dpSmart: a Flexible Group based Recommendation Framework for Digital Repository Systems

Boyuan Guan, Liting Hu, Pinchao Liu, Hailu Xu, Zhaohui Fu, Qingyang Wang[†]

Florida International University, [†]Louisiana State University
Email:{bguan, lhu, pliu002, hxu017, fujen}@fiu.edu, qywang@csc.lsu.edu

***Abstract*—Digital Repository Systems have been used in most modern digital library platforms. Even so, Digital Repository Systems often suffer from problems such as low discoverability, poor usability, and high drop-off visit rates. With these problems, the majority of the content in the digital library platforms may not be exposed to end users, while at the same time, users are desperately looking for something which may not be returned from the platforms. The recommendation systems for digital libraries were proposed to solve these problems. However, most recommendation systems have been implemented by directly adopting one specific type of recommender like Collaborative-Filtering (CF), Content-Based Filtering (CBF), Stereotyping, or hybrid recommenders. As such, they are either (1) not able to accommodate the variation of the user groups, (2) require too much labor, or (3) require intensive computational complexity.**

In this paper, we design and implement a new recommendation system framework for Digital Repository Systems, named `dpSmart`, which allows multiple recommenders to work collaboratively on the same platform. In the proposed system, a user-group based recommendation strategy is applied to accommodate the requirements from the different types of users. A user recognition model is built, which can avoid the intensive labor of the stereotyping recommender. We implement the system prototype as a sub-system of the FIU library site (http://dpanther.fiu.edu) and evaluate it on January 2019 and February 2019. During this time, the Page Views have increased from 8,502 to 10,916 and 10,942 to 12,314 respectively, compared to 2018, demonstrating the effectiveness of our proposed system.

***Keywords*-group based recommendation, stereotyping recommendation, subspace clustering, automatic log mining, automatic user recognition**

I. Introduction

Digital Repository Systems have been used in most modern digital library platforms, which are used to store, archive, and index all of the digital assets in the library as well as to serve those assets to clients and users of libraries over the Internet. Digital Repository Systems have three long existing issues including (1) the low discoverability of the content; (2) lack of assistance to explore the system; and (3) high drop off visits. `dPanther` [1] (http://dpanther.fiu.edu) is the Digital Repository Systems developed and implemented by Florida International University (FIU) to host the digital assets including digital archives and digital born content and publications. The web server log of `dPanther` in 2019 shows that only 24.9% of the the content has been discovered, 62.1% intended search ended with a page review, and 58.8% resulted in drop off visits.

In the past decade, many studies have been conducted to remedy these issues. The most famous one is the recommendation systems [2] [3] [4] [5] [6]. Recommendation systems, as a subclass of information filtering systems, are typically used to produce a list of recommended content from the hosting system by predicting the ”rating” or ”preference” of the users. They improve the content discoverability, provide guidance, and retain more users for the hosting system. However, most recommendation systems have been implemented by directly adopting one specific type of recommender like Collaborative-Filtering (CF) [7], Content-Based Filtering (CBF) [8], Stereotyping [9], or hybrid recommenders [10]. As such, they are either (1) not able to accommodate the variation of the user groups, (2) require too much labor, or (3) require intensive computational complexity.

In this paper, we design and implement a new recommendation system framework for Digital Repository Systems, named `dpSmart`, which allows multiple recommenders to work collaboratively on the same platform, including the Content Based filtering (CBF) [8], Collaborative Filtering (CF) [7], Global Relevance (GR), Query Suggestion (QS)/Term Suggestion (TS) [11] and Location Based Filtering [12] models. The users are grouped by a stereotyping model and different users are served by different recommenders. The user vectors are built based on the user behaviors extracted from the log data. By optimizing the algorithm with multi-processes programming fashion and maximizing the server capacity, the model retraining time can be reduced significantly.

We implement the system prototype as a sub-system of the FIU library site (http://dpanther.fiu.edu) and evaluate it on January 2019 and February 2019. During this time, The Page Views have increased from 8,502 to 10,916 and from 10,942 to 12,314 respectively, compared to Page Views in 2018.

The technical contributions of this paper are as follows:

- We implement a stereotype based recommendation system that can adapt to multiple different recommenders.
- The proposed system avoids the intensive labor and automates the process from the log data extraction to model training.
- By using a stereotyping recommender, we avoid the need of personally identifiable information user data and we reduce the noise recommendation.
- By implementing multi-process programming, the model re-training time can be significantly reduced.

The rest of this paper is organized as follows: Section II summarizes the background and related works in recommendation system research within the digital library domain; Section III presents the design and functional components of dpSmart; The evaluation results using real-world logs are shown in Section IV, and finally we conclude this work in Section V.

II. Related Work

A. Recommendation System in Digital Library

Due to the uncertainty of the content, the variation of patrons, and the unpredictable user interactions with the library system, content based recommendation systems have dominated in digital library domain. C.Musto et al. [13] proposed a Content-Based Recommender System for Digital Library for Cultural Heritage. S.Philip et al. [14] proposed a Content-Based approach in paper recommendation systems for a digital library. However, the problem of a Content-Based recommender lies in that this type of systems are not able to accommodate the variation of the clients and the user behaviors. Another type of recommendation system is customization and personalization for the niche domain. A.F.Smeaton et al. [15] proposed a method to personalize recommendation system for digital library platform. However, this type of system is very hard to generalize. In order to generalize and further potentially automatize the customization and personalization process, the user recolonization [16] was proposed.

In the Research-Paper Recommendation Systems survey [11], 200 publications have been reviewed and 62 different methodologies have been proposed. Among those proposed methods, content based filtering is in the top rank, and 55% of the methods are either based on this or leveraged it. Collaborative Filtering ranked No.2 with 18% of methods used. Graph-based recommender ranked No.3 with in 16% of methods used. Other recommenders like stereotyping, item-centric, and hybrid recommendations are mentioned in 11 of the publications.

