Amazon book recommendation system

Niyanta Shah
Computer Science Department,
University at Albany, SUNY
ID: 001285074
nsshah@albany.edu

Abstract— A**n extensive class of applications that involve predicting user responses to options is called a recommendation system. However, to bring the problem into focus, two good examples of recommendation systems are: 1. Offering news articles to on-line newspaper readers, based on a prediction of reader interests. 2. Offering customers of an on-line retailer suggestions about what they might like to buy, based on their history of purchases and/or product searches.**

Keywords— books, recommender systems, models

INTRODUCTION

Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user based on its previous history. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

The goal of a Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them.

In these cases, preference information in the form of either ratings for product items or product buying behavior is used to make recommendations to other like-minded users.

Several successful startup companies like Fire y, Net Perceptions, and LikeMinds have formed to provide recommending technology.

On-line book stores like Amazon and BarnesAndNoble have popular recommendation services, and many libraries have a long history of providing reader's advisory services.

Motivation

As, we know information overload is increasing now a day, it is required to have some recommendations based on our genre so that we need not have to search through the confusing information. Recommendation systems are thus, an aid for such users.

This topic was decided as we both love to read books and now-a-days with increase in use of e-books and kindle edition, we were inspired that for other users like us, we can recommend books to them based on genre or author.

Recommendation systems are very useful and we can see their increasing need and use.
For example, 65% of recommended movies on Netflix are being watched, amazon sales have increased by 35% because of recommending items.

Potential users

This algorithm covers not only book lovers but also general audiences. The change in data set, by changing one line of code will change the audience. Users like us, students, parents, business persons, teachers, almost every user is covered.

Potential applications

Recommendation systems can not only be used in books but also in movies, music, software, products in general, services, apps, etc.

Our focus is mainly on books. It will mainly be used to recommend to the users who buy from amazon. But it can be increased to kindle users or good reads users also, later in future.

Problem statement

In general, the world is an overcrowded place. It has lots of information. If the user reads books as per his mood, then he will have a lot of genre. Specifically, assume there are Nu users and Nb books, given a set of training examples (i.e. a set of triples (user, books, rating)), user-movie matrix $A \in R^{N_u * N_b}$. Our task is to predict all the users' ratings on all the books based on the training set.

Significance of Problem

As, we can see the problem is highly important, as the user will not know, what all other types of books can he read or in his genre are available.

Related work

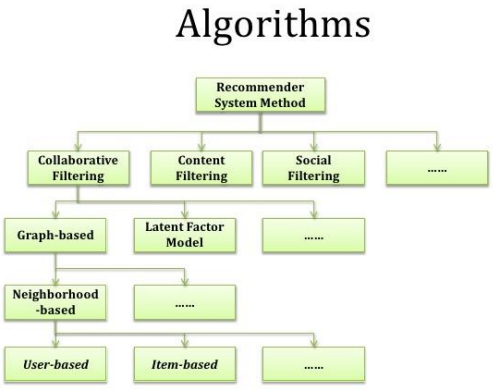
There have been designed several different recommendation systems for real world applications and they have been used at NetFlix, Hulu for recommending movies. At itunes store for recommending music

Some of the works are; Amazon.com recommendations: item-to-item collaborative filtering, Web usage mining: discovery and applications of usage patterns from Web data, Evaluating collaborative filtering recommender systems.

PROPOSED APPROACHES

Existing recommender systems almost exclusively utilize a form of computerized matchmaking called collaborative or social filtering. The system maintains a database of the preferences of individual users, finds other users whose known preferences correlate significantly with a given patron, and recommends to a person other items enjoyed by their matched patrons.

Different types of approaches used are:



What do recommendation systems do?

1. Predict how much you may like a certain product/service
2. Compose a list of N best items for you 3.
3. Compose a list of N best users for a certain product/service
4. Explain to you why these items are recommended to you
5. Adjust the prediction and recommendation based on your feedback and other people

Algorithms

- Collaborative filtering algorithms: this approach assumes that a given user's tastes are generally the same as another user of the system and that enough user ratings are available. Items that have not been rated by enough users cannot be effectively recommended. Unfortunately, statistics on library use indicate that very few patrons utilize most books.

  Therefore, collaborative approaches naturally tend to recommend popular titles, perpetuating homogeneity in reading choices.

  Also, since significant information about other users is required to make recommendations, this approach raises concerns about privacy and access to proprietary customer data.

