Problem:

In this project, we delve deep into movie recommendations by analysing diverse datasets from ImBD, Movie Lens, Kaggle, Rotten Tomatoes, and Netflix, as well as the OMBD and TMBD APIs. Our primary objective is to amplify the efficiency of movie recommendation systems to enhance the media viewing experience. We aim to transform the movie viewing data into a customer-centric dataset. We intend to develop a recommendation system to suggest the best movie for a given profile to customers within each segment who have yet to see these films, ultimately enhancing efficacy and fostering increased customer satisfaction.

Objectives:

- Research: Understand what makes an effective ML algorithm as a recommendation system.
- Data Cleaning & Transformation: Clean the dataset by handling missing values, duplicates, and outliers, preparing it for effective modelling.
- Feature Engineering: Develop new features based on the viewing data to create a customer-centric dataset.
- Recommendation System: Implement a system to recommend the most appropriate movies to users within the same viewing profile who have yet to view those movies to boost customer experience.

Research:

For research, I took inspiration from the following papers to form a balanced view of recommendation systems and models.

The BellKor Solution to the Netflix Grand Prize

Yehuda Koren

August 2009

I. Introduction

This article describes part of our contribution to the "BellKor's Pragmatic Chaos" final solution, which won the Netflix Grand Prize. The other portion of the contribution was created while working at AT&T with Robert Bell and Chris Volinsky, as reported in our 2008 Progress Prize report [3]. The final solution includes all the predictors described there. In this article we describe only the newer predictors.

So what is new over last year's solution? First we further improved the baseline predictors (Sec. III). This in turn improves our other models, which incorporate those predictors, like the matrix factorization model (Sec. IV). In addition, an extension

therefore harder to predict. In a way, this represents real requirements for a collaborative filtering (CF) system, which needs to predict new ratings from older ones, and to equally address all users, not just the heavy raters.

We reserve special indexing letters to distinguish users from movies: for users u, v, and for movies i, j. A rating r_{ui} indicates the preference by user u of movie i. Values are ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest. We distinguish predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}.

The scalar t_{ui} denotes the time of rating r_{ui}. Here, time is measured in days, so t_{ui} counts the number of days elapsed

Evaluating Recommendation Systems

Guy Shani and Asela Gunawardana

Abstract *Recommender systems* are now popular both commercially and in the research community, where many approaches have been suggested for providing recommendations. In many cases a system designer that wishes to employ a recommendation system must choose between a set of candidate approaches. A first step towards selecting an appropriate algorithm is to decide which properties of the application to focus upon when making this choice. Indeed, recommendation systems have a variety of properties that may affect user experience, such as accuracy, robustness, scalability, and so forth. In this paper we discuss how to compare recommenders based on a set of properties that are relevant for the application. We focus on comparative studies, where a few algorithms are compared using some evaluation metric, rather than absolute benchmarking of algorithms. We describe experimental settings appropriate for making choices between algorithms. We review three types of experiments, starting with an offline setting, where recommendation approaches are compared without user interaction, then reviewing user studies, where a small group of subjects experiment with the system and report on the experience, and finally describe large scale online experiments, where real user populations interact with the system. In each of these cases we describe types of questions that can be

electronics

MDPI

Review

A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields

Hyeyoung Ko [1,*], Suyeon Lee [2], Yoonseo Park [1] and Anna Choi [2]

[1] Department of Digital Media Design and Applications, Seoul Women's University, Seoul 01797, Korea; patpark7@swu.ac.kr
[2] Department of Computer Science & Engineering, Seoul Women's University, Seoul 01797, Korea; syou93@swu.ac.kr (S.L.); victoryanna@swu.ac.kr (A.C.)
* Correspondence: kohy@swu.ac.kr; Tel.: +82-2-970-5751

