CSE587: Data Intensive Computing

Book Recommendation System

Kaumudi Moholkar(50388592) Priyanka Dalit Tajane(50366282)

Dr Bina Ramamurthy

Abstract

Recommender system improve admittance to pertinent items and data by making a customized idea dependent on past instances of a user's preferences. Most existing recommender frameworks utilize social separating strategies that base proposals on other users' inclinations. Paradoxically, content-based techniques use data about a thing itself to make ideas. This methodology has the benefit of having the option to prescribe already unrated things to users with novel interests and to give clarifications to its proposals. We portray a content-based book recommender framework that uses data extraction and an AI calculation for text classification. Introductory exploratory outcomes exhibit that this methodology can deliver precise recommendations. These tests depend on appraisals from irregular samplings of things and we examine issues with past tests that utilize skewed samples of users-selected examples to assess execution.

Introduction

A recommender system figures and gives significant substance to the client upheld information on the client, substance, and connections between the client and thusly the thing. Suggestion frameworks are used in various administrations - wherever from internet shopping to music to films. For example, the web retailer Amazon had a significant hand in creating cooperative sifting calculations that prescribe things to clients. Music administrations like Pandora recognize up to 450 particularly distinguishing attributes of melodies to search out music practically like that of their clients' inclinations. Other music web-based features, like Spotify, vigorously depend upon the music determinations of comparable clients to make week after week melody suggestions and customized radio broadcasts. Netflix, a popular TV and film real time feature, utilizes these frameworks to suggest motion pictures that watchers may appreciate. We can perceive how proposal frameworks to a great extent affect the materials customers draw in with throughout their everyday lives.

For our undertaking, we analyzed five frameworks that foresee how users will rate explicit books. Our framework that we made makes these expectations dependent on information assembled from the BookCrossing dataset. To precisely anticipate users' responses to books, we've incorporated a few techniques in the field of recommendation frameworks.

Data Cleaning

We have three data files, Books, Users and Ratings. We merged these tables with inner join to obtain the unique features to work on. We dropped the columns which were not required like book images. In the tables, we performed data cleaning and initial analysis operations such as,

- Dropping non-essential columns.
- Finding duplicates and removing them.
- Finding null values and replacing them with default values.
- Data type modification for better calculations.
- Finding outliers and handling them.
- Extracting features like country from location column as the city, state was not required.
- Dealing with special characters.
- Validation of country as there was irrelevant data.
- Finding the demographics of data.
- Visualization of data.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA), otherwise called Data Exploration, is a stage in the Data Analysis Process, where various methods are utilized to more readily comprehend the dataset being utilized. 'Understanding the dataset' can allude to various things including however not restricted to extricating significant factors and giving up superfluous factors, distinguishing anomalies, missing qualities, or human blunder, understanding the relationship(s), or absence of, between factors. Eventually, boosting your bits of knowledge of a dataset and limiting potential mistake that may happen later simultaneously. We have implemented the below operations as the part of EDA.

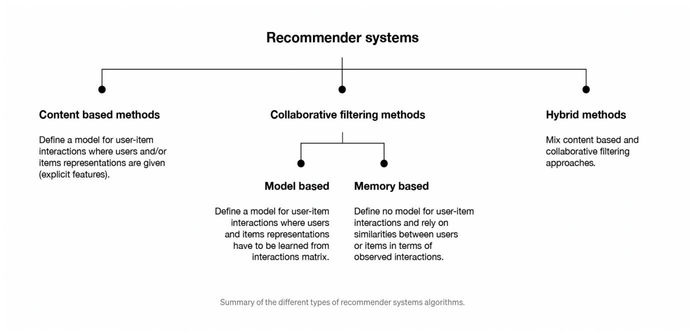
- Top 10 most rated books and their average rating to predict if, the most popular books are highly rated.
- Grouping of years to generate categorization based on years of publication, like ancient, recent, latest etc.
- Getting an understanding of how the users like to rate the books.
- Finding the top-rated books, as this will be the default recommendation for any user.
- Determining the number of books by an author which can give an idea about the favorite authors of the users.
- Understanding the diversity of readers based on age groups, country and the category of books.
- Publishing year of top-rated books to identify the relation among them.

<u>Methods</u>

In recommender systems, as with almost every other machine learning problem, the techniques and models we use are heavily dependent on the quantity and quality of the data we possess. There are three major approaches for recommendation systems:

- Content-based methods
- Collaborative Filtering methods
- Hybrid methods

Extensively, a content-based (CB) approach recommends items to a user that are like the ones the user liked previously. On the other hand, recommendation framework that carry out collaborative filtering (CF) anticipate users' inclinations by analyzing connections among users and interdependencies among items; from these, they extrapolate new associations. Finally, hybrid approaches merge content-based and collaborative approaches, which have integral qualities and shortcomings, along these lines creating more grounded results.

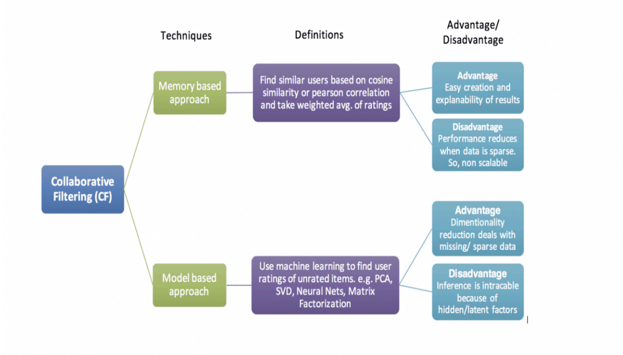


Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.
Summary of the different types of recommender systems algorithms.

Figure 1 - Types of recommender systems

<u>Collaborative Filtering</u>

Collaborative Filtering is the highly implemented and most mature of all the recommender models. This model is based on the assumption that if an item is liked by the user, then the similar will be liked in the future. These systems aggregate the ratings of the user and item then differentiate the commonalities and build a model for predicting the future item ratings. The CF model is majorly of two types. Memory based and Model based.



Techniques
Definitions
Advantage/ Disadvantage
Collaborative Filtering (CF)
Memory based approach
Find similar users based on cosine similarity or pearson correlation and take weighted avg. of ratings
Advantage Easy creation and explanability of results
Disadvantage Performance reduces when data is sparse. So, non scalable
Model based approach
Use machine learning to find user ratings of unrated items. e.g. PCA, SVD, Neural Nets, Matrix Factorization
Advantage Dimentionality reduction deals with missing/ sparse data
Disadvantage Inference is intracable because of hidden/latent factors

Figure 2 - Types of Collaborative filtering systems

<u>Memory-based Collaborative Filtering</u>

In this project we have implemented the memory-based model. Memory based is further of two types: Item-Item Similarity and User-User Similarity. In User-User model we take a particular user and find users that are similar to that user based on similarity of ratings. Then it will recommend items that are similar to the ones the users liked. In Item-Item model we take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and recommends other items.

Collaborative methods work with the interaction matrix that can also be called rating matrix in the rare case when users provide explicit rating of items. These systems try to predict the rating that a user would give an item-based on past ratings and preferences of other users. One of the benefits of these models, is that it doesn't require item metadata like its content-based counterparts.

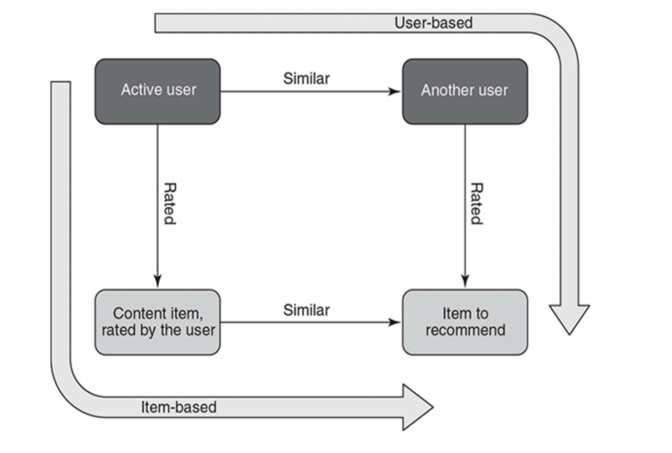


Figure 3 – Item-based and User-based model.

1. **User-User model**:

   Initially we calculated the similarity between the active user and all the rest of users. The users are then listed in order of similarity and a neighborhood is selected to calculate the predictions. There are various ways to decide the neighborhood, here we have used the rating matrix to calculate the similarities. The commonality among the users and their ratings are used to predict the item rating. The similarity metric used in this model is cosine similarity.

   The data is divided into two parts of training data of 80% and test data of 20%. The operations are performed on training data and the model is validated on the test data to see, how good fit the model is. The model is created on user id and ISBN of the books data features. The train and test matrices are created with users and books as their rows and columns. These matrices are then run on the cosine similarity to generate the User model. The model is then validated on the test data.

2. **Item-Item model**:

   In this model the similarity is calculated between the item to be predicted and all other items. The items are ordered by similarity and a neighborhood is selected to calculate the predictions of an item rating. Cosine similarity is used in this model. It provides a matrix of similarities that, for each item, provides a list of similar items. The below is the formula for computing predictions.

$$Pred(u,i) = \bar{r}_u + \frac{\sum_{j \in S_i} (sim(i,j) \times r_{u,j})}{\sum_{j \in S_i} sim(i,j)}$$

where,

- $\bar{r}_u$ is the average rating of the user *u.*
- $r_{u,j}$ is the active user's *u* rating of item *j.*
- S_i is the set of items in the neighborhood that user *u* has rated.
- *Pred*(*u*,*i*) is the predicted rating for user *u* of item *i*.
- *sim*(*i*,*j*) is the similarity between item *i* and item *j*.

The data is divided into two parts of training data of 80% and test data of 20%. The operations are performed on training data and the model is validated on the test data to see, how good fit the model is. The model is created on user id and ISBN of the books data features. The train and test matrices are created with users and books as their rows and columns. These matrices are then run on the cosine similarity to generate the Item model. The model is then validated on the test data.

We used the evaluation metric known as the root mean square error. This tells us on average how different our predictions are from the actual rating a user has given.

Evaluation Metric: RMSE

We calculate the root mean squared error using the following equation

$$RMSE = \sqrt{\frac{1}{n}\sum_{(i,j)}(p_{(i,j)} - r_{(i,j)})^2}$$

Where n is the total number of ratings, $p_{(i,j)}$ is the predicted rating and $r_{(i,j)}$ is the actual rating for a given book j, user i pairing. The RMSE value indicates on average the difference between our model's prediction from the actual rating observed.

The predicted ratings for the development data are informed by users' preference from the training data. We use this evaluation metric to find the best combination of variables (number of neighbors, similarity metric, and normalization) for predicting ratings.

The RMSE evaluated is approximating around **7.813** for Item based and User based.

Model-based Collaborative Filtering

One of the most commonly used methods for matrix factorization is an algorithm called SVD (singular value decomposition). You want to find items to recommend to users, and you want to do it using extracted factors from the rating matrix. The idea of factorization is even more complicated because you want to end up with a formula that enables you to add new users and items without too much fuss.

We used the SVD-based Matrix factorization technique to recommend books to users. While SVD is not best-suited for sparse matrices, we choose this model due to its simplicity and effectiveness when using explicit data. Other optimization methods like ALS are not very effective in case of large explicit ratings matrices.

Instead of assuming that a user u's rating for item i can be described simply by the dot product of the user and item latent vectors, we will consider that each user and book can have a bias term associated with them. The rationale is that certain users might not be very critical and tend to rate all books highly, or certain books may tend to always have low ratings. We also include a global bias term. These biases are learnt by the model. So, for this model, the prediction is set as:

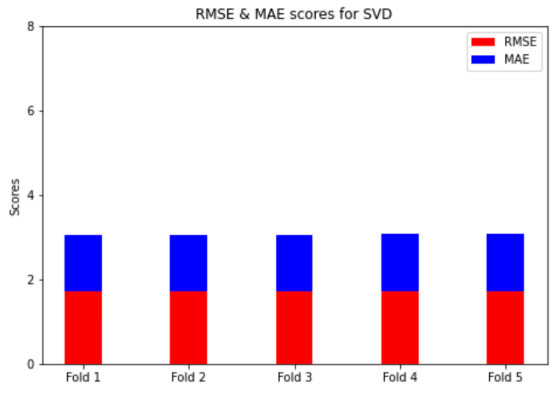
$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Since the Memory based approach gave an RMSE of 7.813, we could try to improve the value by using a SVD-Model based approach.

We have used the surprise library for implementation of SVD. The data is split in 80% of train data and 20% of test data. The RMSE and MAE of the SVD is calculated for the test data and we obtained a value of **1.7178** which clearly shows that the RMSE has improved as compared to the memory-based approach. We have implemented the RMSE and MAE of the Non-Negative Matrix Factorization(NMF) as well which has a value of around **2.31**. This turns out that RMSE of SVD is better than the NMF and we can use the model-based approach with the SVD method.

Two ways you can make recommendations now are to calculate all predicted ratings and take the largest ones that the user hasn't seen before, or to iterate through each item and find similar products in the reduced space. A third way could be to use the new matrices you have to calculate neighborhood collaborative filtering. Using the SVD, we have implemented the recommender and obtained the book titles and their predicted ratings with respect to a particular user. The model has also been tested on an existing book rating and obtained a very small error factor.

The SVD we've implemented so far has few problems: To begin with, it requires that something is done with the unfilled cells in the rating matrix. And it's slow to calculate large matrices. On the positive side, there's the possibility to fold in new users as they arrive, but the SVD model is static and should be updated as often as possible. Therefore, we have implemented the next model Content Based Model.



Plot 1- RMSE and MAE scores for SVD

Content Based Recommendation

Content Based recommenders do not require past data like Collaborative filtering and hence they are a little complicated. They require few high-level steps because it's about extracting knowledge from the given content. Initially, a quantitative model is generated to represent each of the products. These models should represent something about the actual content of the product. Based on these feedbacks, the algorithm builds a user profile. This helps in accurately predicting the user's ratings for modeled items. This classifies in presenting the products that has not yet been tried and would be highly rated by the user.

CB recommendations are particularly useful on datasets which have relatively few ratings or no ratings. This is highly useful as a lot of people do not tend to rate products. We refer to this as 'sparsity'. Since content-based systems don't leverage the power of the community, they often come up with results that are not as impressive or relevant as the ones offered by collaborative filters. In other words, content-based systems usually provide recommendations that are *obvious.* There is little novelty in a *Lord of the Rings* recommendation if *Harry Potter* is your favorite movie.

Our first task is to model the books in our datasets. We chose two different approaches to doing so, both of which produced one vector of real numbers per book. For both approaches, we preprocessed the text of all descriptions and reviews we had for a particular book to remove punctuation, ensure all letters were lowercase, and lemmatize each word to its base form.

Probabilistic strategy like that of Naïve Bayes Classifier, Linear Classifiers and Support Vector Machines and furthermore different strategies like Term Frequency-Inverse Document Frequency, Nearest Neighbor and Relevance Feedback are utilized for content-based proposals. We have implemented our model using Term frequency-inverse document frequency (TFIDF). These terms help in sorting and scoring of the books on which the analysis is based on. The model was implemented on the combined fields of Book-Title, Publisher, Author, an average score was generated for each word appearing in the text describing the book. We decided to try this approach because the vast majority of the text we had describing any given book consisted of very opinionated ratings, and trying a method of modeling the books that leveraged that fact made sense. The similarity was built on the combination of book title, author and publisher. The similarity between the books was calculated using cosine similarity. In the second approach we calculated the decade of the years over which the books were published to calculate the categories of books read by the users.

Because we used classifiers to make predictions about whether or not users will like or dislike books, we chose to evaluate our results with three statistics: accuracy, precision, and recall. The accuracy of our classifier tells us the proportion of predictions we make that are correct, including correctly guessing a user would like a book, and correctly guessing a user would dislike a book:

(True Positives + True Negatives) / All Predictions

The precision tells us how many books were actually liked by a user out of all the books we predict that user would like:

True Positives / (True Positives + False Positives)

The recall tells us how many books our classifier predicts users would like out of all the books those users would actually like:

True Positives / (True Positives + False Negatives)

These three statistics can be interpreted together to give us a thorough understanding of the strengths and weaknesses of our content-based rating prediction system. When choosing between classifiers and data combinations, we chose to prioritize the accuracies.

Clustering- kNN Model

kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors. For example, we first present ratings in a matrix with the matrix having one row for each item and one column for each user.

We then find the k item that has the most similar user engagement vectors. Now, let's implement kNN into our book recommender system. It is possible to use a clustering algorithm, such as k-means, to group users into a cluster and then take only the users from the same cluster into consideration when predicting ratings.

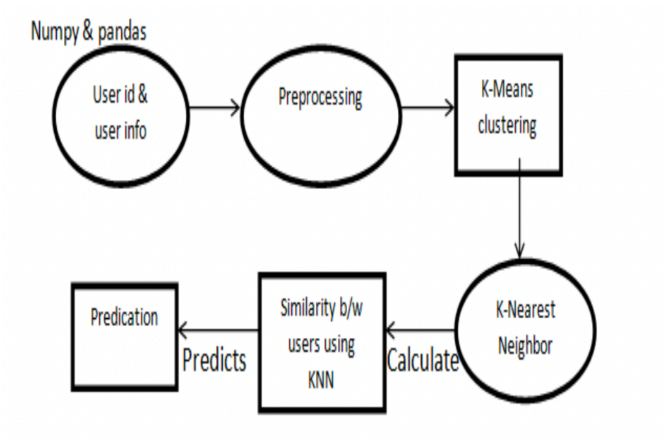


Figure 4 - Flow Diagram for kNN Algorithm

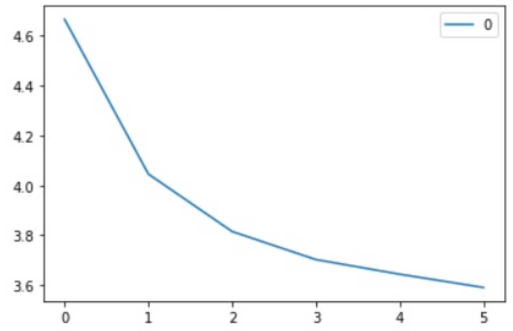
In this section, we will use k-means' sister algorithm, kNN, to build our clustering-based recommender. In a nutshell, given a user, *u*, and a book, *b*, these are the steps involved:

1. Find the k-nearest neighbors of *u* who have rated book *b*
2. Output the average rating of the *k* users for the book *b*

We have implemented unsupervised algorithms with "sklearn.neighbors" library. The algorithm we used to compute the nearest neighbors is “brute”, and we specify “cosine metric” so that the algorithm will calculate the cosine similarity between rating vectors. Finally, we fit the model.

