concepts

The subject of recommender systems is one that has been widely studied, with a plethora of documentation and research around movie and tv recommender systems specifically. There are two main types of filtering methods that are typically used in building recommender systems: content-based and collaborative-based.

Content-based filters produce recommendations based on how similar an input item is to other items based on things like descriptions, tags, paragraphs of text, etc. In the context of movies, genres, casts lists, directors, and tags (such as "Disney" or "noir") are commonly used as the basis of determining whether one movie is similar to another movie. Typically, the content input is translated to a term frequency-inverse document frequency (tf-idf) matrix in which the documents are along one axis and the terms used in the documents are along the other axis. Values are populated by how often a given term pops up in a document, usually excluding common stop words to eliminate noise. From this matrix similar items are obtained by using some sort of distance function, like cosine similarity, to determine how “close” any two items are to one another. One of the common most cited pitfalls of content-based filtering is that it is objectively based, and does not consider subjective item elements, such as the quality of filming of a movie for example [1]. Content-based filters can also suffer from limited content analysis and may struggle when new users are introduced into the ecosystem [5].

Collaborative-based filters on the other hand come in two flavors: user-based and item-based. User-based collaborative filtering focuses on similar users and provides recommendations based on how similar users rated an item; item-based collaborative filtering utilizes how similar items were rated by a single user and makes a recommendation from that point. There are different approaches process-wise for this filtering method, but it is also common to construct matrices here as well and use similarity measures and create neighborhoods to determine how similar items or users are to one another. Commonly cited challenges for collaborative-based filtering methods include issues with sparsity (number of evaluated items < number of total available items by a large margin) and scalability (millions of users and items) [4]. It is also common to use hybrid filtering methods, which combine both content and collaborative filtering in order to bridge the gap in the shortcomings they face individually [5].

2.2 Related Work
In reviewing work related to recommender systems and predicting ratings, content-based, collaborative-based, or hybrid filtering methods are all often used as a baseline, though in varying capacities with slightly different approaches. Christakou and Stafylopatis [1] used the MovieLens 100K datatset and a hybrid filtering approach to build a recommender system. For the content-based piece, multiple neural networks were built for each user based on films they had already evaluated; once trained, the results were stored in a matrix. The same input was used for the collaborative filtering piece, in which Pearson correlation was used to determine which users were most similar for all users in the dataset. These results were then combined for experimentation with different methodologies in which precision and recall were used as metrics. Success rates were found to be lower based on content alone [1]. The work in Patel et al. [5] also mainly focused on hybrid filtering methods and tweaking how similarity is measured over the course of several experiment iterations. Neighborhood parameters were tweaked across experiments but it is interesting to note that hybrid filtering methods not only outperform purely content-based systems, but purely collaborative-based systems as well [5].

Piggybacking off of some of the limitations brought forward in the Christakou and Stafylopatis paper regarding content-based filtering, the work in the Ante Odić et al. [3] paper focuses on developing a context-aware service which attempts to factor in relevant contextual information in a movie recommender system. Three different contextual models were used for predicting ratings, which were based on matrix factorization (collaborative filtering). Context features such as location, time, weather, and mood were considered with results showing the importance of the detection procedure [3].

In Azaria et al. [2], there is more focus on the business perspective of recommender systems as a way of driving revenue as a contradiction to most other work which focuses on user utility. The methods are more survey-based than some of the other work and include asking users whether or not they would watch a movie based on a recommender system, assuming that a “yes” in this situation directly translates to paying for a given movie. Different algorithms are constructed to run experiments on – one for hidden agenda and one for revenue maximization. Depending on survey question asked to each user based on movie recommendations, different groups were established to experiment with each algorithm. Results and conclusions revolved more around whether users stayed loyal to movies they had seen before or were seeking variety [2]. This related work was included more for its application of recommender systems to achieve some other goal, rather than for the goal itself.