B. User-group clustering for User Reorganization

Based on the designed framework, user modeling and user group recognition are key functions to build a flexible recommendation system. Previous publications regarding user modeling have also been reviewed. A recent dissertation from the University of Cornell [17] has been published to introduce Mind Mapping based user modeling for research paper recommendation systems. Although the result is very promising, the approach is heavily rely on an open-source JAVA application for managing PDF files, annotations, and references with mind maps called Decear and required labeled user data. Consider that lack of personally identifiable information, user behavior data is one of the toughest challenges for the recommendation system in Digital Library System hosted by public libraries, we switch the machine learning approach to an unsupervised learning methods to recognize the user group. A good example can be found in [18], which proposed a clustering based method for the web services recommendation system. Although the method proposed in [18] is in web services recommendation domain, it provides an solution for the problem of lack of labeled user behavior information. A more advanced subspace clustering based method [19] was proposed in 2014 for user group identification. This subspace based clustering method has two main features: prune non-promising features for subspace extension dimensions and fault tolerance.

C. Stereotyping Recommendation

Stereotyping has a long history for user modeling. Rich [20] in 1979 initially described how to build user models by using stereotyping. In 2007, additional research was conducted by Guy [20] to implement Stereotyping Recommendation method in media systems. The performance of stereotyping Recommendation, by combining content-based filtering and collaborative filtering, out-performs the regular Adhoc hybrid recommendation system performance. As such, stereotyping recommendation has been considered as the recommender with the best performance. However the drawbacks of Stereotyping recommenders, such as pigeonholing users and labor intensiveness, have significantly limited the usage of stereotyping recommender. These two drawbacks are directly caused by the traditional way of manual processing process to identify user groups.

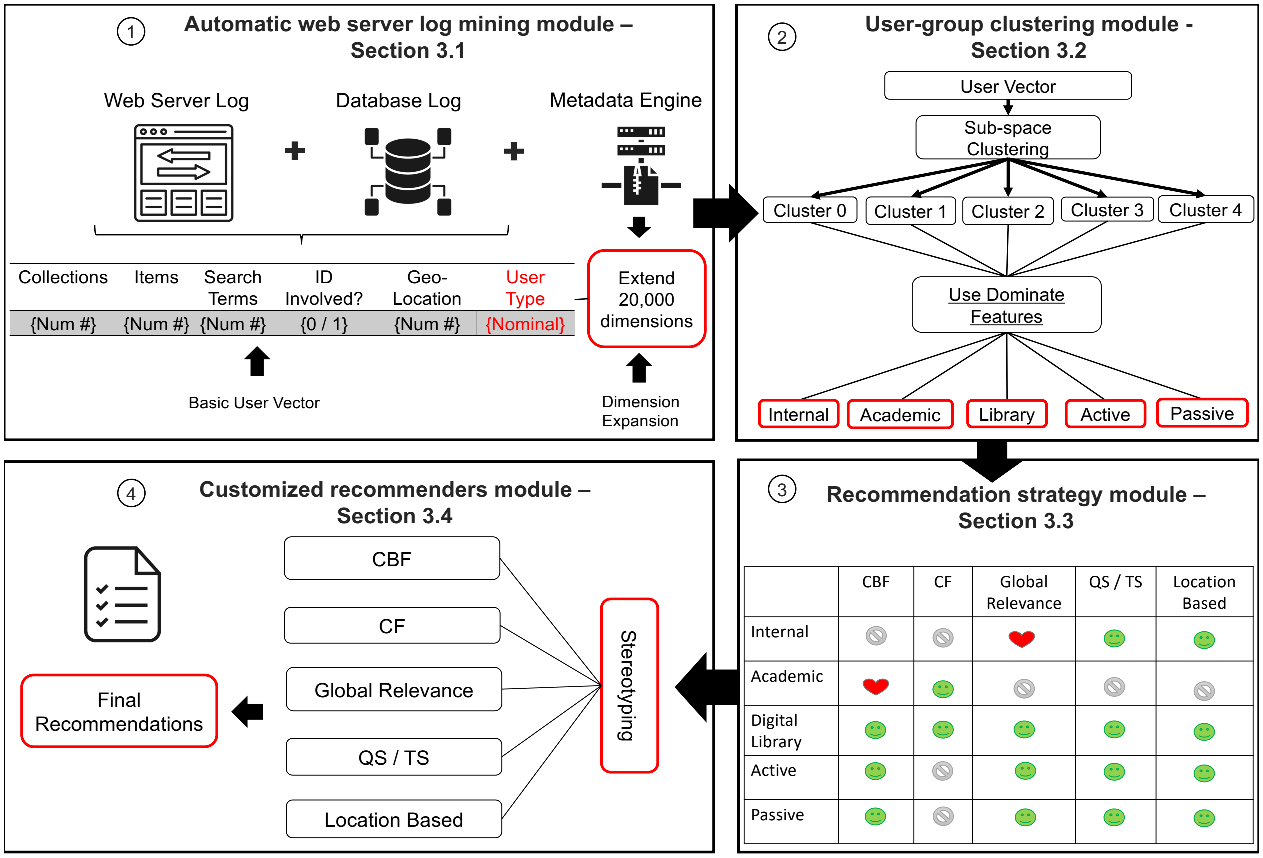
III. System Design

We design and implement a new recommendation system framework for Digital Repository Systems, named dpSmart, which allows multiple recommenders to work collaboratively on the same platform. dpSmart has four major components: 1) the **automatic web server log mining module** to generate user behavior data, 2) the **user-group clustering module** for automatic user group recognition, 3) the **recommendation strategy module** for stereotyping filter based on different user group, and 4) the **customized recommenders module** to implement into dPanther Digital Repository Systems.

Figure 1 shows the overall design of dpSmart. The workflow starts from gathering and processing log data from an IIS web server. The automatic web server log mining module is responsible for processing the data to generate the user vectors. Once the user vectors are generated, the User-group clustering module runs the cluster and generates the group of the users. The recommendation strategy module then maps the generated cluster to the corresponding user group and establishes the stereotyping filter. Based on the stereotyping filter, the customized recommender modules can provide customized recommendations from embedded recommenders. The detailed design and implementation of each module are discussed below.

A. Automatic Web Server Log Mining Module

We utilize two types of resource (i.e., the web server log and the database log) to legally analyze user behaviors. Both types of log data are naturally generated whenever a web-based



	CBF	CF	Global Relevance	QS / TS	Location Based
Internal	⊘	⊘	♥	☺	☺
Academic	♥	☺	⊘	⊘	⊘
Digital Library	☺	☺	☺	☺	☺
Active	☺	⊘	☺	☺	☺
Passive	☺	⊘	☺	☺	☺

Fig. 1: Overall framework of `dpSmart`. `dpSmart` consists of four components: the automatic web server log mining module, the user-group clustering module, the recommendation strategy module, and the customized recommenders module. The automatic web server log mining module first processes the log data into user vectors. Secondly, the user-group clustering module runs the subspace clustering and maps the clusters into user groups by using dominate features. Thirdly, the recommendation strategy module defines the group preference against different recommenders for the stereotyping filter. Finally, the customized recommenders module generates the final recommendations from specific recommenders.

application is published and very limited user information is stored. We generate datasets through the following data preparation and processing methods, which can be used directly to build user vectors.

The IIS logs are the primary resource for our system because they contain some key information, including but not limited to:

- *The client IP address*
- *The time of the activity*
- *The region of where the user comes from*
- *The type of activities the was interacting with the system*

This information is used as dimensions for building the user vectors. The original IIS log has the schema as shown in Table I. Note that we assume that each IP on a particular date represents a single user, so a user vector is generated from the combination of IP address and the date from IIS logs, IP-date log.

We first process the raw IIS logs and make them more useful by filtering out logs from bots, local access, scripts, style sheets, and other file loading. We then rebuild the logs by combining the IP and date to create basic user logs that contain the IP activity of the website on a single date. Next, we convert each IP-date log into an IP-date user vector, and each user vector contains all the numeric item numbers they accessed using the number of transactions. The basic user vector that we can build directly from the IIS log is a vector containing two dimensions which are the number of entries accessed in the database and the number of visits from this IP address.

Secondly, we start expand the dimensions. Since the IP address contains the location information, after geo-coding the IP address, two additional dimensions (latitude and longitude) can be added to the vector to generate a fourth dimensional vector. This base vector can provide information about where a particular user came from, how often users access our system, and whether the user has successfully accessed a particular item in the system.

The dimensions can be expanded more by integrating the database query log. The query log also records the contents of the query as well as the IP address. Since the search action is a very strong indicator of the willingness of a particular user to find content in the system, the number of searches made by a particular user is added as an additional dimension of the user vector. In addition to the number of queries, the search content is also very useful. One of the key terms that conveys the clearest information is the system ID of the project in the database. Since the system ID is defined using a specific name convention, it can be identified from the regular expression. Users who use the system ID as a search term are likely to be internal users from our organization or academic users from close partner organizations. Therefore, the system ID is added to the user vector as another dimension. As shown in Figure 1 step 1, the generated six-dimensional user vector is used as the basic user vector. However, the six dimensions are not sufficient to build any reliable clustering results. Therefore, we must introduce the metadata of the accessed project to expand the dimension.

Last but not the least, the metadata of the record can provide a lot of more dimensions. One of the most unique features of a Digital Repository Systems or digital library system is that all records in the system have prepared metadata that is stored in the metadata engine. With this feature, we can immediately increase the user vector's dimensions to 50, 100 or more by linking user behavior to metadata. For example, when visiting a particular project, such as the Miami 1920 album, users may also be interested in other historical albums, or they may be interested in other projects by photographer Eva FitzGibbon Drummond, or they may be interested to items in the same collection. In this case, with Coral Gables Memory and Miami Metropolitan Archive, we need to carefully select the metadata tags because we want to collect enough data to fit the cluster to our predefined user groups, and we don't want to introduce too many dimensions. So, after building the basic user vector, we can further expand the dimension by introducing metadata information in the metadata engine.

B. User-group clustering Module

Since we pre-define the target user group and recommendation strategy, for any active users, as long as we can identify the user group to which the user belongs, we can dynamically provide appropriate recommendations and minimize noise recommendations.

Ideally, if there are fixed metrics that distinguish users in these groups, we can simply identify the user group based on the criteria. However, these user groups are overlapped and it is difficult to find a clear boundary between user behavior patterns among groups. as shown in Figure 2, we use several dominate features to separate the user group. For example, a very powerful indicator is the appearance of a system ID in a search term. Most likely, the system ID is for internal users only. However, some academic users also save the system-generated ID for quick access to their favorite projects. Another good example is location information. If the location information is internal to the library, then the access should be directed to the internal user group. However, many users of the academic group also work in the library. The identification of the user group can be achieved by using an indicator from the log data. As shown in Figure 2, a sample decision tree illustrates this point. Note that in order to use a tree based classifier, we have to find the tagged training data, which is not available in existing systems. Therefore, clustering methods are applied in our research.

- *Number of visits*
- *Search conducted*
- *Number of Items*
- *ID involved in search*
- *Location of the users*

In the `dpSmart` framework, a subspace clustering method [19] is used to generate the user cluster. Based on a survey [21] published in 2009 for the algorithms of subspace clustering are two major subspace algorithms: bottom-up versus top-down approaches. We choose a topical top-down algorithm, PROCLUS [22] for our framework, since it allows us to define the number of clusters generated for the data set. The software WEKA and a third party package, Opensubspace v3.31 are used to generate the user cluster.

C. Recommendation Strategy Module

Similar to other information management systems, Digital Repository Systems have predefined target user groups. The target user groups can be divided into five groups, namely the Library Internal Users, Digital Library Users, Local Academic Users, Active Web Users, and Passive Users. The detailed definitions for each group are listed below:

1) **Library Internal Users** - the users from our internal organization like the developers, the librarians, the system administrators, and metadata creators.
2) **Academic Users** - professional users who are familiar with Digital Repository Systems as well as the content inside the system. This user group mainly consists of scholars/professors, undergraduate/graduate students, researchers.
3) **Digital Library Users** - the users who are familiar with the structure and interface of Digital Repository Systems. This type of user can retrieve information from the system by following the metadata structure. They are normally not as focus on one specific topic.
4) **Active Web Users** - users who are interested in using the system to conduct the search and try to explore the content but have difficulty finding the right content or are not able to access any content.
5) **Passive Web Users** - users who only stop by the landing page without doing anything.

According to this grouping, we further develop a recommendation strategy as shown in Figure 1 Step 3. The recommendation preference is marked in different types of attitudes, red heart if preferred, smile face if they don't bother, and block sign if they dislike. For internal users who are metadata creators, digital assets owners, and system administrators, they don't want any type of content-based or collaborative filters. They are keen to Query Suggestions, Term suggestions, and Location based suggestions. Global Relevance, which makes recommendations based on the system status like the popular

TABLE I: Formats of W3C Extended Log File.

Log Field	Description
date	Date at which transaction completed, field has type <date>
time	Date at which transaction completed, field has type <date>
s-ip	Server IP address and port, field has type <address>
cs-method	Client to server method
cs-uri-stem	Client to server stem portion alone of Uniform Resource Identifier (URI, omitting query), field has type <uri>
cs-uri-query	Client to server Query portion alone of URI, field has type <uri>
s-port	Server port
cs-username	Client to server user name
c-ip	Client IP address and port, field has type <address>
cs(User-Agent)	Client to server (user-agent)
sc-status	Server to client status code, field has type <integer>
sc-substatus	Server to client substatus code, field has type <integer>
sc-win32-status	Server to client win32 status code, field has type <integer>
time-taken	Time taken for transaction to complete in seconds, field has type <fixed>

items or new added items, is the preferred recommendation for Internal Users.

Academic users need a completely different recommendation strategy than internal users. These users are very professional and well-trained in Digital Repository Systems or Digital Library Systems, so they really reject irrelevant information. Therefore, global relevance, QS/TS, and Location Based recommender should be blocked for them. A Content based recommender is the one they preferred and Collaborative filtering is not considered as noise recommendations. The Digital Library Users are familiar with digital collection systems but do not have a background knowledge of the content in the system, so it is difficult for them to come up with a clear research area. Since these are the main target users, our recommendations are hard to judge for this user group. For these two types of users, we just hide collaborative filtering, since users in the two groups do not necessarily have many similarities in preferred content.

D. *Customized Recommenders Module*

As mentioned before, we utilize the Content-based Filtering (CBF), Collaborative Filtering (CF), Global Reference (GR), Query Suggestion (QS)/Term Suggestion (TS), and Location based Recommendation in the module. These five recommenders are implemented separately and coordinated according to the results of the stereotyping filter. The details of the custom design are shown as below.

1) Content-based Filtering (CBF): Based on the record curation process, there are two suitable models to create the item-similarity model: (a) Jaccard [23], which is good at depicting similarities between two sets and (b) Directed Graph [24], which is good at depicting the similarity of two vectors. Zhou [25] presented the challenge of eliminating noise and sparsity using Graph-based method, which is mainly caused by the metadata type and content. The same problem may be more challenging in dPanther. A feature of dPanther as a Digital Repository Systems is that it provides a large collection of research articles, research papers, engineer studies/reports, newspapers, government documents, music scores and many other types of historical documents. From this perspective, the Digital Repository Systems actually provides services for multiple metadata types, which may be completely different. As such, it is very difficult to build directed graphs for records because there are many missing links when moving across different metadata types. Therefore, we choos to establish a similarity model based on Jaccard similarity. Record vectors in the dPanther system can be constructed by directly using metadata tags and values. By applying Jaccard similarity, the similarity between items in dPanther can be calculated as follows:

$$S_(a,b) = \frac{|A_a \cap B_b|}{|A_a \cup B_b|} \tag{1}$$

- Where *S* is the Jaccard Similarity
- *a, b* are any two items in dPanther system
- *A, B* are the set of metadata fields and value of item *a* and *b*

However, even if this provides default similarity between items in the dPanther repository, the result does not accurately represent the actual relationship between the items, because in the current method, all metadata fields in the collection are equally important, but actually, the metadata fields within the project set have an explicit priority. For example, subject keywords in a set actually play a major role in calculating similarity. If two items share one subject keyword, its impact on the similarity calculation should be greater than the impact of sharing one publisher. Therefore, we modified the original Jaccard method to accept weight data set. Thus, we have:

$$SW_(a,b) = \frac{|\sum_{n=1}^{\infty}((A_n \cap B_n) * W_n)|}{|\sum_{n=1}^{\infty}((A_n \cup B_n) * W_n)|} \tag{2}$$

- Where *SW* is the weighted Jaccard Similarity
- *a, b* are any two items in dPanther system
- *A,B* are any pair metadata field from n selection metadata data fields between item a and b
- *w* is the weight assigned for a specific pair of one metadata field

By implementing the weighted Jaccard Similarity, we have build a reliable measurement system for the items' content. For each individual item, we can have a corresponding ranked similar item list. Then, the next major step is to design another

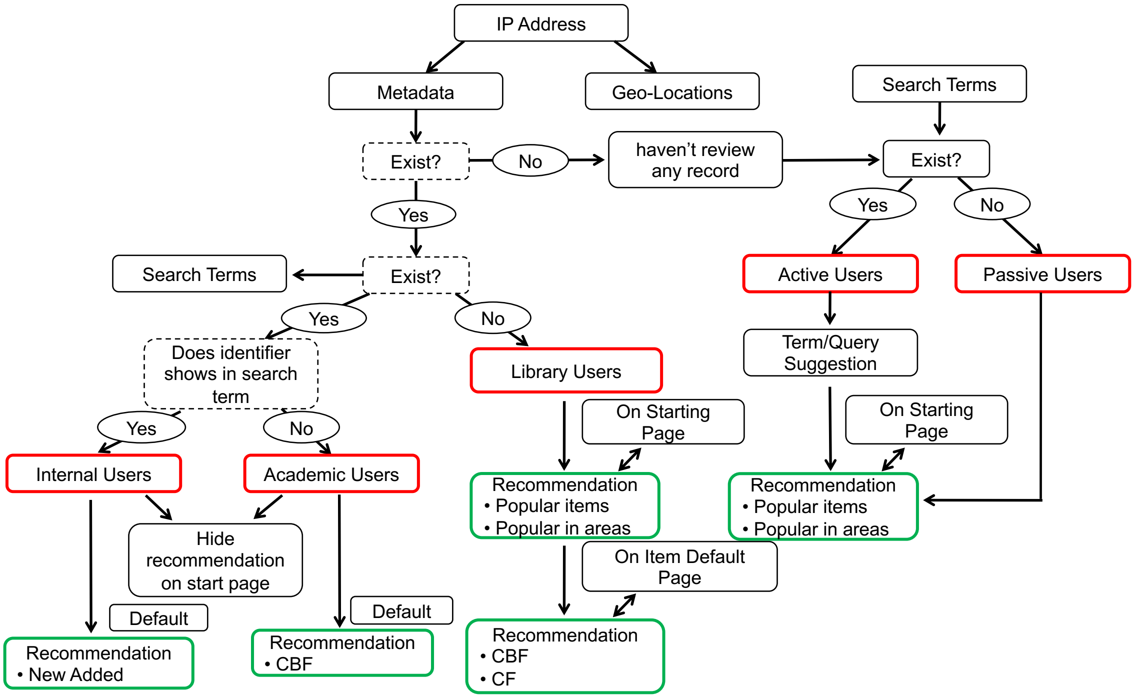


Fig. 2: Sample Decision Tree to demonstrate how to identify the user groups by using the dominate features. Three dominated features, Metadata, Search Terms, and ID-involved are used for the user group recognition. The green box demonstrates the recommenders selection for the different user group.

method to generate measurement for items from user-similarity prospective.

2) Collaborative Filtering (CF): For public library systems, there are two main obstacles to track user information: (a) The privacy policy limits user information and (b)lacks rating motivation. The process of accessing digital collections is considered part of user privacy (*https://library.fiu.edu/using-the-library/patron-privacy-policy*). Even if the system has a user account function, the users account activities are not allowed to be used for this type of research. Besides, unlike Amazon or Ebay, the preferences for the items are very individually based, thus it is hard to find a motivation for the users to provide rating for the digital collections they have viewed in the system. What is more, even if the rating is available, it may not necessarily useful across the users groups. Therefore, in the design of the `dpSmart` recommender, the only available resource to use is the web log which contains the IP addresses of the visits. In our research, we assume one IP address in the same day represents a specific user. From the IP address, we can get an idea of the region where the users came from, time they visited the system, topics they are interested, and then associate the items with these information.

By considering the information from the user visits, we can take advantage of implicit feedback alternatively. Implicit feedback strongly relates with the behaviors of users. There are two ways to get implicit feedback. One way is based on whether users check on the content of items in history. If so, we assume that users may like items and rate 1 as scores, otherwise rate 0. While the other way is to count the number of visits for the same item from the one specific user. Since one ”hit” from the web server log does not necessary mean one visit from the client side, it may also indicate the user downloading the picture from the item detail page, or clicking the link from the client detail page, or playing the video or audio from the detail page. Therefore, this count does not only indicate the actual visits from the user but also provides the interest level from the user.

By considering the two implicit feedbacks, we can get the scores for user-item similarity. The second step is to calculate the similarities between items. Assuming we have two items i and j, if we get scores based on whether users check on contents, then the similarities between i and j can be calculated using standard Cosine Similarity [26]. The basic Cosine Similarity is define as below:

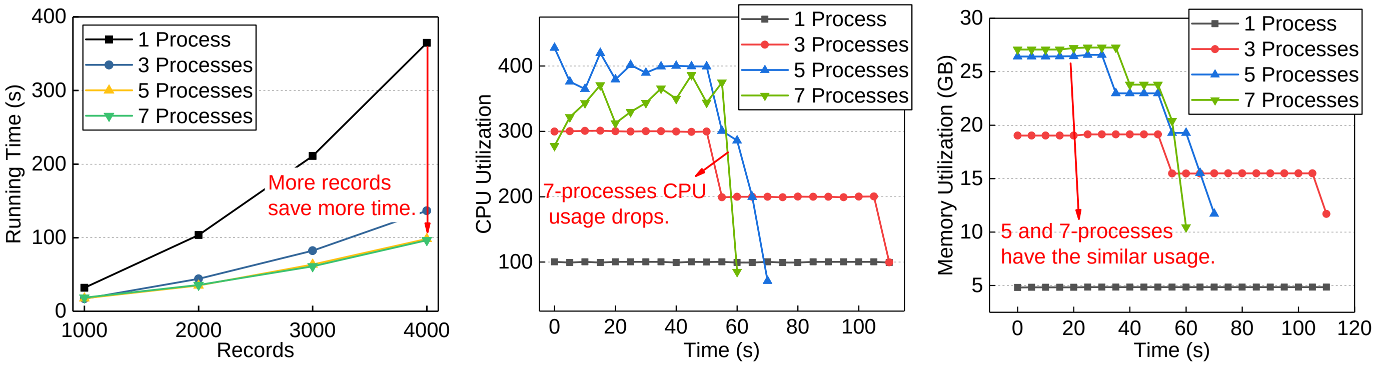
$$S_{(i,j)} = \frac{|N_{(i)} \cap N_{(j)}|}{\sqrt[2]{|N_{(i)}||N_{(j)}|}} \tag{3}$$

- Where S is the cosine-based similarity
- *N(i)* and *N(j)* mean the total number of users who rate 1 for item i and item *j*, respectively

By considering the counting of number of visits for the items, we come up with the adjusted Cosine Similarity as:

$$S_{(i,j)} = \frac{\sum_{u\in U}(R_{(u,i)} - \overline{R}_u)(R_{(u,j)} - \overline{R}_u)}{\sqrt[2]{\sum_{u\in U}((R_{(u,i)} - \overline{R}_u)^2)}\sqrt[2]{\sum_{u\in U}((R_{(u,j)} - \overline{R}_u)^2)}} \tag{4}$$

- Where S is the cosine-based similarity



(a) The Running Time Comparison.

(b) The CPU Usage.

(c) The Memory Usage.

Fig. 3: The running time latency, the CPU usage for 4,000 sample records, and the memory usage for 4,000 sample records.

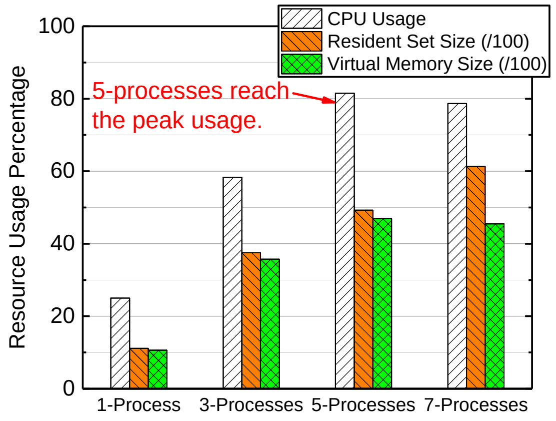


Fig. 4: The Average Usage of CPU, Memory and Virtual Memory by using 1-process, 3-processes, 5-processes, and 7-processes for the task of running 4000 records.

- U stands for the group of users who rate both item i and item j
- $R_{(u,i)}$ or $R_{(u,j)}$ means the score for such the item by the user
- $\overline{R}_u$ is the average of the *u-th* users ratings

3) Global Relevance (GR): The third method we proposed to use in the dPanther system is the Global Relevance (GR) recommender. The original GR is simply defined as:

$$GRScore \propto number\ of\ hits\ of\ the\ item \tag{5}$$

However, the collections in dPanther are generally consistent with geo-registration features, which appeal to users in specific regions rather than general interests. Therefore, we introduce enhanced GR scores with the consideration of the contribution as follows:

$$GR_u \propto N_i \quad where \quad L_i \in L_u \tag{6}$$

- Where *GR* is the Global Reference Score
- *u* is the current user
- *i* is the item in dPanther system
- *N* is the number of hits of item i
- *L* is the location information

4) Term Suggestion (TS)/Query Suggestion (QS): The last method which we implemented is Term Suggestion (TS)/Query Suggestion (QS). Since one of the major components is metadata search/query function, many unsuccessful visits are caused by the search function failures. The suggestion function contains two steps. Firstly, the system suggests the key word based on the users input. After a user hits the search, the system conducts a similarity analysis between the input search term and the subject key words from the metadata and then provides the suggestion. The similarity of the term to the metadata is generated by using the Jaccard index as suggestion by [27]:

$$S_{(a,b)} = \frac{|DS_a \cap DS_b|}{|DS_a \cup DS_b|} \tag{7}$$

- Where S is the Similarity Score between the search term and the item subject keywords
- a is the input search term
- b is the item in the system
- *DS* is the data set that consists of the key works

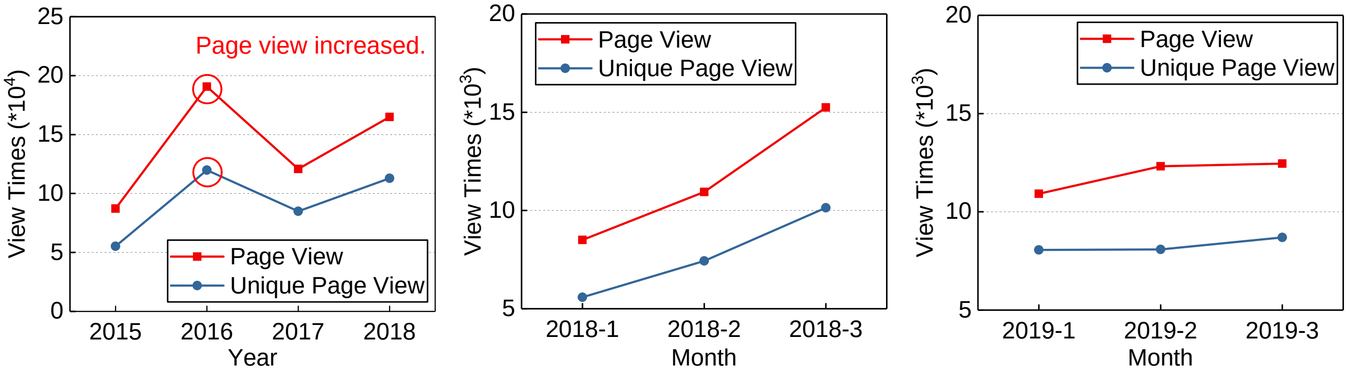
IV. Evaluations

We evaluate dpSmart from two different perspectives: computational complexity and the actual impact to its hosting Digital Repository Systems dPanther. The experimental evaluations answer the following questions:

- *Whether the multiple-process programming algorithm improves the hosting Digital Repository Systems performance?*
- *When it is integrated into the Digital Repository Systems, what are the benefits of dpSmart regarding the page views, bounce & drop-off rate?*
- *How are the Digital Repository Systems usability like Avg. Time on Page, Avg. Pageload, Avg. Direction Time, Avg. Server Reponse Time, and Avg. Page Overload Time improved by integrating dpSmart?*

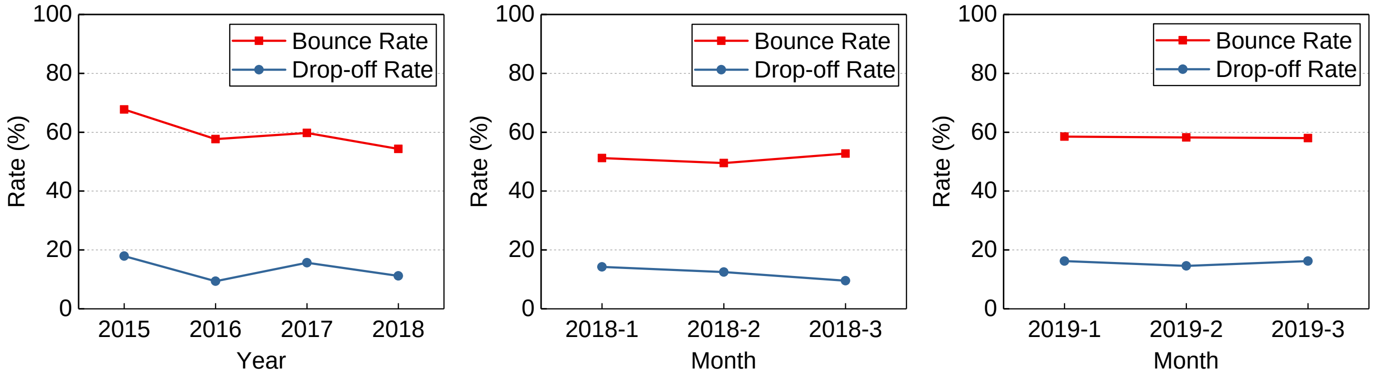
A. Experiment Setup and Data Collection

1) Multi-process Programming Experiment: The experiments were conducted on a windows machine with four cores



(a) The Page View Stats for the year 2015-2018.
(b) The Page View Stats from Jan. to Mar., 2018.
(c) The Page View Stats from Jan. to Mar., 2019

Fig. 5: The Page View Stats for the year from 2015 to 2018, from January to March in year 2018, and from January to March in year 2019.



(a) The Bounce & Drop-off Rate from the Year 2015-2018.
(b) The Bounce & Drop-off Rate from Jan. to Mar., 2018.
(c) The Bounce & Drop-off Rate from Jan. to Mar., 2019

Fig. 6: The Bounce & Drop-off Rate for the year from 2015 to 2018, from January to March in year 2018, and from January to March in year 2019.

CPU and 32 GB of memory. 4,000 metadata records are randomly selected from the database and processed to generate the item vector. We conduct the experiment by run the CB filtering algorithm against the 4,000 records by using 1-process, 3-processes, 5-processes, and 7-processes respectively. The algorithm run time, CPU Usage, and Memory Usage are collected.

2) System Data Collection: For the system evaluation, since `dPanther` is an existing production system, we focus on the real system statistics to evaluate if the recommendation module has a positive impact on the hosting system. Common performance metrics including Page View, Unique Page View, Average Time on Page, Bounce Rate, Drop-off Rate, Avg.Time on Page, Avg. Page Load Time, Avg. Direction Time, and Avg. Page Overload Time are collected for the evaluation. There are some milestones for the evaluation. The first module implemented in the system is the Customized Recommenders Module, which was published for production in January, 2016. The Automatic Web Server Log Mining Module, User-group clustering Module, and Recommendation Strategy Module are related to each other. Therefore, these three modules were implement all at once in January, 2019. In addition, we also applied the optimized multiple processes programming so that we can rebuild the model more frequently. In order to evaluate the impact of these three modules, we compared the system statistics from January, February, and March of year 2018 and year 2019. We rebuilt the model every week in the month of January and February and left the model as is in March. The statistics for this period are also analyzed for the system changes after the new implementation.

B. Multi-process Programming Evaluation

As shown in Figure 3a, the total run time for the 4,000 record process has been reduced from 365 seconds to 137 seconds for 1 process to 3 processes and further reduced to 98 seconds after increasing to 5 processes. But the total run time is only reduced from 98 seconds to 97 seconds after we increased to 7 processes. The figure also indicates that the more records processed, the more time are saved.

The CPU and Memory Usage over time in Figure 3b and Figure 3c also indicates that 5 processes approach best utilized the resources. Figure 4 also reveals that the average CPU Usage and Memory Usage reduced from 81.5% to 78.7% and 0.47% to 0.45% respectively.

C. Impact to the Hosting Digital Repository Systems

In Figure 5a, the page views and Unique Page Views have been increased from 87,168 to 190,665 and 55,311 to 119,783 respectively from the year of 2015 to the year of 2016. This figure also reveals that both Page View and Unique Page View dropped in year 2017 from 190,665 to 120,837 and 119,783 to 84,936, respectively. The reason may lie in that the model wasn't recalculated while new content and log data were generated. After recalculating the recommenders multiple times in the year 2018, we can see that the number increases again from 120,837 to 164,969 and 84,936 to 112,972 respectively. In Figure 7, the Avg. Time on Page also indicates that in 2015, users spent more time on average on a page, which are most likely from internal or academical users. When the recommenders work and attract more general/layman users, the Avg. Time on Page dropped accordingly.

Figure 6a shows the Bounce Rate and Drop-off Rate trend from the year 2015 to the year 2018. Similar to the Page View statistics, both Bounce Rate and Drop-off Rate show a reverse proportional relation to the recommenders module. Even though this is not as clear as the Page View statistic, it still shows that an updated recommenders module can help to reduce the Drop-off rate. However, as shown in Figure 6b and Figure 6c, the same trend cannot be confirmed after implementing the Automatic Web Server Log Mining Modules, User-group clustering Module, and Recommendation Strategy Module. This is mainly because that the stereotyping recommender is more used to enhancing the recommendation quality and reducing the recommendation noise. It is not necessarily has a direct impact on he Bounce Rate and Drop-off Rate.

From the Figure 7 and Figure 8, we can verify that the system usability does not decrease a lot compared to the year 2018. The Avg. Page Load Time, Avg. Direction Time, Avg. Server Response Time and Avg. Page Overload Time are all increased compared to the year 2018, but still much lower than the year 2016 and year 2017. Secondly, the Page View in Figure 5b and Figure 5c also increased in January and February in the year 2019 from 8,502 to 10,916 and from 10,942 to 12,314 respectively. The Page View has decreased in March from 15,239 to 12,450 compared to the year 2018 which caused by the out of dated model.

In summary, dpSmart has remedied the computational complexity problem by implementing the multi-process programming. The system statistics also confirmed that dpSmart effectively increased the Page Views of the hosting system by implementing the customized recommenders.

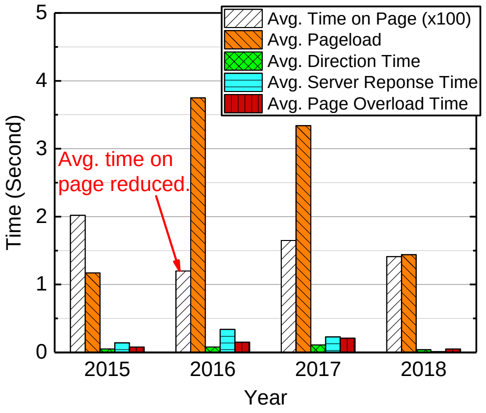


Fig. 7: The system usability statistics from 2015 to 2018.

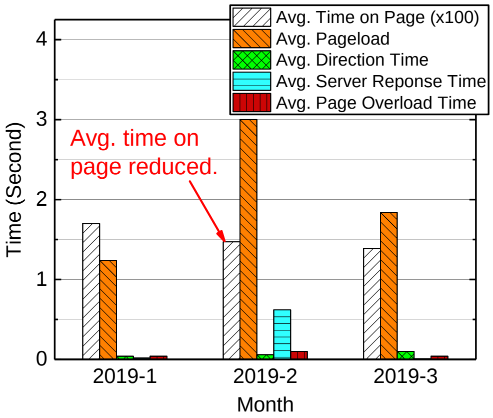


Fig. 8: The system usability statistics from Jan. to Mar., 2019.

V. Conclusion

In this paper, we design and implement a flexible group based recommendation framework, called dpSmart, and integrate it into a real-world Digital Repository System dPanther (http://dpanther.fiu.edu). The goal of dpSmart is to address the limitations of the digital repository system which includes low discoverability, poor usability, and high drop-off visit rates. dpSmart provides capability of automatic user vector generation from log data, clustering based user group identification, flexible recommendation strategy, and multiple recommenders integration. The proposed system also suggests several customizing methods to customize the specific recommendation algorithms for the specific system. The experimental evaluation shows that by applying the multi-process programming, the model building time can be significantly reduced. The system usage statistics also indicate that during the evaluation time from January 2019 to February 2019, the Page Views have increased from 8,502 to 10,916 and 10,942 to 12,314 respectively, compared to 2018, demonstrating the effectiveness of our proposed framework.

Acknowledgment

This work is supported by Department of Homeland Security (DHS 2017-ST-062-000002).

References

[1] "dPanther digital repository system in florida international university." http://dpanther.fiu.edu. Accessed: 2019-04-27.

[2] G. Geisler, D. McArthur, and S. Giersch, "Developing recommendation services for a digital library with uncertain and changing data," in *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '01, (New York, NY, USA), ACM, 2001.

[3] J. Beel, S. Langer, M. Genzmehr, and A. Nürnberger, "Persistence in recommender systems: giving the same recommendations to the same users multiple times," in *International Conference on Theory and Practice of Digital Libraries*, pp. 386–390, Springer, 2013.

[4] T. Bogers and A. Van den Bosch, “Recommending scientific articles using citeulike,” in *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 287–290, Citeseer, 2008.

[5] K. Chandrasekaran, S. Gauch, P. Lakkaraju, and H. P. Luong, “Concept-based document recommendations for citeseer authors,” in *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pp. 83–92, Springer, 2008.

[6] M. Gori and A. Pucci, “Research paper recommender systems: A random-walk based approach,” in *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI’06)*, pp. 778–781, IEEE, 2006.

[7] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, *et al.*, “Collaborative filtering recommender systems,” *Foundations and Trends® in Human–Computer Interaction*, 2011.

[8] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *Recommender systems handbook*, pp. 73–105, Springer, 2011.

[9] B. Lamche, E. Pollok, W. Wörndl, and G. Groh, “Evaluating the effectiveness of stereotype user models for recommendations on mobile devices.,” in *UMAP Workshops*, 2014.

[10] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[11] J. Beel, B. Gipp, S. Langer, and C. Breitinger, “Research-paper recommender systems: a literature survey,” *International Journal on Digital Libraries*, vol. 17, pp. 305–338, Nov 2016.

[12] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, “Mining user similarity based on location history,” in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, p. 34, ACM, 2008.

[13] C. Musto, F. Narducci, P. Lops, M. de Gemmis, and G. Semeraro, *Integrating a Content-Based Recommender System into Digital Libraries for Cultural Heritage*, pp. 27–38. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[14] S. Philip, P. Shola, and A. O. John, “Application of content-based approach in research paper recommendation system for a digital library,” *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 5, no. 10, 2014.

[15] A. F. Smeaton and J. Callan, “Personalisation and recommender systems in digital libraries,” *International Journal on Digital Libraries*, vol. 5, pp. 299–308, Aug 2005.

[16] D. Sulieman, M. Malek, H. Kadima, and D. Laurent, “Toward social-semantic recommender systems,” *Int. J. Inf. Syst. Soc. Chang.*, vol. 7, pp. 1–30, Jan. 2016.

[17] J. Beel, “Towards effective research-paper recommender systems and user modeling based on mind maps,” *CoRR*, vol. abs/1703.09109, 2017.

[18] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, “Web service recommendation via exploiting location and qos information,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 1913–1924, July 2014.

[19] E. Ntoutsi, K. Stefanidis, K. Rausch, and H.-P. Kriegel, “Strength lies in differences: Diversifying friends for recommendations through subspace clustering,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 729–738, ACM, 2014.

[20] E. Rich, “User modeling via stereotypes,” *Cognitive Science*, vol. 3, no. 4, pp. 329 – 354, 1979.

[21] H.-P. Kriegel, P. Kröger, and A. Zimek, “Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 1, p. 1, 2009.

[22] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, “Fast algorithms for projected clustering,” in *SIGMOD Conference*, 1999.

[23] R. Real and J. M. Vargas, “The probabilistic basis of jaccard’s index of similarity,” *Systematic biology*, 1996.

[24] G. Jeh and J. Widom, “Simrank: a measure of structural-context similarity,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 538–543, ACM, 2002.

[25] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles, “Learning multiple graphs for document recommendations,” in *Proceedings of the 17th International Conference on World Wide Web*, WWW ’08, (New York, NY, USA), pp. 141–150, ACM, 2008.

[26] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, “A new user similarity model to improve the accuracy of collaborative filtering,” *Knowledge-Based Systems*, 2014.

[27] T. Lüke, P. Schaer, and P. Mayr, “A framework for specific term recommendation systems,” *CoRR*, vol. abs/1407.1539, 2014.