IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

COLLABORATIVE FILTERING OF PRODUCT RATINGS USING COSINE SIMILARITY

[1]M.Ravi, [2] K.Lasya, [3] M.Jashwanth [4] S.Rakesh

[1]Assistant Professor, [2]Student, [3]Student, [4]Student
[1]Department of Information Technology,
[1]JBIET, Hyderabad, India

Abstract: Aiming at the data sparse and cold start problems in collaborative filtering recommendation algorithm, an optimized solution based on user characteristics and user ratings is proposed in this paper. Based on users' basic attributes and users' history score record, the similarity of users and the similarity of items are calculated, and the nearest neighbour users and similar items are obtained. The advantage of the algorithm is that it combines the user's score and personal attributes to calculate the similarity between users and to recommend items. The optimized algorithm is applied to the recommendation of insurance products. Experiments based on real data from insurance company show that this method can reduce the average absolute error and improve the accuracy of recommendation. Collaborative filtering is a technique used by recommender systems. In the newer, narrower sense, collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users.

Index Terms **-user attributes; collaborative filtering recommendation; Insurance products; algorithm optimization**

INTRODUCTION

There has been a lot of work done both in the industry and academia on developing new recommender algorithms to recommender systems over the last decade. Since the seminal paper on collaborative filtering appeared in the mid-1990s, recommender systems have become an important research area. According to Breese, algorithms for collaborative recommendations can be grouped into two general classes: memory-based and model-based. The main difference between collaborative modelbased techniques and memory-based approaches is that the model-based techniques calculate utility (rating) predictions based not on some ad hoc heuristic rules, but, rather, memory-based technique is based on a model learned from the underlying data using statistical and machine learning techniques.

With the popularization of cloud computing, .Internet of things, especially the development of mobile applications, the amount of information data is explosive growth, and the type of data and their relations have become complex and diverse. The information contained in big data has brought about the problem of "information overload" for people to understand and make decisions. As an effective way to solve this problem, the recommendation system is playing a significant role.

As the most successful technology of the recommendation system, collaborative filtering, by the user's history score, the nearest neighbour is found and predict the interest of items to do recommendation. Although collaborative filtering algorithm has been well developed in the application of real application, it is also facing new challenges in new applications, such as electronic commerce, For example: the score matrix of the data is sparse, the cold start of new item and new user, the real-time problem, and the data processing and storage scalability issues. In order to improve its performance, a lot of researches have been done.

Based on previous studies, combined with the characteristics of e-commerce in the insurance industry, this paper proposes an optimized collaborative filtering algorithm based on user attributes. The new algorithm comprehensive considers the similarity of user's rating score and the similarity of user's attributes, and the final similarity is more accurate. The nearest neighbour users are closer and their interests are more similar. We use real encrypted data from an insurance company to carry out the experimental verification, and the experimental results prove the validity of the algorithm.

Collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users. The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue that of a randomly chosen person. Note that these predictions are specific to the user, but user information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes. In the more general sense, collaborative

filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources.

The project comprises of four modules. The first module deals with adding the customer details and product ratings. The second module deals with creating the comma separated value data from the existing data in the DB. The third module deals identifying the positions in the data file, where the rating estimates are to be made. The final module deals with estimating the ratings at the identified positions.

The numpy tool is used in this project to calculate the dot products of the ratings of different customers. These dot products are further used to identify the similar customers. Numpy tool is also used to normalize the product ratings vectors. The 'flatten' and 'partition' functions of numpy are used to identify the next customers, with highest similarity coefficient. The 'savetext' function of the numpy module is used to generate a .csv file, with the new as well as existing ratings. The 'genfromtext' tool is used to load the .csv data file into numpy arrays for further processing**.**
Tools Used: numpy, genfromtext

LITERATURE SURVEY

One of the potent personalization technologies powering the adaptive web is collaborative filtering. Collaborative filtering is the process of filtering or evaluating items through the opinions of other people. Collaborative filtering brings together the opinions of large interconnected communities on web, supporting filtering of substantial quantities of data.