In Li and Yamada [4], there is a far greater focus on the pitfalls of both content-based and collaborative-based filtering and using decision trees to make inductive-learning-based recommendations that close the gaps in terms of sparsity, scalability, and transparency (readability). Similar to the user engagement in Azaria et al. [2], the system in this study is constructed via having users evaluate several movies which then serves as training data to feed to the decision tree [4]. Only 20 university students participated in the experiment so despite the positive results, it is hard to generalize. The use of a simple model like a decision tree may not be sufficient if the sample size in the experiment had been sizable. In this paper, random forests as a step up from decision trees and their performance for this problem.

3. Methodology

3.1 The Data

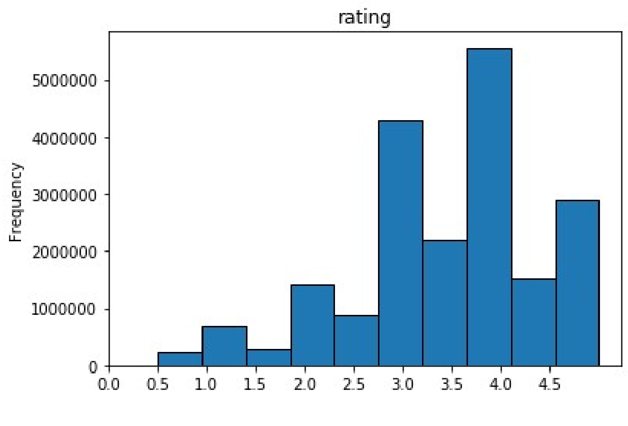
For this project, the initial dataset used was the MovieLens 20M dataset, which consists of 20 million movie ratings, various movie tags across some 27,000 movies (between 1995 and 2015) and over 100,000 users. Throughout the preprocessing and data evaluation process, it became necessary to also supplement this data with data from another movies set, so The Movies Dataset was also used. This dataset includes 26 million movie ratings (for movies release on of before 2017), metadata for approximately 45,000 movies (such as revenue, budget, and vote counts and averages from TMDB), and about 270,000 users. The primary focus was on the data in the MovieLens set, but as the project progressed it became clear that there weren't enough features in that set alone to build a sufficient model, which will be covered in detail in subsequent sections.

3.2 Exploratory Analysis & Preprocessing

The data in the MovieLens data itself didn't require a lot of transformation or gap filling; there was next to no missing data and values were fairly clean throughout. Movie titles did include the year in parentheses after the title, so that was separated to keep each feature covering a single concept. The data was split across several different files though, so merging data together was necessary based on ID, for which pandas was primarily used. Features in the dataset include:

feature	***description***
movieId	unique identifier of the movie
imdbId	corresponding id on IMDB
tmdbId	corresponding id on TMDB
title	title of the movie
genres	list of genre tags for the movie
tagId	unique identifier of an individual tag
tag	name of the tag
relevance	tag relevance to the movie it is attached to
rating	1-5 rating on movie given by a user
timestamp	date time - one for rating and one for tag

There were some immediate issues with processing power, which seemed to stem from pandas usage and just the sheer size of the data (even many more millions of rows once certain things were concatenated), but I was able to proceed with exploratory analysis, engineer initial features, and build recommender systems on this data.



Apart from movie metadata, ratings were a key component in many aspects of tackling this problem. The distribution of 1-5 star ratings in the dataset (which also consisted of half start ratings) was skewed left, with a denser concentration of ratings above 2.5. It was from these ratings that the target variable was established. For each movie in the dataset, the average rating was computed using all user ratings for that movie.

genre	*count*
Drama	5,677,224
Comedy	4,218,720
Thriller	2,266,152
Romance	1,976,256
Action	1,939,032
Crime	1,408,872
Adventure	1,324,272
Horror	1,152,816
Sci-Fi	1,020,840
Fantasy	783,960
Children	699,360
Mystery	693,720
Documentary	525,648
Animation	515,496
War	496,320
Musical	452,328
Western	247,032
IMAX	184,992
Film-Noir	126,336
No Genres Listed	1,128

Genre distribution was heavy in more traditional genres like "Drama", "Comedy", and "Action" with fewer movies tagged for genres like "Western" and "War". A small handful of movies had no genres listed and were therefore omitted from further analysis. In order to be able to use the genres as features in my models, I created dummy variables for each – where each genre became its own feature with "1" or "0" as a value.

The tags included in the data were used as the basis for the content-based filter that was built, so to make them useable they were aggregated into a single block of text for each movie to prepare for tf-idf matrix construction. Relevance scores and timestamps were not used in further analysis as the relationship dynamic changes between tags and movies and timestamp and ratings/tags.

3.3 Feature Engineering Round 1

After preprocessing the data, the next step I took was to build out a content-based recommender in order to use the results to engineer more features. As mentioned briefly in the preprocessing section, I aggregated the tags for each movie into a single field and then using the TfidfVectorizer from sklearn, I created a tf-idf matrix of all tags along one axis and all movies across the other, with term frequency populated as values and all normal English stop words omitted. I then used the cosine similarity to calculate the similarity between movies in the matrix, ultimately creating a function to take a single movie input and output a specified number of most similar movies.

Using the content-based recommender, I ran four different iterations for each movie in the dataset: one that pulled the top five movie recommendations, one that pulled ten, one that pulled fifteen, and one that pulled twenty. I then averaged out the top most similar movie scores for each to engineer avg_rating5, avg_rating10, avg_rating15, and avg_rating20 as features. The idea was to use average ratings of the most similar movies to predict the average rating of the target movie.

3.4 Modeling Round 1

I decided to focus my model building around regression, using random forests, boosting methods, and neural networks given some of the literature review outlined previously and to incorporate more than one ensemble method so that performance could be compared and the optimal model could be chosen. The initial round of features, which included only the four engineered content-based filtering features, produced weak results across the board (even with some model parameter tweaking). RMSE and explained variance were primarily used as metrics for performance evaluation and based on the four features related to similar movies, explained variance was very low, with very large confidence intervals in either direction.

	Random Forest	**Gradient Boosting**	**Ada Boosting**	**Neural Network**
RMSE	0.47 (+/- 0.15)	0.46 (+/- 0.15)	0.51 (+/- 0.09)	0.46 (+/- 0.15)
Expl Var	0.26 (+/- 0.27)	0.29 (+/- 0.27)	0.16 (+/- 0.17)	0.30 (+/- 0.26)
Run Time	8.14818716	0.428850889	0.436833382	3.081756353

In an effort to improve the models but still keep the features content-based initially, the Boolean movie genre features were pulled in and movie metadata from The Movies Dataset were pulled in as well, including revenue, budget, popularity, vote_average, and vote_count (TMDB scores). All features including the target were normalized in order to account for the widely varying scales between them. Resulting models had improved metrics, but still some large confidence intervals. With more features in this round, I did also run an iteration with wrapper selection on to see if there was a pattern that would emerge on which were important. For random forests, vote_average, vote_count, avg_rating5, avg_rating10, avg_rating15, and avg_rating20 were selected whereas in the boosting and neural network, only vote_average was selected.

Feature Selection Off

	Random Forest	Gradient Boosting	Ada Boosting	Neural Network
RMSE	0.28 (+/- 0.12)	0.28 (+/- 0.13)	0.36 (+/- 0.06)	0.32 (+/- 0.12)
Expl Var	0.70 (+/- 0.27)	0.69 (+/- 0.31)	0.60 (+/- 0.19)	0.61 (+/- 0.32)
Run Time	3.238393784	1.201810837	2.43545866	3.303164959

Feature Selection On

	Random Forest	Gradient Boosting	Ada Boosting	Neural Network
RMSE	0.54 (+/- 0.15)	0.62 (+/- 0.25)	0.69 (+/- 0.19)	0.60 (+/- 0.25)
Expl Var	0.70 (+/- 0.17)	0.59 (+/- 0.37)	0.55 (+/- 0.26)	0.63 (+/- 0.35)
Run Time	1.596730471	0.198505878	0.201425314	0.186499834

3.5 Feature Engineering Round 2

In a final effort to help improve the models, I went back to engineer more features from the data. It was apparent from the history and literature on the subject that engineering features based on collaborative filtering was the next logical step, so based on users, movies, and ratings, I used matrix factorization and singular value decomposition. In order to overcome processing power issues from the sheer volume of data, I focused only on the ten highest volume users based on how many ratings they had given across the board. For each of the movies I then generated the user predicted rating for each of the ten users to create ten new features for each movie – user1, user2, … user10.

