Assignment 2: LibraryThing Recommendation System

Derek So ddso@ucsd.edu

Abstract

Recommender systems are an information filtering system that predicts a user's rating or preference for an item. In this paper, I construct a rudimentary recommendation system using a book review dataset from LibraryThing.

Introduction

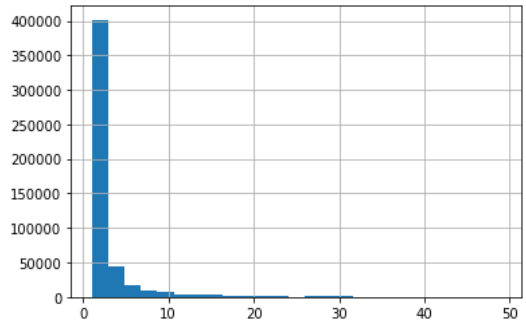
Online consumerism and socialization are at the forefront of the digital age, seen by the explosive growth and continued cultural relevance of companies like Amazon and Facebook. A significant amount of their success may be attributed to recommendation systems, utilizing and linking masses of data to further drive user purchases and/or networking. These systems are applicable to relatively smaller data, like the LibraryThing book review dataset selected for this project.

The data has 1,707,070 samples and 8 features:

work: the integer label of the book reviewed
flags: a list that may contain 'abuse' or 'not a review' warning labels
unixtime: the unixtime of the review
stars: the star rating given by the user to the work, from .5 to 5.0 in .5 intervals, with nan
nhelpful: the integer amount of 'likes' to indicate others found the sample review helpful
time: a string format of the date
comment: the comment string left by the user
user: the username of who left the review

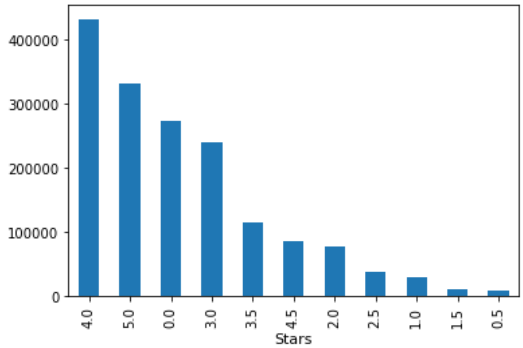
There are a total of 506,165 unique works in the dataset. Almost 90% of works have fewer than 10 reviews, with the top 3 user review counts at 2364, 1731, and 1278. Meanwhile, there are 83,195 unique users, with about 90% reviewing fewer than 40 works, and 31% having only reviewed 1 work, though the most prolific reviewers reviewed over 5,000 works.

Histogram of Users per Work



Unixtime and time are redundant and broadly fell between the years of 2005 and 2013, review counts growing steadily between each year. Some reviews apparently occurred in the year 1969, likely due to some error involving the unixtime start date. 50% of star ratings were 4.0 and above, though NaN star ratings were the third most common, seen below as '0.0'. Notably, there were no naturally occurring 0-star ratings in the data. 99.5% of reviews had fewer than 5 'helpfuls;' and the same proportion of comments were under 7,000 characters long, though the max length comment was almost ten times that at 64,234 characters.

Bar Chart of Star Rankings



Given the data of works and users who reviewed those works, a recommendation system of books for users based on thresholds of popularity and Jaccard similarity between the users of works is the ideal model.

Methods

A. Setup

The flags are used to clean the data of errors, reducing the sample size to 1,635,950. The last 10,000 samples are split for validation, while the rest are used to train the recommendation system. As the validation set is comprised of all positive recommendations, negative recommendations are created by the following. For each user in the set, a book they have not reviewed will be paired, along with the user's average 'helpful's and the selected book's average rating. The validity of the model will be judged on how accurate it predicts the validation set, seen as the target label 'read.'

Ex. of original positive and generated negative

work	user	stars	nhelpful	read
24671	Thalia	4.0	0.000000	1
385125	Jellyn	5.0	0.032609	0

A baseline predictive model using just a popularity threshold is created by iterating through the works by most reviewed, adding them to a set totaling their count until greater than half the total amount of reviews. The simplicity of the model and the arbitrary halfway breakpoint is a useful baseline for improvement on the model.