- The first approach in collaborative filtering is the item-based K-nearest neighbor (KNN) algorithm. Its philosophy is as follows: to determine the rating of User u on books b, we can find other movies that are like books b, and based on User u's ratings on those similar movies we infer his rating on books b. To determine which books are "similar", we need to define a similarity function (similarity metric), as in [2], we use the adjusted cosine similarity between books a and b:

$$sim(a,b) = \frac{\sum_{u \in U(a) \bigcap U(b)} (R_{a,u} - \bar{R}_u)(R_{b,u} - \bar{R}_u)}{\sqrt{\sum_{u \in U(a) \bigcap U(b)} (R_{a,u} - \bar{R}_u)^2 \sum_{u \in U(a) \bigcap U(b)} (R_{b,u} - \bar{R}_u)}}$$

Book a, $\bar{R}$ u is User u's average rating, U(a) is the set of users that have rated book a and hence U(a) T U(b) is the set of users that have rated both book a and b. The advantage of the above-defined adjusted cosine similarity over standard similarity is that the differences in the rating scale between different users are taken into consideration. As its name indicates, KNN finds the nearest K neighbors of each movie under the above defined similarity function, and use the weighted means to predict the rating. For example, the KNN algorithm for movies leads to the following formula:

$$P_{m,u} = \frac{\sum_{j \in N_u^K(m)} sim(m,j) R_{j,u}}{\sum_{j \in N_u^K(m)} |sim(m,j)|},$$

- To formulate the EM algorithm formula in this case, let latent random variable Gm ∼ Qm(·) denotes the group of book b. Define U(m) as the set of users that have rated book b. Thus, the E-step formula is described as follows:

$$\begin{aligned} Q_m^{(t+1)}(g) &= P(G_m^{(t+1)} = g | R_{u,m}; \mu_{g,u}, \sigma_{g,u}^2) \\ &= \frac{Q_m^{(t)}(g) \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g,u}} exp\left(-\frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2}\right)}{\sum_{g'} Q_m^{(t)}(g') \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g',u}} exp\left(-\frac{(R_{u,m} - \mu_{g'})}{2\sigma_{g',u}^2}\right)} \end{aligned}$$

$$\max_{\mu_{g,u}, \sigma_{g,u}^2} \sum_m \sum_g Q_m(g) \left[log\left\{ \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g,u}} exp\left(-\frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2}\right)\right\} - log(\right.$$

$$\max_{\mu_{g,u}, \sigma_{g,u}^2} \sum_{m,g} Q_m(g) \sum_{u \in U(m)} \left[-log(\sigma_{g,u}) - \frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2} \right].$$

$$\mu_{g,u} = \frac{\sum_{m \in M(u)} Q_m(g) R_{u,m}}{\sum_{m \in M(u)} Q_m(g)}$$

$$\sigma_{g,u}^2 = \frac{\sum_{m \in M(u)} Q_m(g)(R_{u,m} - \mu_{g,u})^2}{\sum_{m \in M(u)} Q_m(g)}$$

$$P_{u,m} = \sum_g Q_m(g) \mu_{g,u}.$$

- The second approach is: An alternative approach to solve this problem is Item-based EM algorithm. In this algorithm, we classify books into G groups, and each book b belongs to Group g with probability Qm(g). For a given Group g of books, we assume the ratings of different users are independent and Gaussian, that is, P(Ru,m|m ∈ g) ∼ N(µg,u, σ2 g,u).

$$Q_m^{(t+1)}(g) = P(G_m^{(t+1)} = g | R_{u,m}; \mu_{g,u}, \sigma_{g,u}^2) = \frac{Q_m^{(t)}(g) \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g,u}} exp\left(-\frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2}\right)}{\sum_{g'} Q_m^{(t)}(g') \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g',u}} exp\left(-\frac{(R_{u,m} - \mu_{g',u})^2}{2\sigma_{g',u}^2}\right)},$$

$$\max_{\mu_{g,u}, \sigma_{g,u}^2} \sum_m \sum_g Q_m(g) \left[log\left\{ \Pi_{u \in U(m)} \frac{1}{\sqrt{2\pi}\sigma_{g,u}} exp\left(-\frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2}\right)\right\} - log(Q_m(g)) \right],$$

$$\max_{\mu_{g,u}, \sigma_{g,u}^2} \sum_{m,g} Q_m(g) \sum_{u \in U(m)} \left[-log(\sigma_{g,u}) - \frac{(R_{u,m} - \mu_{g,u})^2}{2\sigma_{g,u}^2} \right].$$

$$\mu_{g,u} = \frac{\sum_{m \in M(u)} Q_m(g) R_{u,m}}{\sum_{m \in M(u)} Q_m(g)}$$

$$\sigma_{g,u}^2 = \frac{\sum_{m \in M(u)} Q_m(g)(R_{u,m} - \mu_{g,u})^2}{\sum_{m \in M(u)} Q_m(g)}$$

$$P_{u,m} = \sum_g Q_m(g) \mu_{g,u}.$$

- Second approach is content based recommendation system. In Content based Recommendation Engine, system generates recommendations from source based on the features associated with products and the user's information. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.

  In Context based Recommendation Engine, system requires the additional data about the context of item

consumption like time, mood, and behavioral aspects. These data may be used to improve the recommendation compared to what could be performed without this additional source of information.

- Third approach is sparse SVD algorithm which is based upon sparse matrix SVD. This approach models both users and books by giving them coordinates in a low dimensional feature space i.e. each user and each book has a feature vector. And each rating (known or unknown) is modeled as the inner product of the corresponding user and movie feature vectors. In other words, we assume there exist a small number of (unknown) factors that determine (or dominate) ratings, and try to determine the values (instead of their meanings) of these factors based on training data. Mathematically, based on the training data (sparse data of a huge matrix), we try to find a low-rank approximation of the user-movie matrix A. This approach is called sparse SVD algorithm

$$\min_{\mathbf{u}_i,\mathbf{m}_j} \sum_{(i,j)\in I} (R_{i,j} - \mathbf{u_i}^T\mathbf{m_j})^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u_i}\|^2 + \sum_j n_{m_j} \|\mathbf{m_j}\|^2 \right),$$

- One another model is the utility model. Without building a utility matrix it is impossible to give recommendation systems. Users have preferences for certain items, and these preferences must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item.

SYSTEM DESIGN AND IMPLEMENTATIONS

Data set:

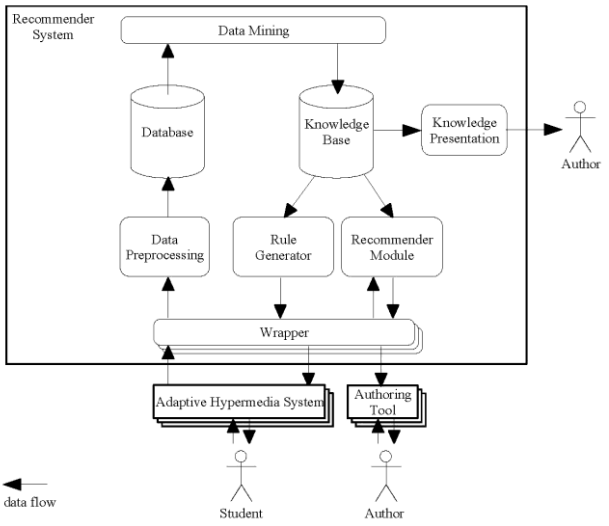
Our data set mainly was initially in a csv format. It has user ID, Book ID, and rating.

The size of our data set is around of 1 gigabytes. Then as we converted it to csv it got compressed.

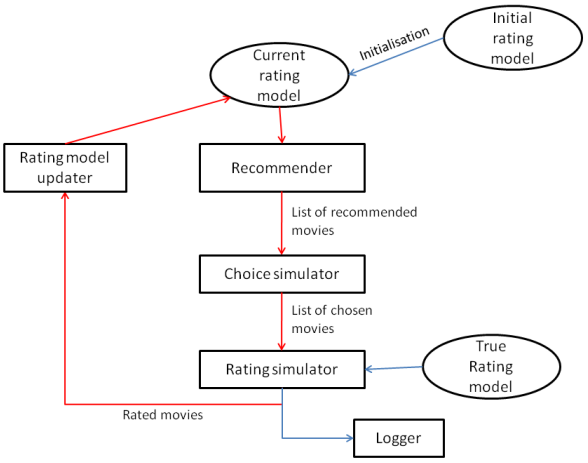
The example is as below:

	A	B	C
1	User ID	Book Id	ratings
2			
3	AH2L9G3DQHHA	116	4
4			
5	A2IIIDRK3PRRZ	116	1
6			
7	A1TADCM7YWP	868	4
8			
9	AWGH7V0BDOJ	13714	4
10			
11	A3UTQPQPM4T	13714	5
12			
13	A8ZS0I5L5V31B	13714	5
14			
15	ACNGUPJ3A3TI	13714	4
16			
17	A3BED5QFJWK	13714	4
18			
19	A2SUAM1J3GNI	13714	5
20			
21	APOZ15IEYQRF	13714	5
22			

Here are some of the architectures of already proposed systems:



One of the general approaches for recommendation system is:



Graphic user interface

We have not implemented any GUI. We have tried to implement the recommendation system and get the output in form of a file.

Major components:

The main components in this system are user and book ID. The user ID is linked to book ID.

The algorithms and implementation uses matrix factorization to give the recommended books.

MODEL

A basic matrix factorization model Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. Accordingly, each item i is associated with a vector $q_i \in P^f$, and each user u is associated with a vector $p_u \in P^f$. For a given item i, the elements of qi measure the extent to which the item possesses those factors, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, qi T p_u, captures the interaction between user u and item i—the user's overall interest in the item's characteristics.

This approximates user u's rating of item i, which is denoted by r_{ui}, leading to the estimate $\hat{r}_{ui} = q_i^T p_u$.

Such a model is closely related to singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values caused by sparseness in the user-item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting.

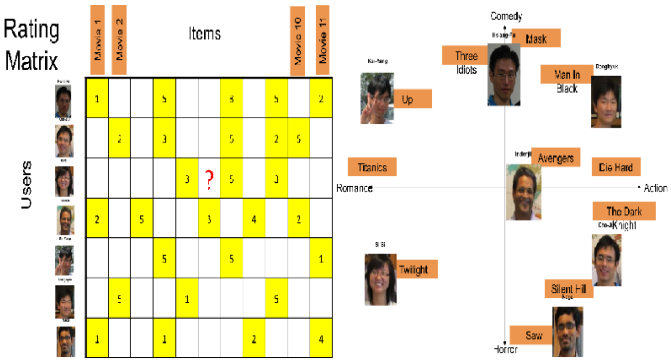
To learn the factor vectors (p_u and q_i), the system minimizes the regularized squared error on the set of known ratings:

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

Here, κ is the set of the (u,i) pairs for which r_{ui} is (the training set).

Matrix factorization can be done using SVD:

This is how it will be done, if we used SVD. a



Here, in our algorithm we are creating a matrix using user ID, book ID, and ratings. The matrix created is:

```
In [28]: matrix
Out[28]:
```

BookID	0000047116X	000077135X	000100039X	000102521X	000136118X	000161102X	000171130X	000171287X	000184749X	000195783X	...	6551297	6551
userid													
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
20	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

Output:

```
In [121]: already_rated.head()
Out[121]:
```

	UserID_x	BookID	Ratings_x	userid_x	UserID_y	Ratings_y	userid_y
0	A1W7WTTAB0U9W9	1712772	5	837	ADAKCLAWYLQYK	5	831
1	A1W7WTTAB0U9W9	1712772	5	837	A2OURT6VB1TMG0	5	832
2	A1W7WTTAB0U9W9	1712772	5	837	A2FS9CYCWC6HLW	5	833
3	A1W7WTTAB0U9W9	1712772	5	837	A24EC8GFQBMCDB	2	834
4	A1W7WTTAB0U9W9	1712772	5	837	AO2WMECV4UPQW	5	835

```
In [120]: predictions
Out[120]:
```

	BookID
14079	2247399
12427	2247399

```
In [109]: predictions['BookID'].unique()
Out[109]: array(['2247399'], dtype=object)

In [ ]:
```

CONCLUSION

There are limitations to this recommendation system. It is not a fully functional and can be used online recommendation system. This system is not efficient as for larger set of data, it will take larger time, and becomes slow. The output shows which user has read how much number of books and how many has he given ratings too. For a user, the system generates the recommended movie.

REFERENCES:

1) Https://beckernick.github.io/matrix-factorization-recommender/

2) Evaluating collaborative filtering recommender systems by johnathan l herlocker, joseph a konstan, john t ridel, loren g terveen.

3) An energy-efficient mobile recommender system yong ge hui xiong1 , alexander tuzhilin , keli xiao, marco gruteser, michael j.pazzani

4) Amazon.com recommendations: item-to-item collaborative filtering

5) http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

6) A proposed book recommender system abhilasha sase1 , kritika varun2 , sanyukta rathod3 , prof. Deepali patil4

7) Aggarwal - data mining the textbook - 2015.pdf

8) Lecture-7-a-recommendation system.pdf

9) Evaluating collaborative filtering recommender systems

10) Item-based collaborative filtering recommendation algorithms

11) http://www.cs.utexas.edu/~ml/papers/libra-sigir-wkshp-99.pdf

12) http://www.ugr.es/~essir2013/slides/ESSIR_2013_Recsys_final.pdf