3.6 Modeling Round 2

Adding the new collaborative-based features to the content-based features, I ran the models again, this time with far better results.

Feature Selection Off

	Random Forest	Gradient Boosting	Ada Boosting	Neural Network
RMSE	0.44 (+/- 0.06)	0.44 (+/- 0.08)	0.50 (+/- 0.08)	0.53 (+/- 0.04)
Expl Var	0.82 (+/- 0.04)	0.82 (+/- 0.05)	0.78 (+/- 0.04)	0.72 (+/- 0.10)
Run Time	4.03224206	1.619636774	3.384946108	3.553496361

Feature Selection On

	Random Forest	Gradient Boosting	Ada Boosting	Neural Network
RMSE	0.45 (+/- 0.06)	0.51 (+/- 0.07)	0.54 (+/- 0.06)	0.49 (+/- 0.08)
Expl Var	0.81 (+/- 0.03)	0.74 (+/- 0.04)	0.73 (+/- 0.03)	0.75 (+/- 0.04)
Run Time	1.756227493	0.173533916	0.225399017	0.544994831

Again, the target and featured were normalized to accommodate the varying scales and feature selection was used in one of the iterations. For random forest, the wrapper-based feature selection returned vote_average, avg_rating5, avg_rating10, avg_rating15, avg_rating20, user1, user4, and user5 and the boosting methods and neural network again just returned vote_average. The overall explained variance had a notable difference in confidence interval size, regardless of feature selection turned on or off.

Summary of Features

20M Movielens Dataset	**Movie Dataset**	**Engineered**
Adventure	revenue	avg_rating5
Animation	budget	avg_rating10
Children	popularity	avg_rating15
Comedy	vote_average	avg_rating20
Crime	vote_count	user1
Documentary		user2
Drama		user3
Fantasy		user4
Film-Noir		user5
Horror		user6
IMAX		user7
Musical		user8
Mystery		user9
Romance		user10
Sci-Fi		
Thriller		
War		
Western		

Content-based features

Collaborative-based features

round 1.1 features	***round 1.2 features***	***round 2 features***
avg_rating5	Drama	Drama
avg_rating10	Comedy	Comedy
avg_rating15	Thriller	Thriller
avg_rating20	Romance	Romance
	Action	Action
	Crime	Crime
	Adventure	Adventure
	Horror	Horror
	Sci-Fi	Sci-Fi
	Fantasy	Fantasy
	Children	Children
	Mystery	Mystery
	Documentary	Documentary
	Animation	Animation
	War	War
	Musical	Musical
	Western	Western
	IMAX	IMAX
	Film-Noir	Film-Noir
	avg_rating5	avg_rating5
	avg_rating10	avg_rating10
	avg_rating15	avg_rating15
	avg_rating20	avg_rating20
	revenue	revenue
	budget	budget
	popularity	popularity
	vote_average	vote_average
	vote_count	vote_count
		user1
		user2
		user3
		user4
		user5
		user6
		user7
		user8
		user9
		user10

Results

Based on the results from each round of modeling, it is apparent that predicting the rating of a movie is reliant not only on features related to the movie itself but also on user behavior (which is no surprise from a logical standpoint or from what was researched in the literature). There was generally similar performance RMSE and explained variance-wise across the different ensemble methods that were modeled. Within each of those models different hyperparameters were tweaked (such as number of estimators and criterion), with little change to the results. In the first round of modelling the explained variance on the models is very low, and in some of the models the confidence interval range could bring it down to zero. Adding in other movie features in the second iteration of the first round of modeling brought RMSE down across models, but the confidence interval range on the explained variance was still so large that arguably performance was not significantly different from the first round. It is noteworthy that none of the genres were selected in the feature selection round, but also suspicious that for the boosting and neural network iterations that only vote_average was selected as important. In the second round of modeling, explained variance truly seemed to improve and the confidence intervals on it were much tighter. It is interesting to note, though, that the RMSE values worsened from the second iteration of the first round of modelling. Feature selection was tested here as well, showing that the same single feature for the boosting and neural network was important as last round – vote_average. On the random forest the content generated features along with some of the top five users data were important. Overall, given that model performance was similar across the board, the gradient boosting without feature selection is likely most optimal, since its run time was more efficient.

Discussion

The goal of this project was to leverage content and collaborative filtering methods in predicting movie ratings, though it truthfully wasn't clear in what capacity at the start. As the project progressed, it became more apparent that the best way to leverage the filtering methods was to engineer features from them where the original data was lacking. The features that were engineered based on similar content alone were not sufficient. Feature selection was used in the model building process more as a benchmark because a good portion of features were engineered, and it is almost strange to note that vote_average was the only feature selected for the boosting and neural network methods. It is possible there is more to explore on that front and to perform some further processing to run through the model again. It may also be worth revisiting features to manually omit, such as vote_count or even all of the genre features.

The results overall just make sense from a logical standpoint when thinking about how human beings make selections. While it may be true that movies that are very similar in content will prove to be decent recommendations, the context-dependent factors are too strong to ignore. If I have two movies about aliens but one is Sci-Fi and the other is Horror, there is a decent chance I am speaking to completely different audiences, so it is important to understand how users relate to one another. This I think was reflected in the final model consisting of a combination of both types of features.

Future Work

While the final round of modeling got to some decent results in terms of performance, there are some obvious areas where work could be enhanced. For one, the processing challenges that came with the amount of data that was being used may have been better solved in a tool like Spark. By the end of the project to get to the final modeling stage there was some significant reduction of data – the final model was really only looking over the stretch of about 3000 movies. Aside from the "use more data" default response for future work, there is also very clear opportunity to leverage more weighted calculations. Since some features were engineered based on similar movies, I could have assigned a more weighted average rating to those that were more similar as opposed to less similar. In terms of the collaboratively generated features, the method with matrix factorization and singular value decomposition was fairly standard and could likely be expanded and made more complex or at least be used to generate additional features. It’s also possible to have approached this as a classification problem, given that 1-5 ratings naturally lend themselves to binning.

Conclusion

Content and collaborative filtering are the two main methods used as the foundation for recommender systems. From this project and previous work, we know that either one by itself is not as strong as when they are combined. We have also found that these filtering methods can serve as the foundation for predicting other things like movie rating. Even outside of the media realm, being able to anticipate a level of success of a product, restaurant, etc. can help to drive strategy for marketing content to users in a specialized way, which drives revenue. From a user perspective, having a way to filter through content and get to things we enjoy quickly is almost an integral part of everyday life. The work done in this project explored one portion of a much more complex web of recommender system strategies but seeing that content-similar movies and users who have reviewed lots of different movies can be important features in predicting rating, that may be able to be generalized to other topics – restaurants that are similar and users who have rated a lot of restaurants, books that are similar and users who have rated a lot of books, etc. Even if this specific piece can’t be used inside the realm of predicting movie ratings because it was too small in scope, the findings can be generalized to help further studies in determining what features to consider in their own predictions.

References

[1] Christina Christakou and Andreas Stafylopatis. A Hybrid Movie Recommender System Based on Neural Networks. 2005.
[2] Amos Azaria1 , Avinatan Hassidim1 , Sarit Kraus1,2 , Adi Eshkol3 , Ofer Weintraub3 Irit Netanely3. Movie Recommender System for Profit Maximization.
[3] Ante Odić et al. Predicting and Detecting the Relevant Contextual Information in a Movie-Recommender System. 2013.
[4] Peng Li, Seiji Yamada. A Movie Recommender System Based on Inductive Learning. 2004.
[5] Ronak Patel, Priyank Thakkar, K Kotecha. Enhancing Movie Recommender System. 2014.