Computers and the web allow us to advance beyond simple word-of-mouth. Instead of limiting ourselves to tens or hundreds of individuals the internet allows us to consider the opinions of thousands. The speed of computers allow us to process these opinions in real time and determine not only what a much larger community thinks of an item, but also develop a truly personalized view of that item using the opinions most appropriate for a given user of groups of users.

To be more formal, a rating consists of the association of two things – user and item –often by means of some value. One way to visualize ratings is as a matrix. Without loss of generality, a ratings matrix consists of a table where each row represents a user, each column represents a specific movie, and the number at the intersection of a row and a column represents the user's rating value.

The absence of rating score at this intersection indicates that user has not yet rated the item. The term user refers to any individual who provides ratings to a system. Most often, we use this term to refer to the people using a system to receive information (e.g. recommendations) although it also refers to those who provided the data (ratings) used in generating this information.

Collaborative filtering systems produce predictions or recommendations for a given user and one or more items. Items can consist of anything for which a human can provide a rating, such as art, books, CDs, journal articles, or vacation destinations. Ratings in a collaborative filtering system can take on a variety of forms.

Scalar ratings can consist of either numerical ratings, such as the 1-5 stars provided in Movie Lens or ordinal ratings such as strongly agree, agree, neutral, disagree, strongly disagree.

Binary ratings model choices between agree/disagree or good/bad.

Unary ratings can indicate that a user has observed or purchased an item, or otherwise rated the item positively.
The absence of a rating indicates that we have no information relating the user to the item (perhaps they purchased the item somewhere else).
Ratings may be gathered through explicit means, implicit means, or both. Explicit ratings are those where a user is asked to provide an opinion on an item. Implicit ratings are those inferred from a user's actions. For example, a user who visits a product page perhaps has some interest in that product while a user who subsequently purchases the product may have a much stronger interest in that product.

PROPOSED SYSTEM

The project is useful to the corporate stores to estimate the ratings of the products. It is also useful in directing the advertisements to the specifically interested customers. It finally leads to the quality of the services of corporate stores**.** It also uses the latest technology software which leads to estimate the effective ratings.

Using latest numpy & genfromtext techologies to deal with .csv data files. Using Python, which is chosen as the best programming language, by the Programming Community. More Functionality can be implemented with less number of lines of code in Python. PyQt tool is used to create the Graphical User interfaces**.** All the Front end code is generated automatically by PyUIC. Qt is designed for developing applications and user interfaces once and deploying them across several desktop and mobile operating systems.

The easiest way to start application development with Qt is to download and install Qt 5. It contains Qt libraries, examples, documentation, and the necessary development tools, such as the Qt Creator integrated development environment (IDE).

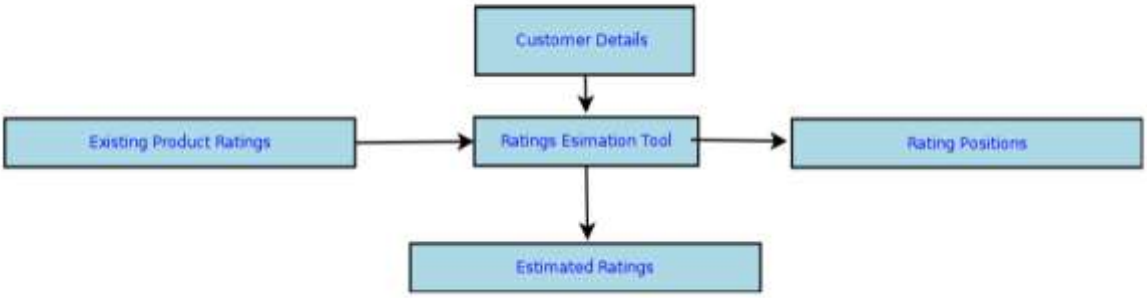
Qt Creator provides you with tools for accomplishing your tasks throughout the whole application development life-cycle, from creating a project to deploying the application on the target platforms. Qt Creator automates some tasks, such as creating projects, by providing wizards that guide you step-by-step through the project creation process, create the necessary files, and specify settings depending on the choices you

make. Also, it speeds up some tasks, such as writing code, by offering semantic highlighting, checking code syntax, code completion, refactoring actions, and other useful features.