The data has been limited to US and Canada users as we might run into memory issues, if implemented on large amount of data. All the books which are in the neighboring of k=6, with the least average distance is recommended based on their rating.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff522295d50>
```



Plot 2 – plotting the RMSE values against the k values

When we take k=1, we get a very high RMSE value. The RMSE value decreases as we increase the k value. At k= 6, the RMSE is approximately 3.59, and remains almost constant as we keep increasing the k.

Hybrid Model

A blend of CB and CF procedures, alluded to as mixture approach, eases a portion of the issues that every recommender experiences separately. Infact numerous fruitful frameworks rely upon the two procedures to make suggestions.

To begin with, we'll set up the SVD CF recommender, and the content-based recommender. We picked these models since, they performed better when contrasted with different models. At the point when you sign in interestingly, Book Recommender beats the cold start problem of collaborative filters by utilizing a content-based recommender, and, as you steadily begin reading and rating books, it brings its collaborative filters into play. This is undeniably more effective, so most practical recommender frameworks are hybrid in nature.

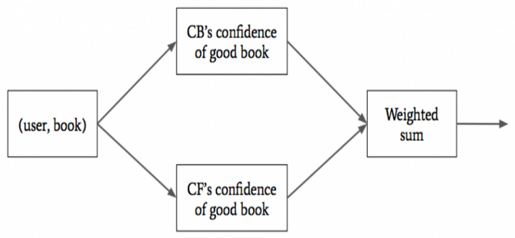


Figure 5 - Hybrid Model flow

First, we have implemented the content-based filter on the books and their ratings. Their similarity scores and book indices are calculated for the books. These indices are sorted in descending model and the top 50 books are calculated. This data is stored in a form of a list. This list is then implemented with the SVD CF model, which outputs an estimation value of the required books. The estimated value is sorted in the descending order and top 10 books are displayed. Hence, the best of both worlds is obtained by implementing the CF and content-based model.

A recommender system can be greatly optimized by adding the output of several algorithms. Hybrid recommenders enable you to combine the forces of different recommenders to get better results.

Analysis

Recommender frameworks are an incredibly intense device used to help the determination interaction simpler for users. With implementation of various models, we have observed that every model is unique in its implementation and output. CF model is based on previous user’s ratings. The users conduct may incorporate recently watched recordings, bought things, given evaluations on things. Thus, the model discovers a relationship between the users and the items. The model is then used to anticipate the item or a rating for the item in which the users might be intrigued. Content-based methodology is based with respect to a depiction of the thing and a record of the user's inclinations. It utilizes a succession of discrete, pre-labeled attributes of a thing to suggest extra things with comparative properties. This methodology is most appropriate when there is adequate data available on the items yet not on the users. Thus, using the benefits of both models, we created the Hybrid model.

Discussions

The design was executed in python programming dialects using surprise, SciPy, K-NN means. The control framework achievement comprises of numerous post-areas that are additionally typical strategies to be went with while settling any profound learning issue.

The primary stage in the advancement strategy is the gathering of the data. The proper arrangement of information is picked in this stage regarding accomplishing extra estimations. The data set is gathered from books recommendation program BX Books. The dataset contains 1048576 books from (1-10) positioning. Moreover, it has 276272 purchasers and 271380 books. With that sort of information, extra estimations are completed utilizing programming bundle Python.

The second measure in the organization stage is the technique for getting ready data. Preprocessing of the data is acted in this stage. It portrays the extra force vector that advises which users has arranged that book. This is cultivated by partitioning user's data and book information all through numerous information blocks, first. At that point, the vector factorization is created simply utilizing the data focuses.

The subsequent choice in the organization stage is the preparing of the data. The gathering strategy for CF, content and hybrid model is carried out in this stage. The appropriate measure of bunch is chosen. Utilizing the K-Means Segmentation framework, prior to choosing the right no of gathering books are isolated across gatherings. RMSE is estimated to decide the creator's dependability.

Conclusion

The entirety of our frameworks content-based, collaborative-filtering, and hybrid performed very well. Thinking back on the project, one thing that we may have decided to do another way by and large would have been to invest more energy looking for a dataset of evaluations with a higher rating difference for every user. Had we had the option to discover such a dataset, our executions of algorithms would have been tested on information that would have been more representative of what a regular business suggestion framework could access in making its expectations. Notwithstanding, given the information that was accessible to us, just as the outcomes our different methodologies delivered, our recommenders were to a great extent effective, giving understanding into how the various approaches we routinely use work and the varying algorithms that make that conceivable.

References

- Practical Recommender System book by Kim Falk
- Hands-on Recommendation system with Python by Rounak Banik
- Recommendation Systems — Models and Evaluation by Neeraja Doshi
- G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of the-art and possible extensions," IEEE Trans. Knowl. Data Eng.
- Xiaoyuan Su and Taghi M Khoshgoftaar. "A survey of collaborative filtering techniques." In: *Advances in artificial intelligence* 2009 (2009), p. 4.
- Introduction to Recommender Systems Handbook, Francesco Ricci, Lior Rokach, Bracha Shapira, F. Ricci et al. (eds.), Recommender Systems Handbook.
- Ramakrishnan, G. (2020). Collaborative filtering for book recommendation system. SocProS.
- Shah, K. (2019). Book Recommendation System using Item based Collaborative Filtering . International Research Journal of Engineering and Technology .
- Using Cosine Similarity to Build a Movie Recommendation System by Manhoor Javed
- Evaluating Recommendation Systems by Guy Shani and Asela Gunawardana