B. Improving the Baseline Model

Popularity is often a very strong indicator of 'participation,' or in this case, having read and reviewed the work. More exposure, sensibly, leads to more exposure in a positive feedback loop. As such, a model based on the most popular works being recommended is an effective baseline predictor. Accuracy can also be improved by looping through different proportions of the popularity threshold to find the optimal breakpoint.

However, the accuracy of the recommendations can be further tempered by other factors, such as similarity in users. Presuming that those who read the same work are interested in similar things, recommendations based on those similar interests is logically an effective model. Strictly mathematically, this can be achieved by calculating the Jaccard similarity between the works by who read them.

Jaccard Similarity Formula

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

intersection of two sets divided by their union

Unfortunately, as there are many users who reviewed a small number of works, the usefulness of this model is relatively lower than popularity. Most Jaccard values are small, and the nuance of finding the proper threshold to return positive may be less effective than popularity by itself.

The scalability of the Jaccard model is also an issue, as the calculations required are time-consuming. As the accessing, looping, and calculating works, users, and their similarities, takes approximately a tenth of a second per sample, the relatively small validation set of 20,000 takes 17 minutes to calculate, and a dataset the size of the original would take a day. Moreover, another issue is the assumption that the similarity is inherently positive, as that does not account for unfavorable reviews. While the dataset is positively skewed, less popular or well-rated books may have higher similarities and not be ideal for a recommendation.

Optimizing the model is a matter of iterating through each and finding the optimal threshold for both popularity and Jaccard similarity simultaneously. This is the best fit for the data and unlikely to cause any overfitting issues.

Regrettably, while I attempted to implement more conditions/thresholds utilizing the other two kept features, stars and nhelpful, such additions were detrimental. Part of the failure was difficulty generating those features for negative validation in a realistic manner, and finding thresholds for the recommendation was also more finicky for those features. Also, using stars as a threshold, if successful, likely was redundant with popularity and could have led to overfitting. Such features would have been more

useful in a model predicting their values, but that is not this model.

Another model I had considered was using some form of normalization with the comment lengths as another feature for classification, but that idea was scrapped for similar reasons as to why the other features failed.

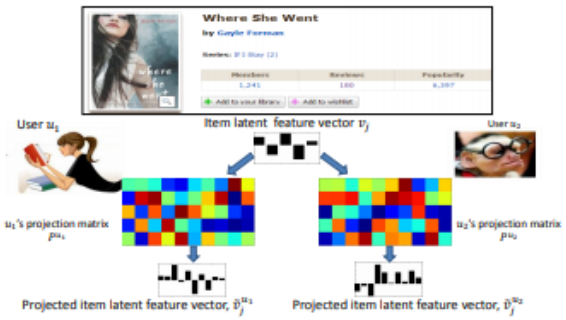
Ultimately, the final model is primarily based on popularity and uses Jaccard similarity to fine-tune some of the predictions.

Literature

The LibraryThing dataset was acquired from a paper, **Improving Latent Factor Models via Personalized Feature Projection for One Class Recommendation**, by Tong Zhao, Julian McAuley, and Irwin King. In this paper, the LibraryThing dataset was used as one of several to test whether recommender systems could be improved by new popular methods. Latent factor modeling, the basis for the experiment, improves upon recommender systems by introducing a vector weight, "such that the inner product of a user's and an item's latent factors represents the preference score or compatibility between the user and the item" (Zhao, McAuley, and King 1). In application to my own process, latent factor modeling may have been achievable using the work's average star rating as a bias.

However, the focus of this paper is to explore the effectiveness of a specific form of latent factor model, one-class recommendation. Given the lack of numerical ratings from the dataset, researchers into this method propose using "one-class" to model rankings for recommendation. The Jaccard similarity in my model uses this method, as it lacks any explicit ratings to influence the model. There are some drawbacks to this method, namely the one-dimensional nature.

The paper's personalized feature projection is designed to counteract that drawback by learning a user's latent features as a projection matrix instead of a vector. This allows works to create unique items in combinations with different users. Below is a figure from the paper, showing how they used this personalized model for the LibraryThing dataset.



Unfortunately, as the results of this paper are measured by area under curve, my model is not strictly comparable, as I use accuracy to judge the effectiveness of my model. The conclusions of the paper does convincingly prove that personalized feature projection would meaningfully improve my own model regardless of the method of model evaluation.

Another state-of-the-art method of recommender systems utilizes deep neural networks to improve models. **Wide & Deep Learning for Recommender Systems** explores combining both jointly trained wide linear models and deep neural networks in order to combine the positives of both types of models: memorization and generalization (Chang, et al.). This allows their model to account for both low-dimensional sparse features of uncommon items and for over-generalized more popular items. These researchers produced and evaluated over one million apps and one *billion* users on Google Play, resulting in significantly better app acquisition compared to models using only wide or deep methods.

Meanwhile, a series of articles from "Towards Data Science explores recommendation systems in both academia and the industry. The third article presents six research directions of deep recommendation systems as of 2019. Noting that there are three major issues to deep learning—interpretation difficulty (black box), big data, and extensive hyper-parameter tuning—writer James Le then presents areas in which recommender systems can improve. Some points are specifically tailored to video

recommendations, as is the focus of the series, but other points are more generalizable. In particular, he suggests joint learning frameworks, in which “each type of information source (review text, product image, numerical rating, etc) is adopted to learn the corresponding user and item representations based on available (deep) representation learning architectures” (Le), much like the personalized feature projections in the first paper. He also references recent developments such as models that allow extraction of complex features, like convolutional neural networks, to better predict ratings, or a new latent relational metric learning model that uses inferred relations to alleviate the inflexibility of metric models, improving both performance and modeling capability. The paper he references describes models improved by six to seven percent on movie datasets from netflix and MovieLens20M (Tay, Yi, et al.).

Results

Model	**Accuracy**	**Thresholds**
Popularity Baseline Half	.7234	.5 * total
Popularity Optimal	.7501	.66 * total
Jaccard Optimal	.73625	.022 Jaccard
Stars Baseline	.57865	>= 4 stars
Help Baseline	.42	>= average
Popularity & Jaccard	.7591	.8 * total .014 Jaccard
Popularity | Jaccard	.77415	.58 * total .103 Jaccard

As we can see from the above results, the best baseline is the popularity model with a .7234 accuracy. And as stated before, baseline models based on thresholds of the extra features of stars and helpfuls are far inferior, and the computational complexity of adding them to the final model meant that they were removed. A baseline Jaccard was not implemented due to the lack of a meaningful baseline, as predicting true given the similarity was above the average Jaccard was a measly .2763 accurate. Fitting to the best popularity baseline at containing the top two-thirds of the population resulted in a .7501 accuracy, which slightly outperformed the optimal Jaccard model.

Ultimately, the best performance was achieved by combining the two models. Fitting to a model where features had to meet both thresholds marginally improved upon the optimal population model. Fitting to a model where features only had to meet one of the thresholds, however, further raised the accuracy to .77415, with more intuitive thresholds for popularity and Jaccard similarity. Compared to the baseline model, the popularity breakpoint is reduced, better adheres to the meaning of popularity, and the threshold for Jaccard similarity is greatly raised from .022 to .103, giving more weight to similarity in the model. The either or nature of the final model allowed greater flexibility and better thresholds, along with the combination of both popularity and Jaccard similarity.

Conclusion

The simplicity of popularity threshold model is clear, though improvements can easily be made. My final model, compared to the initial baseline, improved accuracy by 7%. However, it is also evident that model improvements require a more sophisticated, if not delicate touch, as seen by the paper the LibraryThing dataset was originally from, along with the other examples of recommender systems.

References

Heng-Tze Cheng, et al. “Wide & Deep Learning for Recommender Systems.” *Wide & Deep Learning for Recommender Systems | Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 1 Sept. 2016, dl.acm.org/doi/10.1145/2988450.2988454.

Le, James. “Recommendation System Series Part 3: The 6 Research Directions of Deep Recommendation Systems That...” *Medium*, Towards Data Science, 28 June 2020, towardsdatascience.com/recommendation-system-series-part-3-the-6-research-directions-of-deep-recommendation-systems-that-3a328d264fb7.

Tay, Yi, et al. “Latent Relational Metric Learning via Memory-Based Attention for Collaborative Ranking.” *ArXiv.org*, 13 Feb. 2018, arxiv.org/abs/1707.05176.

Zhao, Tong, et al. *Improving Latent Factor Models via Personalized Feature Projection for One Class Recommendation*. cseweb.ucsd.edu/~jmcauley/pdfs/cikm15.pdf.