Here we are using PyQt designer, the PyQt installer comes with a GUI builder tool called Qt Designer. Using its simple drag and drop interface, a GUI interface can be quickly built without having to write the code. It is however, not an IDE such as Visual Studio. Hence, Qt Designer does not have the facility to debug and build the application. Creation of a GUI interface using Qt Designer starts with choosing a top-level window for the application

SYSTEM ARCHITECTURE

This entry screen consists of six push buttons. Upon clicking the first button, customer1.py program is instantiated resulting in a screen, by using which the user can enter the customer details. The second button leads to the invoking of ratings1.py program, which results in a screen, where the user can enter the known product ratings. Upon clicking the third button, createcsv1.py program is instantiated resulting in the creation of .csv file, by uploading the existing ratings data in the database. Upon clicking the fourth button, the existence of the data file can be verified. The fifth button leads to the invoking of sim6.py program, which results in the identification of the positions to be estimated. The sixth button creates the output file with existing as well as estimated ratings.



ALGORITHM

Step1: Calculate the user's rating similarity
Input: User rating matrix
Output: User rating similarity set

```
def UserGradeSimilarity(gradetrain):
    W1 = dict()
      for u in train.keys(){
      for v in train.keys(){
          if u == v do continue
              W1[u][v] = len(train[u] & train[v])
              W1[u][v] = /= math.sqrt(len(train[u]) * len(train[v]) * 1.0)}
}Return W1
```

Step2: Calculate user's attribute similarity
Input: user attribute matrix
Output: User attribute similarity collection
Step3: fusion user's rating similarity and user's attribute similarity
Input: user rating similarity set user attributes similarity set
Output: user similarity set
Step4: User's prediction rating based on user similarity
Input: user's similarity set, nearest neighbor K, user rating matrix
Output: user prediction rating set

```
def Recommend(K, train, W):
    rank = dict()
    interacted_items = train[user]
    for v, wuv in sorted(W[u].items, key=itemgetter(1), reverse=True)[0:K]:
    for i, rvi in train[v].items:
      if i in interacted_items do continue
      rank[i] += wuv * rvi
    Return rank
```

Step5: Test set evaluation
Input: test set, user's prediction rating set
Output: mean absolute deviation

MODULES

I. *Data Collection*

Firstly, Dataset can be collected from various sources of any organization. The right dataset helps for the estimation of product ratings and it can be manipulated as per our requirement. Our data mainly consists of the details of the various customers about the various product ratings. The data can be collected from the organization based on the type of products for which the ratings are to be estimated.

II. *Data Processing*

In Item-Based Collaborative filtering, we compare two items and assume them to be similar when one user gives the two items similar ratings. We then predict that user's ratings for an item based on the preferences of similar customers using cosine similarity.
At the beginning when the data was collected, it contains the details about the points at which the ratings has to be estimated. By using cosine similarity we can find out about the details of the customers with similar taste and preference, we can estimate the ratings at the place where the rating has to be estimated.

III. *Training the Data*

After pre-processing the data, the ratings from the dataset are given to the cosine similarity function for training, then the positions where the ratings are to be estimated are identified. Now our function is ready to estimate the unknown ratings.

IV. *Deploying the Model*

After training the data the cosine similarity function will be able to predict the unknown ratings. It helps to give the output or accurate ratings at the points we selected for estimating ratings.

CONCLUSION

This project entitled “Collaborative Filtering of Product Ratings using cosine similarity.” is useful to estimate the unknown product ratings. The project is useful to the corporate stores to estimate the ratings of their products. The project is also useful in directing the advertisements to the specifically interested customers. This project finally leads to the improvement of quality of the services of corporate stores.

REFERENCES

[1] Adomavicius, G., Tuzhilin A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering.

[2] Aggarwal, C.C., Wolf J., Wu K.L., Yu P.S.: Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining..

[3] Avery, C., Resnick P., Zeckhauser, R.: The Market for Evaluations. American Economic Review.

[4] Balabanovíc, M., Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, 40(3): p. 66-72.

[5] https://www.python.org/

[6]https://github.com/baoboa/pyqt5/blob/master/pyuic/uic/pyuic.py

[7] https://www.numpy.org/

[8] https://riverbankcomputing.com/software/pyqt/intro

[9] https://www.ubuntu.com/