Active Recommendation Method Based Mobile Network Using Book Management Application

M. B. Chung

***Abstract*—This paper suggests a mobile-based book recommendation method utilizing a book management application. It will help smart phones to recognize Barcodes and QR codes and collect information on books through the open APIs. Such information is divided into that of books already purchased and that of the books the user would like to buy. This divided information is stored in an application, while the server measures the similarity among users in terms of their information stored in the database created. Then, if it is found that users with similar preferences do not have certain books, recommendations for the books will be made through push notification; the users receiving the recommendations get the book details directly from the server. To evaluate the performance of the proposed method, 300 users were selected as subjects and the success rate was 85%. Therefore, the proposed method will be useful in a mobile-based recommendation system.**

***Index Terms*—Smartphone, book management application, QR code, recommendation method.**

I. Introduction

The development of smart phone technologies and mobile environments has popularized the use of smart phones. In addition, activities for developing the applications to be used in smart phones are now expanding from the corporate level to the individual level. We could say that most of the smart phone applications in the early days were created at a corporate level or by device-making companies that commissioned application development companies to do the job. Currently, however, individual enterprises and developers have registered their own IDs in application markets, such as Apple, the Android market, and SK T-Store; they have developed and registered various applications, and are providing related services.

There have also been some changes in trends related to the creation of applications. Games were the leading applications in the smart phone market in the early days. However, a variety of applications with diverse themes that can be applied to daily life, such as household accounting books, English learning, navigation, various psychological tests, and classical music, have just started to emerge. Services that use smart phone cameras have also been developed. In the game sector, Augmented Reality (AR) games, in which characters or enemies are displayed on the camera overlay through the marker recognition function, have emerged [1], [2]. Navigation apps (App: Application) based on AR can lead a user to a destination by displaying the targeted destination and the distance to it on the camera overlay, utilizing the smart phone's GPS and Mapkit framework [3], [4].

Technology based on smart phone cameras has been applied not just for AR games and navigation functions, but, in many cases, also for the recognition of objects or markers. One application, called Pudding Camera, provides a service that uses Open CV to recognize a face and match a photo of the face to an entertainer with a similar image. QRDic and QROO–QROO, well known for recognizing QR (Quick Response) codes, provide information that compares the prices of goods by recognizing their QR codes. Such research regarding code recognition technologies has been mostly carried out based on existing computer and mobile platforms [5], [6]. While those technologies could have been developed internally, free libraries that have already been developed are used in many cases. Representative code recognition libraries include Zxing, developed based on the Java language, and Zbar, based on the C++ language [7], [8]. As those recognition technologies do not have a great deal of data, most applications provide data in detail via internal servers or Open APIs by utilizing networks. As an example, in the case of the advertisement for Avante in 2010, which used a QR code, data for the relevant web address were just planted in the QR code while providing the actual details, as well as video data, through a web server. QROO–QROO also provides detailed price comparison information through the Open API of Daum.

This paper not only introduces a mobile book management application that utilizes the barcode and QR code recognition technologies of iPhone, but also suggests a mobile-based book recommendation method designed on the application. It will help smart phones to recognize Barcodes and QR codes and collect information on books through the open APIs of Google (http://Google.com) and Naver (http://naver.com). From the information that has been collected, a user can check the author, the publishing company, the publishing date, the book summary, and the price; when necessary, the information can be divided and separately saved into Scan List and My List. Information on the books not yet purchased by a user is classified into Scan List and that of the books purchased and kept by the user is sorted into My List. Then, the stored information for the book recommendation service is sent to a server upon the approval of the user. The data sent to the server are used to measure the similarity among users based on the lists they maintain. Then, the books of users with similar preferences are checked, and if it is found that certain

Manuscript received August 15, 2018; revised February 12, 2019. This research was supported in part by Ministry of Education, under Basic Science Research Program (NRF-2013R1A1A2061478 and NRF-2016R1C1B2007930), respectively.

M. B. Chung is with the Division of Computer Engineering, Sungkyul University, South Korea (e-mail: nzin@sungkyul.ac.kr).

users do not have a specific book, recommendation for the book will be made through Apple Push Notification Services (APNS). This method is very different from the existing book applications that recommend books to individual users. Our novel method can consistently pass recommendations to a user whose book preferences are similar to other users. Accordingly, the suggested book recommendation method is a useful technology that can help users manage books through an application on his/her iPhone.

The remainder of this paper is organized as follows: Section II introduces various web-based recommendation methods and book applications that are similar in function to the proposed application. Section III explains the configuration of the developed application and the overall structure of the system, which are necessary for applying our suggested method. There is also an explanation of our suggested recommendation method. Section IV presents the tests conducted for judging the validity of the developed application, based on the design of Section III as well as that of the suggested recommendation method along with the results. Section V concludes the paper.

II. Previous Work

This section introduces the current various web-based recommendation methods. Then, we introduce the existing book management applications that are similar in function to the proposed application.

A. Various Recommendation Methods

Providing a recommendation involves predicting and suggesting what a user might want. In early web services, recommendation methods classified similar users and recommended to each group, because computing speeds were slow. The existing group classification methods are Support Vector Machine (SVM), Bayesian networks, and clustering. Support Vector Machine is a machine learning method used for classification and regression analysis [9]. It often uses bio-information, statistics, and image processing fields. Through predictions according to the classification role, SVM constructs new classification roles from the given input data and classifies new input data. A standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input. Boser, Guyon, and Vapnik proposed a way to create nonlinear classifiers by applying the kernel trick [10]. In addition to this, multiclass SVM reduces the single multiclass problem into multiple binary classification problems and sorts input data into several classifications. Error-Correcting output codes and DAGSVM are a kind of multiclass SVM [11], [12]. Clustering includes k-NN clustering, Gaussian clustering, and Gaussian Mixture Model (GMM) clustering. Among these, k-NN is often used for classification, because it has the ability to calculate input data simply.

Because of improvements in computing speed, the recommendation method has changed to allow personal classification of items. It calculates the evaluation value of each person or item and estimates the similarities between persons or items. The most recently developed methods for recommending include user-based collaborative filtering [13], [14] and item-based collaborative filtering, which has gained notoriety through its use at Amazon.com [15], [16]. In the case of user-based collaborative filtering, similarities between users predict the level of preference shown by other users toward a specific item, while item-based collaborative filtering predicts other users' preferences for specific items by collecting a variety of evaluations about those items and measuring the similarities between them. Both methods require measurements of similarities between users or items. The cosine-based method and a Pearson correlation coefficient-based method are used for measuring similarity. The cosine-based similarity measurement is shown in Eq. (1).

$$\mathrm{sim}(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2} \tag{1}$$

When used in the item-based collaborative filtering method, ***i*** and ***j*** are the points of the preference given by users for two items. When used in the user-based collaborative filtering method, ***i*** and ***j*** are the points of the preference given by two users for the items. The Pearson correlation coefficient-based similarity measurement is shown in Eq. (2).

$$\mathrm{sim}(i,j)=\frac{\sum_{u\in U}(R_{u,i}-R_i)(R_{u,j}-R_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-R_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-R_j)^2}} \tag{2}$$

In Eq. (2), which represents the measurement of the similarity between items, *i* and *j* denote each item, and R_i and R_j denote the average values of the preference shown by the users for each item. $R_{u,i}$ and $R_{u,j}$ denote the evaluations given by individual users as the items *i* and *j* are evaluated in terms of preference. sim(*i*, *j*) denotes the similarity between items *i* and *j*. In the measurement of similarity between users, *i* and *j* denote each user. R_i and R_j denote the averages of all the values of the preferences shown by individual users for each item. $R_{u,i}$ and $R_{u,j}$ denote the evaluation values of preference given by individual users for item *u*. sim(*i*, *j*) denotes the similarity between users *i* and *j*. Either the weighted sum method or the regression method can be used to predict preference once the similarity is measured. In the case of the item-based collaborative filtering method, the preference can be predicted as shown in Eq. (3).

$$P_{u,x}=\frac{\sum_{\text{all-similar-items},N}(\mathrm{sim}(x,N)\times R_{u,N})}{\sum_{\text{all-similar-items},N}(|\mathrm{sim}(x,N)|)} \tag{3}$$

$P_{u,x}$ is the predicted preference shown by user *u* toward item *x*, $P_{u,N}$ is the preference value already entered by user *u* for item *N*. *sim*(*x*, *N*) is the similarity between item *x* and item *N*. In the case of the user-based collaborative filtering method, the preference can be calculated as shown in Eq. (4).

$$P_{u,x}=R_u+\frac{\sum_{j\in\text{Raters}}(R_{j,x}-R_j)\times\mathrm{sim}(u,j)}{\sum_{j\in\text{Raters}}|\mathrm{sim}(u,j)|} \tag{4}$$

R_u and R_j are the average preference values given by users u and j for the items. $sim(u, j)$ is the similarity between users u and j. $R_{j,x}$ is the preference shown by user j toward item x. Raters is the collection of users that evaluate the test items in terms of preference.

Accordingly, when it comes to accurately recommending items to users, Eq. (2) and Eq. (3) can be used for cases requiring item-based collaborative filtering, and Eq. (2) and Eq. (4) can be used for cases requiring user-based collaborative filtering.

B. Existing Book Management and Recommendation Applications

GoodReads, a privately run "social cataloging" website, was started in December 2006 by Otis Chandler, a software engineer and entrepreneur. In 2010, an iPhone application was published by Goodreads. In 2011, GoodReads had 5.2 million members [17] and introduced an algorithm to suggest books based on a user's library [18]. This application uses the database from the GoodReads website. The GoodReads application can scan Barcodes and search for book information. It can recommend a book to an application user through various methods, such as websites.

MyBookDroid, which is used on Android OS, is another book management application. It manages the user's books as if the user has the book, as if the user wants to buy the book, or as if the user is reading the book. It recommends similar books to the user based on item-based collaborative filtering.

The upside of both applications is that they provide barcode scan functions that are used to search book information and book recommendation functions for the application user. However, if the application user wants to recommend a book, he/she has to launch the application and estimate the books.

III. The Application Design and Recommendation Method

This section will describe the overall structure of the suggested book management application system and how to provide a recommendation using the data transferred from the application.

A. Design of the Book Management Application

The book management application will search through books in sequence, as shown in Fig. 1.

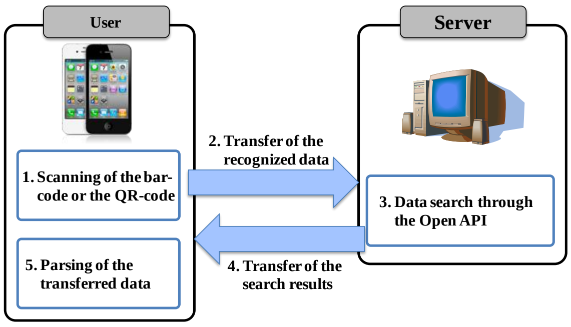

User
Server
1. Scanning of the bar-code or the QR-code
2. Transfer of the recognized data
3. Data search through the Open API
4. Transfer of the search results
5. Parsing of the transferred data

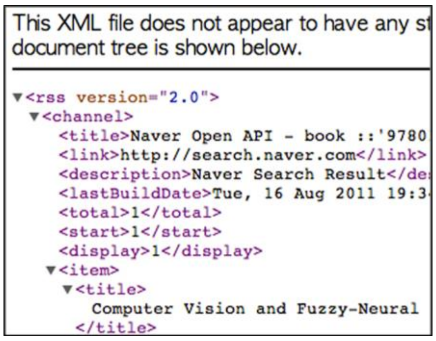
Fig. 1. Book search application process.

A user who wants to store information about a book needs to scan the book's barcode or QR code using an iPhone camera, and then the automatically recognized data will make a request to the Naver API and the Google API in order to obtain more detailed information. The reason why the Naver API and the Google API need to be used at the same time is because most Korean books can be searched through the Naver API while some foreign books cannot, so the Google API needs to be used for supplementing the search for foreign books that would not be otherwise searched. The request search method is different for the Naver API and the Google API, as shown in Table I.

TABLE I: Book Search Example Using Naver and Google API

API	Example of the search address
Naver	http://openapi.naver.com/search?key=user_key&query=9780135705995&target=book&d_isbn=9780135705995
Google	https://www.googleapis.com/books/v1/volumes?q=9780135705995+isbn:9780135705995&key=user_key

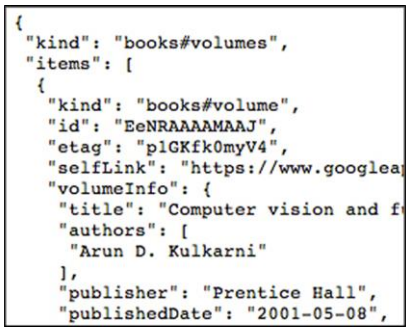
The Naver and Google APIs deliver key values, which are assigned to users. The number of requests made corresponding to the key values will be counted. In the case of Naver, a maximum of 25,000 requests can be made for one key for one day; in the case of Google, there is no limit to the number of requests that can be made, as Google runs a lab-based operation. As it is required for the data recognized from the barcode and QR code to be transferred as the search word, the value of the query to be searched needs to be entered after the query for Naver, and the target needs to be set for searching books. For more detailed queries, the desired result can be obtained when d_isbn, an additional query statement, is prepared when the search is carried out using the ISBN for books. In the case of Google, the form will be made shorter by preparing the query language after 'q' and the ISBN, a transfer factor that needs to be used as an additional search word to progress the searching process. In other words, the query request, as described in Table 1, shows that the book requested through both Naver and Google is a book whose ISBN is 978-01-3570-599-5. When it comes to the searching of results, Naver transferred a result in the form of Extensible Markup Language (XML), as shown in Fig. 2, and Google transferred a result in the form of JavaScript Object Notation (JSON), as shown in Fig. 3. In the configuration of the XML data shown in Fig. 2, <title> refers to the title of a book and <link> shows the address where the detailed information exists. In the configuration of the JSON data shown in Fig. 3, "title" in "volumeInfo" represents the title of the book, "authors" represents the author, "publisher" represents the publishing company, and "publishedDate" represents the publishing date of the book.



This XML file does not appear to have any st
document tree is shown below.

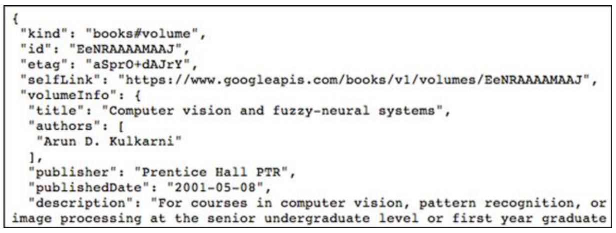
```
▼<rss version="2.0">
  ▼<channel>
     <title>Naver Open API - book ::'9780
     <link>http://search.naver.com</link>
     <description>Naver Search Result</de
     <lastBuildDate>Tue, 16 Aug 2011 19:3
     <total>1</total>
     <start>1</start>
     <display>1</display>
    ▼<item>
      ▼<title>
         Computer Vision and Fuzzy-Neural
       </title>
```

Fig. 2. Search result using Naver API.



```
{
 "kind": "books#volumes",
 "items": [
  {
   "kind": "books#volume",
   "id": "EeNRAAAAMAAJ",
   "etag": "plGKfk0myV4",
   "selfLink": "https://www.googlea
   "volumeInfo": {
    "title": "Computer vision and f
    "authors": [
     "Arun D. Kulkarni"
    ],
    "publisher": "Prentice Hall",
    "publishedDate": "2001-05-08",
```

Fig. 3. Search result using Google API.



```
{
 "kind": "books#volume",
 "id": "EeNRAAAAMAAJ",
 "etag": "aSprO+dAJrY",
 "selfLink": "https://www.googleapis.com/books/v1/volumes/EeNRAAAAMAAJ",
 "volumeInfo": {
  "title": "Computer vision and fuzzy-neural systems",
  "authors": [
   "Arun D. Kulkarni"
  ],
  "publisher": "Prentice Hall PTR",
  "publishedDate": "2001-05-08",
  "description": "For courses in computer vision, pattern recognition, or
image processing at the senior undergraduate level or first year graduate
```

Fig. 4. Updating process for similarity measurements between new user and the existing users.

In the case of the Google API, there will be one more request for the address of "selfLink" in order to acquire data from the detailed description and, as shown in Fig. 4, "description" will deliver a detailed description. The result acquired through the search request will be sent back to an iPhone. Then, in order to process the transferred data, the Naver API uses the NSXML Parser and the Google API uses the JSON Parser. The data processed using each Parser will be displayed on the screen showing the book cover, the title, the author, the publisher, the publishing date, and the detailed website link.

Then, to manage their books, users can store the results in Scan List or My List. At this time, it would be necessary to show the information in My List by author, so it is required to process data for listing books by author. The relevant configurations of the book and author information are shown in Fig. 5.

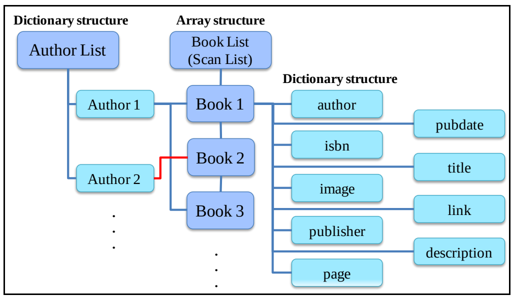


Fig. 5. Book search result data processing structure.

The Book List (Scan List) shown in Fig. 5 is used for displaying information from Scan List and My List. The detailed information data is shown in the dictionary structure to the right side of the book. When it is necessary to generate a list by author, the individual books connected to the author will be retrieved through Author List. Then, the initial data of the page node below each book is 0 and the node will be used when a user later enters the number of pages read. The comment node is for a brief comment by the user after he/she has read the book and the evaluation section is to store the user's evaluation points.

B. Recommendations through the Book Management Application

Recommendations using the suggested book management application are processed in the sequence shown in Fig. 6. When a user completes the evaluation of a book in My List of the book management application, the evaluation points are transferred to the server. Then, the transferred data are put into the user evaluation database. At the same time, other users with similar preferences are judged to see whether they have the same book. The structure of the database table on which users can enter their evaluations is shown in Table II.

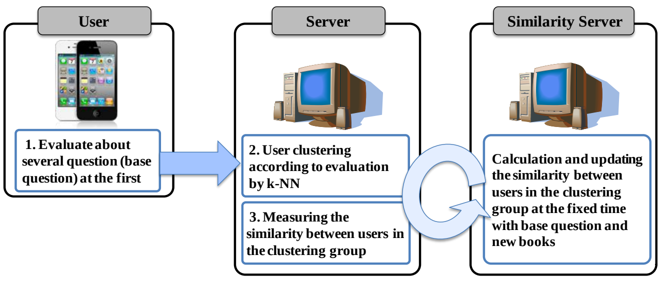

User
Server
Similarity Server
1. Evaluate about several question (base question) at the first
2. User clustering according to evaluation by k-NN
3. Measuring the similarity between users in the clustering group
Calculation and updating the similarity between users in the clustering group at the fixed time with base question and new books

Fig. 6. Flow of the recommendation method based on the proposed application.

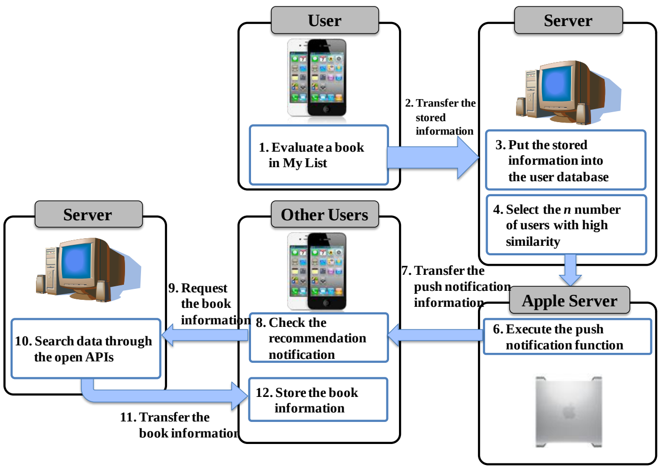

User
1. Evaluate a book in My List
2. Transfer the stored information
Server
3. Put the stored information into the user database
4. Select the *n* number of users with high similarity
Apple Server
6. Execute the push notification function
7. Transfer the push notification information
Other Users
8. Check the recommendation notification
9. Request the book information
Server
10. Search data through the open APIs
11. Transfer the book information
12. Store the book information

Fig. 7. Updating process for similarity measurements between new user and the existing users.

TABLE II: TABLE SCHEMA FOR USER EVALUATIONS

Field	Type	Description
no	int	Index
isbn	varchar(20)	Book ISBN number
title	varchar(127)	Book title
evaluation	int	Evaluation point given by user
reg_date	int	Date of the evaluation

There will be one single user evaluation table for each user. Every time the user evaluates a book, the book information, such as its isbn, title, evaluation point, and evaluation date, is put into the user evaluation table. Then, if the evaluation point is lower than a certain value, the next activities will not take place so that a recommendation is not provided.

For similarity measurements of each user, the system classifies user-based groups using k-NN classification according to early input data of the user under the supposition that it can be many users, and then measures user similarity values using early input user data, as shown in Fig. 7. When a new user is registered, the system calculates only similar users and the similarity value for the new user. Updated similarity values for all users are calculated at regular intervals on another server computer so that there is no effect on the recommendation service.

For reflection similarity values for new books, the system saves the evaluation count for each book; if the evaluation count of the book is over the threshold, the system includes the book in the similarity value update.

In this paper, we measure similarity by using (2) and (4) of Section II.A, which are constructed on user-based collaborative filtering methods. Users with high similarity are put into a table structure, as shown in Table III.

TABLE III: TABLE SCHEMA SHOWING SIMILARITY AMONG USERS

Field	Type	Description
no	int	Index
user	varchar(20)	User name
R1	varchar(20)	User1 with high similarity
R2	varchar(20)	User2 with high similarity
R3	varchar(20)	User3 with high similarity
…	…	…
R10	varchar(20)	User10 with high similarity
E1	int	Similarity with User1
E2	int	Similarity with User2
…	…	…
E10	int	Similarity with User10

R1, R2, R3, …, R10 fields in the table show the top 10 users with high similarities to the current user, and the E1, E2, E3, …, E10 fields show the similarities among them. Of the selected ‘n’ number of users, those who already have the same book are excluded from the list to be sent to APNS. Only the users who do not have the book are added to the list and the request for the relevant service is made to the Apple server. Then, the Apple server provides a recommendation, including the book title and the ISBN information, for the targeted users through the Push Notification function. Users who receive the recommendation can obtain the information for the recommended book through the Naver or Google APIs when they check the notification. Accordingly, when a user evaluates a book in his/her own list, the information will be utilized to automatically recommend the book to other users with high similarity values, such as in Fig. 8.

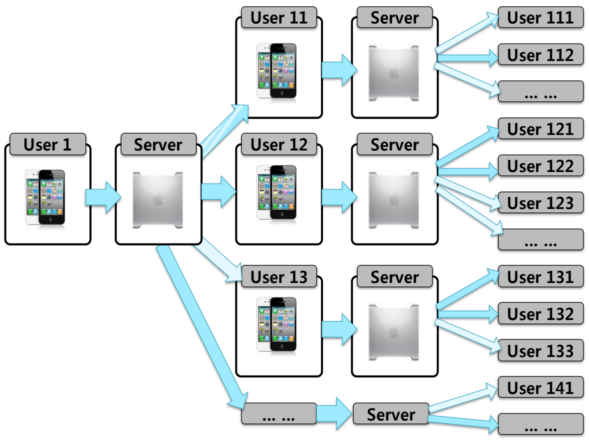


Fig. 8. Recommendation process for the proposed method.

If ***User1*** evaluates a book as satisfactory, the server sends a book recommendation message to ***User11, User12, …*** and ***User1N***, according to the satisfaction value of ***User1***. And, if ***User11, User12, …*** and ***User1N*** evaluate the book as satisfactory, the server sends a book recommendation message to ***User111, User112, … User1NN*** again. The proposed recommendation method does not yet have a website or existing book application; however, the proposed application will be able to actively recommend books to users. The proposed method is expressed in Eq. (5).

$$\begin{aligned} & if\,(R_{u,x} > \theta)\{ \\ & \quad GR_{u,x} = G_{u,k} \wedge (G_{j,x})^c \\ & \quad GP_{u,x} = G_{u,k} \wedge G_{j,x} \\ & \} \end{aligned} \tag{5}$$

$R_{u,x}$ denotes satisfaction value about book item x of user u, and θ is the threshold for the recommendation occurrence. $GR_{u,x}$ denotes a set of users who can be recommended, and $G_{u,k}$ denotes a set k of k-NN which includes user u. Thus, $G_{j,x}$ is a set of users who have evaluated book x and $(G_{j,x})^c$ is a set of users who have not evaluated book x. We can get $GR_{u,x}$ from Eq. (5) and predict the value of the recommended users, as shown in Eq. (6).

$$P_{i,x} = R_{u,x} + \frac{\sum_{j \in GP_{u,x}} (R_{j,x} - R_j) \times \mathrm{sim}(i,j)}{\sum_{j \in GP_{u,x}} |\mathrm{sim}(i,j)|} \tag{6}$$

R_j denotes the average value of user j satisfaction, and $R_{j,x}$ denotes the satisfaction value of user j about book x. $P_{i,x}$ is the predict value of user i about book x.

IV. DEVELOPMENT OF THE APPLICATION AND TESTS FOR PROVIDING A RECOMMENDATION

This section shows the results of the application developed based on the design in Section III and describes the tests used to check the provision of recommendations using the suggested application.

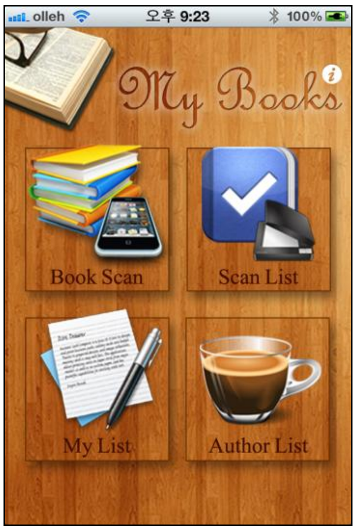
A. Book Management Application

When a user first runs the application, he/she needs to enter his/her name in the notification window. This user name creates the user evaluation table, as described in Section IIIB, which cannot be changed, so a section for the user to change his/her name, or to look at it, is not created. When the user registration is complete, the initial screen of the application will be displayed, as shown in Fig. 9(a). In order for the user to enter the book information, the Book Scan menu needs to be selected to scan the barcode or the QR code of the book. When the Book Scan menu is selected, the application displays the embedded iPhone camera and activates the Zbar library, which allows scanning to take place in real time. Fig. 9(b) shows the screen with a scanned book titled “Computer Vision and Fuzzy-Neural Systems.” As it is impossible to gain a book summary from the Naver API, the application acquires results from the Google API and displays them. When “Web” at the center right of the screen, shown in Fig. 9(b), is touched, the page details of the book available from Google can be reviewed.

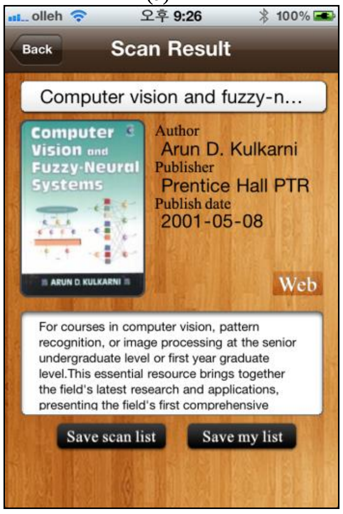
The user can store the search results in Scan List or My List. In Scan List, the books are scanned and stored, but not purchased; these are books you would like to review at a library or a bookstore at a later date. In My List, purchased books are sorted out and your thoughts and comments are entered; this is where you evaluate the book by assigning it points. Fig. 10(a) shows a list of books stored in My List. Books are listed so that the most recently registered books are on the top of the list. The user can select a book from the list of registered books and see its details. After reading a book, the user can leave comments and evaluation points. Fig. 10(b) shows a screen displaying the comments and points given by a user. When the evaluation by the user is complete, the evaluation point is transferred to the server and the evaluated book is recommended, based on the given value, to other application users Fig. 11.

B. *Usability Test and Recommendation Test Using the Suggested Application*

We tested inputting the book information in the proposed application with 10 books. The total number of subjects was 20, and one group consisting of 10 among 20 recorded the book information for the proposed application by hand. The other group consisted of the other 10, who inputted the book data in the proposed application using Barcodes or by detecting QR codes. Table IV shows the results of the book information input test.



(a)



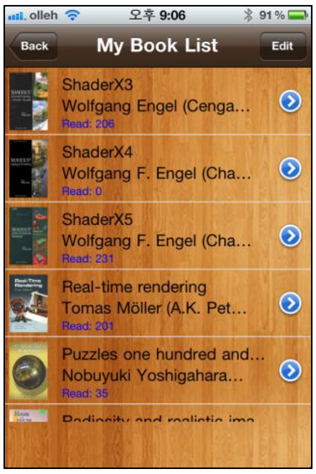
Computer vision and fuzzy-n...

Author
Arun D. Kulkarni
Publisher
Prentice Hall PTR
Publish date
2001-05-08

For courses in computer vision, pattern recognition, or image processing at the senior undergraduate level or first year graduate level.This essential resource brings together the field's latest research and applications, presenting the field's first comprehensive

(b)

Fig. 9. Each scene of the book management application; (a) First main scene, (b) Scene of the search results by the Book Scan menu.



ShaderX3
Wolfgang Engel (Cenga...
Read: 206

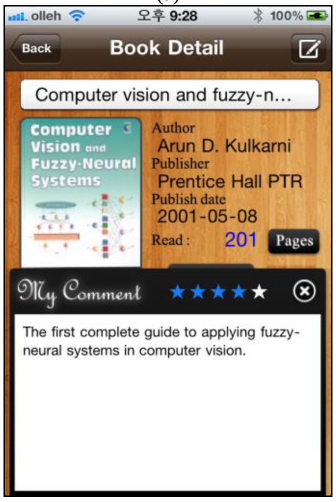
ShaderX4
Wolfgang F. Engel (Cha...
Read: 0

ShaderX5
Wolfgang F. Engel (Cha...
Read: 231

Real-time rendering
Tomas Möller (A.K. Pet...
Read: 201

Puzzles one hundred and...
Nobuyuki Yoshigahara...
Read: 35

(a)



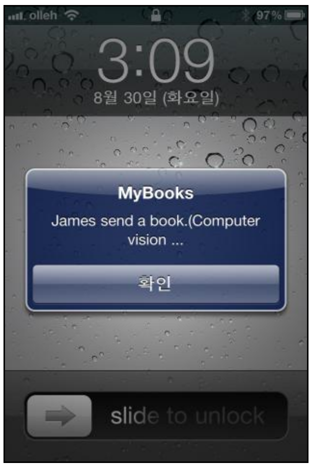
Computer vision and fuzzy-n...

Author
Arun D. Kulkarni
Publisher
Prentice Hall PTR
Publish date
2001-05-08
Read : 201

The first complete guide to applying fuzzy-neural systems in computer vision.

(b)

Fig. 10. The others scenes of the book management application; (a) My list scene, (b) User memo and evaluation point scene.



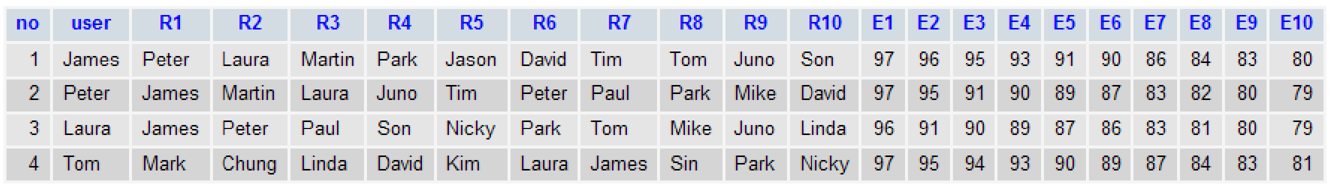
MyBooks
James send a book.(Computer vision ...

Fig. 11. Scene of a book recommended to other users.

TABLE IV: THE RESULT OF THE BOOK INFORMATION INPUT TEST

Input type	Average input time per book	Average user satisfaction
By hand	43.1 s	26 %
Barcode or QR codes detection	4.7 s	90 %

The input time was checked by the time required to input all the data, which included four pieces of information, such as book title, cover image, publisher, and book author. Then user satisfaction was investigated with a survey divided into five values: very uncomfortable (20), uncomfortable (40), normal (60), comfortable (80), and very comfortable (100) with the usability of the type of input. The input time using barcode or QR code detection was about 38 seconds faster than the time for hand-operated input. In addition, the user satisfaction using barcode or QR code detection was about 64% better than by hand. Therefore, the proposed application is very effective for managing books using an iPhone.



no	user	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
1	James	Peter	Laura	Martin	Park	Jason	David	Tim	Tom	Juno	Son	97	96	95	93	91	90	86	84	83	80
2	Peter	James	Martin	Laura	Juno	Tim	Peter	Paul	Park	Mike	David	97	95	91	90	89	87	83	82	80	79
3	Laura	James	Peter	Paul	Son	Nicky	Park	Tom	Mike	Juno	Linda	96	91	90	89	87	86	83	81	80	79
4	Tom	Mark	Chung	Linda	David	Kim	Laura	James	Sin	Park	Nicky	97	95	94	93	90	89	87	84	83	81

Fig. 12. Example of similarity among user.

Next, the Linux OS, a Pentium 4 3.0Ghz CPU, a 2G memory, and a MySQL 5.0.5.1 database were used as environments for testing the recommendation process. The total number of subjects was 300, and twenty books (a total of 200 books) for each of the 10 genres (Children's stories, Detective Fiction/Mysteries, Fantasy, Historical Fiction, Humor, Literature, Poetry, Romance, Science Fiction, Self-help) were selected as the target books. One hundred books were used for learning purposes and the remaining 100 books were used for testing purposes.

The subjects evaluated 100 academic books individually. In reality, it was impossible for all subjects to use the application to evaluate all 100 books, so the evaluation was handled via a website. As in the application, points could be given ranging from 1 to 5 through the website, while ensuring that the subjects evaluated the 100 books without missing any. When the subjects finished the evaluation of the academic books, the individual book evaluation data created by each subject contained 100 values. A user-based collaborative filtering method was used to calculate the similarities among the individual subjects. Fig. 12 shows an example of similarity values calculated for each user after the learning process was completed.

Similarity is a value expressed in numerical characters reflecting the degree of similarity between two users in terms of their evaluation of a book. James, shown in Fig. 12, has the highest similarity to Peter and vice versa. In the next test, we measured the results of the recommendation obtained when each subject selected one book to evaluate. For example, when James, No. 1, selects a book from the 100 books prepared for the test, and gives a score by checking, the book will be recommended to Peter and Laura who have similar preferences to James. Then Peter and Laura, for whom the recommendation is provided, would each evaluate the book. The success of the predicted recommendation value is judged through comparison with the value calculated in Eq. (4).

In the test, the recommendation was sent to the top 10 subjects with highest similarity. When a user received and stored a recommendation, an ensuing recommendation could take place; therefore, for the purpose of measuring a result in this test, the system was manipulated so that an ensuing recommendation process would not take place. When it came to a case in which a recommendation was provided but an evaluation was not made, the related data were excluded from being used to judge whether the predicted recommendation value was successful. At the first, we tested 100 subjects who had been selected randomly from the 300 subjects. In this test, we counted the recommend times, the evaluation times, and the correct evaluation times of the existing algorithm and the proposed algorithm. We used user-based collaborative filtering with the existing algorithm, which we modified and used for the proposed recommendation algorithm. Table V shows the results of the recommendation test.

TABLE V: FIRST RECOMMENDATION TEST RESULTS

Method type	Existing method	Proposed method
Recommendations	100	852
Evaluations	81	799
Evaluation result	67 (82.71 %)	686 (85.86 %)
RMSE	0.142	0.147

TABLEVI: SECOND RECOMMENDATION TEST RESULTS

Method type	Existing method	Proposed method
Recommendations	200	1739
Evaluations	159	1579
Evaluation result	136 (85.53 %)	1370 (86.76 %)
RMSE	0.148	0.143

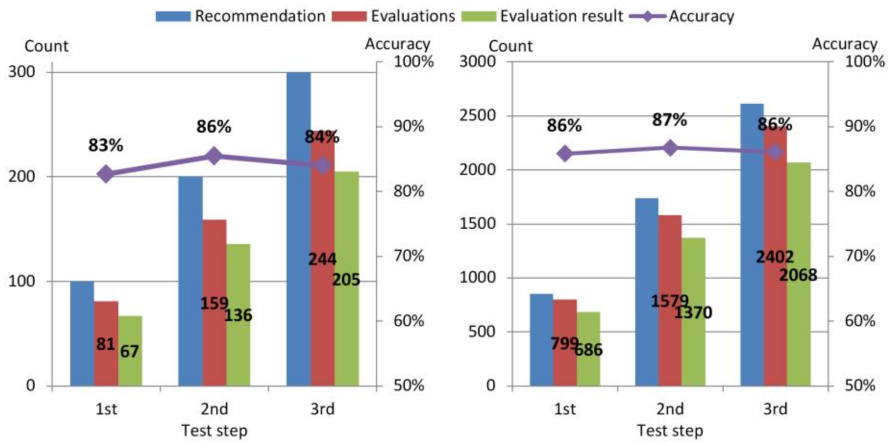
Determining whether the predicted evaluation values and the actual evaluation values were accurate was based on the difference of ±0.5 between the two values. For example, if the predicted evaluation point was 3.76 or 4.02, and the actual evaluation point given by the user was 4, the prediction could be deemed accurate. For the existing algorithm, if a user evaluated his or her preference for a book, the algorithm recommended to the user only one other book according to the user's preference. A user evaluated about the recommended book again 81 times, and the number of correct predictions was 67. The accuracy of the predicted evaluation value was about 82%. Considering that there were 100 subjects in the proposed method test, and each subject recommends 10 times, the total number of recommendations made should amount to 1,000. However, the reason the total number of recommendations was just 852 was that, if a user selected a book randomly, recommendations were made to other users in a duplicate manner, depending on the subjects, leading to the situation that no more recommendations could be made to a subject when the subject had already received a recommendation and provided an evaluation. In addition, if the evaluation points were low, no recommendation was made. The number of evaluations made corresponding to the recommendations reached a total of 799, indicating that the answer rate was about 93%. The accuracy of the predicted evaluation value was about 85%. Therefore, from the test, we observed that the proposed method leads to more recommendations and evaluations than the existing method, and the accuracy of the proposed method was similar to that of the existing method. To analyze the statistical significance, for the second test, we tested 200 subjects by adding 100 subjects and Table VI shows the results.

TABLE VII: THIRD RECOMMENDATION TEST RESULTS

Method type	Existing method	Proposed method
Recommendations	300	2615
Evaluations	244	2402
Evaluation result	205 (84.02 %)	2068 (86.09 %)
RMSE	0.145	0.142

The recommendations in the second test were twice as high as those for the first test for the existing algorithm and the proposed algorithm. The answer rate of the existing method was about 79%, and the accuracy of the predicted evaluation value was about 85%. The answer rate of the proposed method was about 90%, and the accuracy of the predicted evaluation value was about 86%. Those rates were similar to the first test results. For the third test, we tested 300 subjects by adding the other 100 subjects. Table VII shows the results.

In the third test, we saw that the recommendations and evaluations increased by the same rate according to the increase in the number of subjects, and the accuracy of the predicted evaluation value was similar to the first and second test results, too. Then, we created a graph of the first, second, and third test results, as shown in Fig. 13. The recommendations and evaluations in Fig. 13(a) and Fig. 13(b) increased as the number of subjects increased. The graph shows that the proposed method had more recommendations and evaluations than the existing method. Nevertheless, the existing method and the proposed method were similar in the accuracy of the predicted evaluation value: the existing method was 83%, 86%, and 84%, and the proposed method was 86%, 87%, and 86%.



(a) (b)

Fig. 13. Result comparison with the existing method and the proposed method; (a) The test result of the existing method, (b) The test result of the proposed method.

Next, we tested the accuracy of the predicted evaluation value for continuing recommendations. Previously, the first, second, and third tests used only one recommendation according to the user's preference input. However, the proposed algorithm could recommend continuously recommend the book to other persons using the user's evaluation. Therefore, we tested the change of the degree of the continuous recommendation. We randomly selected 100 subjects among the 300 subjects and changed the degree of the continuous recommendation from one to three. At this time, the continuous recommendation occurred when the predicted evaluation value and the user's evaluation value were equal and did not occur when the user had already evaluated a book that would be recommended or the book had been recommended by another person. Table VIII shows the results of the continuous recommendation test.

TABLE VIII: CONTINUOUS RECOMMENDATION TEST RESULTS

The degree of Recommendation	First	Second	Third
Recommendations	852	1359	1765
Evaluations	799	1121	1415
Evaluation result	686 (85.86 %)	961 (85.73%)	1214 (85.79%)
RMSE	0.147	0.146	0.147

The second recommendation led to 1,359 recommendations automatically by the first recommendation. If we calculated the number of recommendations, it has to be 10,000 (100 subjects × 10 recommendations × 10 recommendations). However, the number of second recommendations decreased, because they were due to 686 users' evaluations that predicted the value was accurate at the first recommendation and did not occur when the user had already evaluated a book that would be recommended or had been recommended by another person. The third recommendation led to 1,765 recommendations due to 961 users' evaluations at the second recommendation. This meant that if the degree of the proposed algorithm was three, the proposed method could recommend about 17 times, while the existing algorithm could recommend only one time. Nevertheless, the accuracy of the predicted evaluation value was similar. Consequently, the proposed method can automatically recommend many books to many individuals with one user's evaluation, and the accuracy of the predicted evaluation value is the same as that of the existing method.

V. CONCLUSION

The application suggested in this paper, not only helps a user see detailed information about a book by using an iPhone camera to recognize a barcode or QR code, but it can also sort out users' own books, their thoughts after reading them, their comments, and their evaluations of the books. The application also can provide a new service that recommends books to other users based on user evaluations. The proposed method is very different from existing website recommendation methods and book applications. Because it uses Push Notification on mobile devices, this novel recommendation method will be able to actively recommend books to users. Such a method can be useful not only in this suggested application, but also for Social Network Service-based merchandizing coupons as well as digital and audio products.

In future research, we plan to test a recommendation method that uses an item-based collaborative filtering method, as opposed to a user-based collaborative filtering method, as in this study. Our research will also cover new contents that can combine the camera's Barcode and QR code recognition technologies, GPS, and AR-based navigation technology.

REFERENCES

[1] C.T. Tan and D. Soh, "Augmented reality games: A review," *Proceedings of GAMEON-ARABIA, EUROSIS,* 2010.

[2] O. Oda and S. Feiner, "Rolling and shooting: Two augmented reality games," in *Proc. 28th International Conference on Extended Abstracts on Human in Computing Systems,* Atlanta, GA, USA, 2010, pp. 3041-3044.

[3] Layer. [Online]. Available: http://www.layar.com

[4] Ovjet. [Online]. Available: http://ovjet.com

[5] E. Ohbuchi, H. Hanaizumi and L.A. Hock, "Barcode readers using the camera device in mobile phones," in *Proc. International Conference on Cyberworlds,* Kanagawa, Japan, 2004, pp. 260-265.

[6] R. Muniz, L. Junco and A. Otero, "A robust software barcode reader using the Hough transform," in *Proc. International Conference on Information Intelligence and Systems,* Rockville, Maryland, USA, 1999, pp. 313-319.

[7] ZBar, bar code reade. [Online]. Available: http://zbar.sourceforge.net

[8] Zxing, Multi-format 1D/2D barcode image processing library. [Online]. Available: Available: http://code.google.com/p/zxing

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[10] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annual ACM Workshop on COLT,* Pittsburgh, PA, USA, 1992, pp. 144-152.

[11] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," *Advances in Neural Information Processing Systems,* vol. 12, pp. 547-553, 2013.

[12] K. B. Duan and K. S. Sathiya, "Which is the best multiclass SVM method? An empirical study," in *Proc. 6th International Workshop on Multiple Classifier Systems,* Seaside, CA, USA, 2005, pp. 278-285.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proc. 2nd ACM Conference on Electronic Commerce,* Minneapolis, Minnesota, USA, 2000, pp. 158-167.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-A case study," *Minnesota Univ Minneapolis Dept of Computer Science,* New York, USA, 2000.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. the 10th International Conference on World Wide Web,* Hong Kong, China, 2000, pp. 285-295.

[16] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *Internet Computting,* vol. 7, no. 1, pp. 76-80, 2003.

[17] M. Frassica, "For ebook devotees, reading is a whole new experience," *The Louisville Courier Journal*.

[18] C.C. Miller, "Need advice on what to read? Ask the internet," *The New York Times.*

M. B. Chung received the MS from the Department of Media and the PhD from Soongsil University, in 2004 and 2010. He worked on BK21 project as a post-doctoral follow at the Soongsil University at Seoul, in 2010 and 2011. From 2012 to 2014, he was with the School of Information and Communication Engineering, Sungkyunkwan University (Korea). Since 2015, he is now an assistant professor of Division of Computer Engineering, Sungkyul University (Korea). His research interests include copyright protection technique, mobile computing, mobile software development, audio signal processing, and recommendation system.