**DOCUMENTATION**

**Generic Recommender Systems**

The increasing importance of the Web as a medium for electronic and business transactions has served as a driving force for the development of recommender systems technology. Recommender System helps business to increase their sales through cross-selling and advertisements. It discovers information about the items that are likely to be of interest to the users. More than 35% of product sales come from recommendations in Amazon.com, 66% of movies rented in Netflix are recommended ones, 38% more click-throughs are generated from recommendations in Google News.

All these statistics tell about the importance of recommender system in the present day both from the perspective of end-user and industries running an online business.

In this project I have tried to implement the concept of generic recommender systems as the name suggests I have worked on generalisation of recommender system, to understand this let's have a look on some famous real-world recommender systems-

|  |  |
| --- | --- |
| System | Product Goals |
| Amazon.com | Books and other products |
| Netflix | DVDs, Streaming videos |
| Jester | Jokes |
| GroupLens | News |
| MovieLens | Movies |
| Last.fm | Music |
| Google News | News |
| Google Search | Advertisements |
| Facebook | Friends, Advertisements |
| Pandora | Music |
| Youtube | Online videos |
| Tripadvisor | Travel products |
| IMDb | Movies |

Idea of generalisation of recommender systems is that to build a single program that can be used to recommend products/services in most of the real world scenarios.

Now let’s discuss detail working of generic recommender system. First task is to extract data for our program. For the working of generic recommender systems I have define four categories of columns (in a single table) That will require (not necessary all) to perform functionality of my program.

|  |  |
| --- | --- |
| Customer Details | |
| cutomerID | customerName |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Customer Interactions | | | | | | | |
| views | like | dislike | bookmark | rating | follow | comment | purchased |

|  |  |  |  |
| --- | --- | --- | --- |
| ServiceDetails | | | |
| ServiceID | ServiceName | ServiceCategory | ServiceDescription |

|  |  |  |
| --- | --- | --- |
| Service Provider Details | | |
| ServiceProviderID | ServiceProviderName | ServiceProviderRating |

Now the user who will enter the data has to define a mapping from his set of columns to the above mentioned set of columns.

|  |  |
| --- | --- |
| cutomerID | Column-A |
| customerName |  |
| views | Column-B |
| like | Column-C |
| dislike |  |
| bookmark | Column-D |
| rating | Column-E |
| follow |  |
| comment |  |
| purchased | Column-F |
| ServiceID | Column-G |
| ServiceName | Column-H |
| ServiceCategory | Column-I |
| ServiceDescription | Column-J |
| ServiceProviderID | Column-K |
| ServiceProviderName |  |
| ServiceProviderRating |  |

**Note:-** It can be seen from above that It is not necessary that user provide mapping to each and every columns.

Now we will extract the required columns from the data provided and will rename the columns with the help of the mapping provided by the user.

I have worked on 4 different types of recommendations model that are :-

**1**.**Popularity based recommendations**:- It recommends top N popular products/services from the data to every customer.

**2**.**Personalised recommendations**:- It recommends top N products/services from the data to the customer based on its previous interaction history.

**3**.**Recommendations based on last purchase**:- It recommends top N products/services that are likely to be purchased based upon the last purchase and the interest of the user.

**4**.**Recommendations of customers for a particular product**:- It recommends top N customers that are very likely to interact/buy that product.

Let’s discuss the working of each type of recommendation models.

**Popularity based recommendations**

To build this type of recommendation model we will use Service Id, Service Name, customer interaction columns from the data provided. Where customer interaction columns can have different combination of columns depending upon the mapping provided.

For example consider customer interaction columns have following columns:-

Views , like , bookmark , rating, purchased

Firstly normalise the rating column and form additional column which will mention rated purchases for each product.

now we will define interection strength for each product/service as:

total interactions= views + likes + bookmarks + purchased + rated purchases

interation strength = no. of views on that product\*log(1+total interactions/ total views)

+no. of likes on that product\*log(1+total interactions/ total likes) +no. of bookmark on that product\*log(1+total interactions/ total bookmarks)

+no.of rated on that product\*rating(after normalising)\*log(1+total interactions/ total rated purchases)

Now based on this interaction strength we will recommend top N products/services with high interaction strength

**Personalised recommendations**

Machine learning algorithms in recommender systems that are typically classified into two categories — content based and collaborative filtering methods. Content based methods are based on similarity of item attributes and collaborative methods calculate similarity from interactions. In this project I have used Hybrid approach and combined both methods for better result.

**Content based approach**

In this approach we are using a very popular technique in information retrieval (search engines) named TF-IDF. This technique converts unstructured text into a vector structure, where each word is represented by a position in the vector, and the value measures how relevant a given word is for an document. As all items will be represented in the same Vector Space Model.

Using this method we can build item profiles from the service details columns.

Now after this we will build user profiles by taking weighted average of the items profiles which customer has interacted previously based upon its interaction strength with each item.

**Note**:- Interaction strength mentioned here is calculated in a same way discussed in personality based recommendations but the difference is that their we have calculated interaction strength of all users on each item but here we will calculate interaction strength of each user on each item.

Now If we want to recommend top N products to a customer we have to calculate cosine similarity between his user profile and the item profile of each and every item and recommend top N similar products.

**Collaborative filtering approach**

There are many model-based CF algorithms, like neural networks, bayesian networks, clustering models, and latent factor models such as Singular Value Decomposition (SVD) and, probabilistic latent semantic analysis.

Here we a use popular latent factor model named Singular Value Decomposition (SVD).

To implement this method first we have to form user-item matrix in which the values represent the interaction strength of user on a particular item and fill the left out values with zero.

After the factorization, we try to reconstruct the original matrix by multiplying its factors. The resulting matrix is not sparse any more. It was generated predictions for items the user have not yet interaction, which we will use to recommend top N products/services based on predicted interaction strength.

**Hybrid approach**

hybrid methods have performed better than individual approaches in many studies Let's build a simple hybridization method, by only multiply the CF score with the Content-Based score, and recommend top N products/services by resulting score.

**Recommendations based on last purchase**

Let’s assume we have customer x who has recently purchased product y now we need to recommend such products that are more likely to be purchased with product y and also matches the interest of customer x.

To generate this type of recommendation firstly we need to findout top N commonly purchased products by the customers who have also purchased product y and now we will rank these N products based upon the interest of customer x by calculating cosine similarities between item profile of these N items and the user profile of customer x.

**Recommendations of customers for a particular product**

To generate this type of recommendation we can again use item profiles and user profiles. This time we will calculate cosine similarities between item profile of the given product and the user profiles of all the customers and recommend top N similar user profiles based on the score.

**Improvements**

Here I am mentioning few points that can be used in the model to improve its performance.

* Considering Time stamp as a feature because customer generally prefer new products/services.
* Catching customer sentiments towards the product from the review made by customer.