Abstract: This paper reviews the research trends that link the advanced technical aspects of recommendation systems that are used in various service areas and the business aspects of these services. First, for a reliable analysis of recommendation models for recommendation systems, data mining technology, and related research by application service, more than 135 top-ranking articles and top-tier conferences published in Google Scholar between 2010 and 2021 were collected and reviewed. Based on this, studies on recommendation system models and the technology used in recommendation systems were systematized, and research trends by year were analyzed. In addition, the application service fields where recommendation systems were used were classified, and research on the recommendation system model and recommendation technique used in each field was analyzed. Furthermore, vast amounts of application service-related data used by recommendation systems were collected from 2010 to 2021 without taking the journal ranking into consideration and reviewed along with various recommendation system studies, as well as applied service field industry data. As a result of this study, it was found that the flow and quantitative growth of various detailed studies of recommendation systems interact with the business growth of the actual applied service field. While providing a comprehensive summary of recommendation systems, this study provides insight to many researchers interested in recommendation systems through the analysis of its various

check for updates

Citation: Ko, H.; Lee, S.; Park, Y.; Choi, A. A Survey of Recommendation Systems:

Egyptian Informatics Journal (2015) **16**, 261–273

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

EGYPTIAN Informatics JOURNAL

REVIEW

Recommendation systems: Principles, methods and evaluation

CrossMark

F.O. Isinkaye [a,*], Y.O. Folajimi [b], B.A. Ojokoh [c]

[a] *Department of Mathematical Science, Ekiti State University, Ado Ekiti, Nigeria*
[b] *Department of Computer Science, University of Ibadan, Ibadan, Nigeria*
[c] *Department of Computer Science, Federal University of Technology, Akure, Nigeria*

Received 13 March 2015; revised 8 June 2015; accepted 30 June 2015
Available online 20 August 2015

KEYWORDS

Collaborative filtering;
Content-based filtering;
Hybrid filtering technique;
Recommendation systems;
Evaluation

Abstract On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. This paper explores the different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems.

Item-Based Collaborative Filtering Recommendation Algorithms

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
{sarwar, karypis, konstan, riedl}@cs.umn.edu
GroupLens Research Group/Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455

ABSTRACT

Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success on the Web. The tremendous growth in the amount of available information and the number of visitors to Web sites in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations, performing many recommendations per second for millions of users and items and achieving high coverage in the face of data sparsity. In traditional collaborative filtering systems the amount of work increases with the number of participants in the system. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems. To address these issues we have explored item-based collaborative filtering techniques. Item-based techniques first analyze the user-item matrix to identify relationships between different

through all the available information to find that which is most valuable to us.

One of the most promising such technologies is *collaborative filtering* [19, 27, 14, 16]. Collaborative filtering works by building a database of preferences for items by users. A new user, Neo, is matched against the database to discover *neighbors*, which are other users who have historically had similar taste to Neo. Items that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice, and in both information filtering applications and E-commerce applications. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems.

The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual users for whom the site has large amounts of information. For instance.

Palo Alto Research Center

Using Collaborative Filtering to Weave an Information Tapestry

David Goldberg, David Nichols, Brian Oki, and Douglas Terry

Datasets overview:

- Netflix - publicly available dataset released by Netflix for an optimisation challenge
- Movie Lens - publicly available data set
- Kaggle Challenge - publicly available dataset uploaded to Kaggle
- Rotten Tomatoes - publicly available data set
- Imbd - titles and ratings publicly available dataset
- APIs : ombd and tmbd, publicly available APIs

Columns overview:

- User profile (from user input):
    - userID
    - Name
    - Username
    - Email
    - Password
- User preferences (from user input):
    - Genres
    - age rating
    - user ratings
    - release year range
- CSV data from datasets stored in SQLAlchemy database:
    - movieID
    - title,
    - yrmade,
    - isAdult,
    - runtime,
    - Genres,
    - ratingavgscore,
    - actors,
    - company,
    - agerating,
    - tags,
    - votes,
- Data pulled from the API:
    - Title
    - genre(s)
    - rating
    - run time
    - age rating

Conclusion

This project successfully created a recommendation system for movie viewing. Going forward it would be great to use web scraping to further enhance the dataset and to be able to spend more time on data engineering to better extract, transform and load the data sets.

Appendix

Formulas used:

Cosine similarity

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}^{\intercal}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{y}^{\intercal}{}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}}$$

TF IDF algorithm

$TF - IDFscore(w_{ij}) = TF_{ij} * IDF_i$

Algorithm visualised: