Items Ranking Algorithm For Museums Selection

Alexander Setiawan[1, a)], Silvia Rostianingsih[1, b)], Andreas Handojo[1, c)]

Author Affiliations
[1] *Informatics Department, Faculty of Industrial Technology, Petra Christian University*
Siwalankerto Street No.121-131, Surabaya, East of Java, 60236

Author Emails
[a)] Corresponding author: alexander@petra.ac.id
[b)] silvia@petra.ac.id
[c)] handojo@petra.ac.id

Abstract. The ranking algorithm procedure is an algorithm used to determine the rank or position in a dataset for certain criteria. One complaint that is often found when using an application is "the speed of user interaction with the application". This is due to many things, one of which is an application that complicates/complicates the user when used. For example, when a user wants to see a certain product that has been seen before, but there is an application that requires the user to go to the same section to be able to see the product again. This causes application users to decrease. Therefore, an algorithm is needed to make it easier for users to access a feature in the application. One of the algorithms implemented to make it easier to access the features found in the applications we use is the Least Recently Used (LRU), Least Frequently Used (LFU), and Least Recently/Frequently Used (LRFU) item ranking algorithms. We use this algorithm concept to print the conditions where the features most frequently visited or used by users will be displayed. This process will speed up and make it easier for users who want to visit or access previous data. Ranking algorithm procedure is an algorithm used to determine rank or position in a dataset for certain criteria.

INTRODUCTION

Indonesia is a country with a lot of history, from various cultures and religions. However, many people do not know Indonesian history well. They only know Indonesian history through their history lessons at school. In addition, with a museum that maintains Indonesian history, visitors / people who are interested in visiting the museum are still small. This is because people do not know where to look for the nearest museum to them. The public is also still confused about what categories are in the museum and what content to the present.

It is often found that the use of applications is long-winded, causing a decrease in the number of users. "Interaction speed" is one of the important factors in the development of features in an application. Then a method or algorithm is needed that can overcome the negligence of user interaction with the application. The existing algorithm is expected to speed up, simplify, and provide brief instructions for application users.

Through the Insight Surabaya application, visitors will have access to audio guides, virtual tour guides, information regarding museums in Surabaya. This application will also provide a search feature that makes it easier for visitors to find collections and specific information according to their interests and desires.

Apart from providing benefits for visitors, the Insight Surabaya application will also be an important tool for curators and museum staff in promoting their collections, communicating important information to visitors, and building public awareness about the importance of preserving and maintaining cultural heritage.

Based on the data that we have researched and taken from opendata.surabaya.go.id, the number of museum visitors in Surabaya is relatively low when compared to other entertainment or tourist locations. Although there are many factors that could cause this phenomenon to occur, our group wants to address factors namely: lack of awareness of the museum, not knowing the location of the museum, not knowing what the museum contains. To overcome this

problem, we created the Insight Surabaya application which is capable of doing virtual tours and filtering of museums in Surabaya.

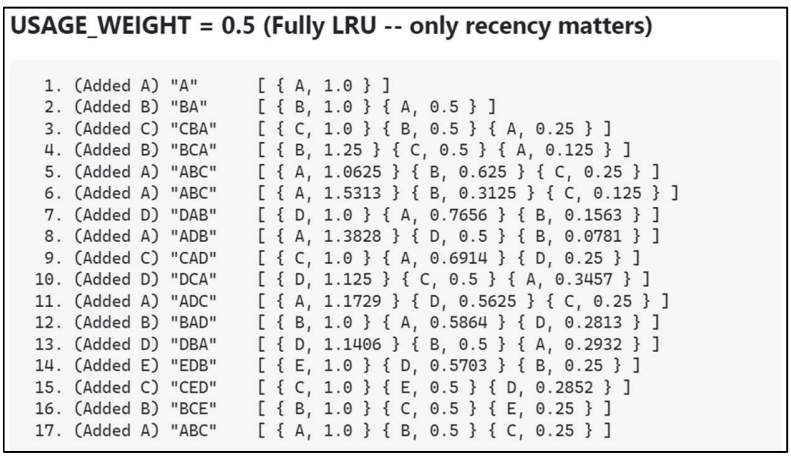
LITERATURE REVIEW

This application has several features linked as an alternative to Google. Namely the virtual tour feature, content-based filtering, and collaborative filtering. With this feature, users can search for museums that are more suitable to their preferences, and at the same time can take a tour to find out the layout and attractions of the museum they want to visit/find out more about. However, the application has limitations, namely the application is only able to review collections from the museum broadly, and is not able to show every detail of the museum. With these applications and features, our group hopes to be able to increase the number of visitors to museums in Surabaya.

A ranking algorithm is an algorithm or procedure for creating a sequence or sorting data in a certain category. Usually this sorting can be based on number, weight, size, or any type that can be compared between one data and another. One example of applying the ranking algorithm is sorting data from the largest size to the smallest size or sort by size in the folders that we use every day in folders with Windows OS. There are also many applications of ranking algorithm methods, one of which is the LRU, LFU, and LRFU methods.

The Least Recently Used (LRU) page replacement policy assumes that page reference pattern in the recent past is a mirror of the pattern in the near future. Pages that are accessed recently are likely to continue to be accessed and ought to be kept in physical memory.

The Least Recently Used (LRU) is the most optimal algorithm, because the LRU algorithm replaces pages that will not be used again for a long time. That way the efficiency will increase and reduce the occurrence of page faults. This algorithm has the lowest page fault among all other algorithms, but this algorithm is very difficult to implement. The system does not know which pages will be used next. The Model LRU can be seen Figure 1.



USAGE_WEIGHT = 0.5 (Fully LRU -- only recency matters)

```
 1. (Added A) "A"      [ { A, 1.0 } ]
 2. (Added B) "BA"     [ { B, 1.0 } { A, 0.5 } ]
 3. (Added C) "CBA"    [ { C, 1.0 } { B, 0.5 } { A, 0.25 } ]
 4. (Added B) "BCA"    [ { B, 1.25 } { C, 0.5 } { A, 0.125 } ]
 5. (Added A) "ABC"    [ { A, 1.0625 } { B, 0.625 } { C, 0.25 } ]
 6. (Added A) "ABC"    [ { A, 1.5313 } { B, 0.3125 } { C, 0.125 } ]
 7. (Added D) "DAB"    [ { D, 1.0 } { A, 0.7656 } { B, 0.1563 } ]
 8. (Added A) "ADB"    [ { A, 1.3828 } { D, 0.5 } { B, 0.0781 } ]
 9. (Added C) "CAD"    [ { C, 1.0 } { A, 0.6914 } { D, 0.25 } ]
10. (Added D) "DCA"    [ { D, 1.125 } { C, 0.5 } { A, 0.3457 } ]
11. (Added A) "ADC"    [ { A, 1.1729 } { D, 0.5625 } { C, 0.25 } ]
12. (Added B) "BAD"    [ { B, 1.0 } { A, 0.5864 } { D, 0.2813 } ]
13. (Added D) "DBA"    [ { D, 1.1406 } { B, 0.5 } { A, 0.2932 } ]
14. (Added E) "EDB"    [ { E, 1.0 } { D, 0.5703 } { B, 0.25 } ]
15. (Added C) "CED"    [ { C, 1.0 } { E, 0.5 } { D, 0.2852 } ]
16. (Added B) "BCE"    [ { B, 1.0 } { C, 0.5 } { E, 0.25 } ]
17. (Added A) "ABC"    [ { A, 1.0 } { B, 0.5 } { C, 0.25 } ]
```

FIGURE 1. Model LRU

So from the example above, every time an item is added, the value of that item will be added by 1 then the remainder is multiplied by the usage weight which is 0.5. For example, in the first step, A is added, so the value of A is 1 and nothing is multiplied by 0.5 because previously it was empty. Then when B is added, the contents of the List contain 2 items, namely B and A. Because B has just been added, the value of B is 1, while the value of A, which was previously 1, is multiplied by 0.5 so that it becomes 0.5. So the value added in the Nth step will always have the largest value of the value in the previous steps regardless of how many times the item is added.

The LRU-K Algorithm specifies a page replacement policy when a memory frame is needed for a new page requested: the page p to be dropped (i.e., selected as a replacement victim) is the one whose backward K-distance (bt(p,K)) is the maximum of all pages in memory. The only time the choice is ambiguous is when more than one page has bt(p,K) $= \infty$. In this case, a subsidiary policy may be used to select a replacement victim among the pages with infinite Backward K-distance; for example, classical LRU could be employed as a subsidiary policy [5].

The proposed Least Frequently Used (LFU) algorithm has a runtime complexity of O(1) for each of the dictionary operations (insertion, lookup and deletion) that can be performed on an LFU cache. This is achieved by maintaining 2 linked lists; one on the access frequency and one for all elements that have the same access frequency [2].

The Least Frequently Used (LFU) is an algorithm that assumes lower-value classes will be reused in a relatively long time. The LFU algorithm is easy to understand and implement. The implementation of the algorithm is very efficient because there are not many steps in page selection [3]. The Model LFU can be seen Figure 2.

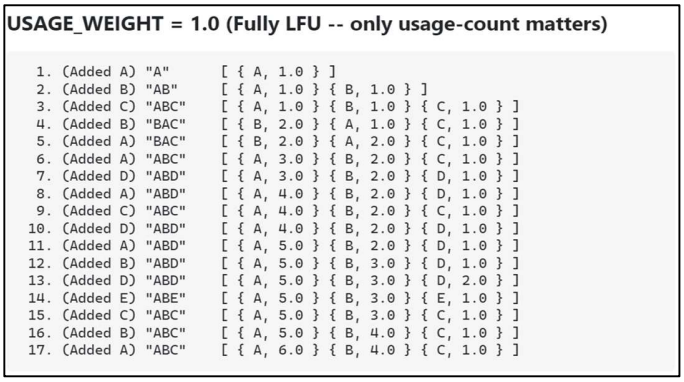


FIGURE 2. Model LFU

The Least Recently/Frequently Used (LRFU) policy associates a value with each block. This value is called the CRF (Combined Recently and Frequency) value and quantifies the likelihood that the block will be referenced in the near future. Each reference to a block in the past contributes to this value and a reference's contribution is determined by a weighing function F(x), where x is the time span from the reference in the past to the current time [4].

The LRU and LFU algorithm with popularity content generated. Results show Popularity Content replacement policy achieved a higher hit rate ratio compared with LRU and LFU. Moreover, LRFU replacement policy compared with LRU based on content store size variation detailed in [1]. The results evaluate hit rate ratio and time complexity of algorithm and show that LRFU achieved higher hit rate ratio compared with LRU. However, due complex weighting function of LRFU affect higher time complexity. The Model LRFU can be seen Figure 3.

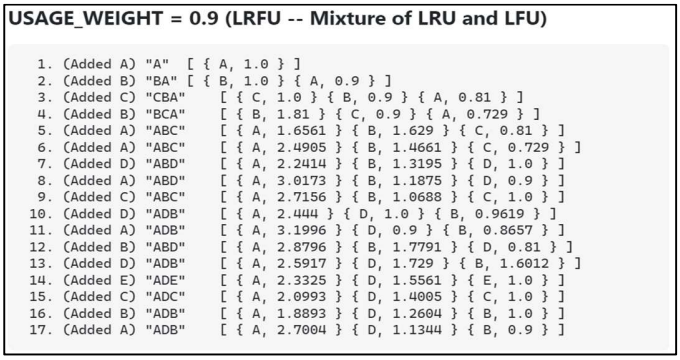


FIGURE 3. Model LRFU

ANALYSIS MODEL TECHNIQUE IN THIS RESEARCH

In making the Insight Surabaya application, we took photos of the museum from various angles with a first-person point of view which we later combined to become a virtual tour of the Insight Surabaya application. In addition, we also take documentation and record information from various objects contained in the museum, which later we will present information about these objects in the Insight Surabaya application can be seen Figure 4.

FIGURE 4. Example Insight Surabaya Application

In an increasingly advanced digital era, the presence of technology can be an effective tool to bridge this gap. The Insight Surabaya application is an innovative solution designed to provide new interactive experiences to museum visitors. With the help of this application, visitors can access in-depth information about the museum's collections.

Through the Insight Surabaya application, visitors will have access to audio guides, virtual tour guides, information regarding museums in Surabaya. This application will also provide a search feature that makes it easier for visitors to find collections and specific information according to their interests and desires [6].

APPLICATION OF THE LRU RANKING ALGORITHM IN MUSEUM APPLICATIONS.

To apply the ranking algorithm to museum applications, we use the LRU method for each museum visited. We take the museum id with the initial weight set to 1.0 (float/double). We also apply a data length limit that can be visited by 3 for each museum visited. Then, with this ranking data, we process it again so that this position condition is returned to the recently viewed feature in descending sort position. The ranking will always be changed or processed every time the user selects the museum being visited, the weight will be added by 1 and the remainder will be multiplied by 0.5. So, the museum whose id is the last or Nth entry must have the highest rank value, can be seen Figure 5. With the Sort by Rating feature, users can see museum recommendations in the Insight Surabaya application based on the rating of each museum, the search for a museum can be seen Figure 6, and input keywords can be seen Figure 7.

FIGURE 5. The Highest Rank Value

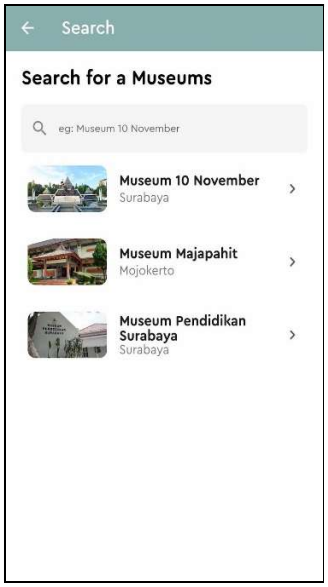


FIGURE 6. The Search for a Museums

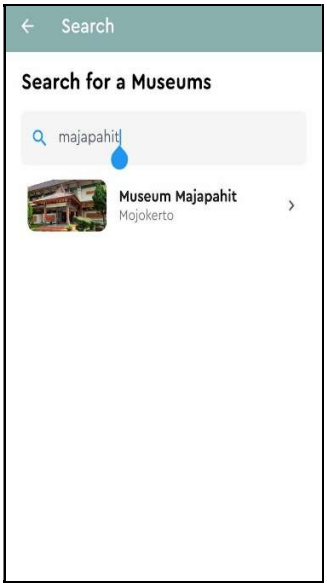


FIGURE 7. Input The Keywords

After that, it is necessary to find the rating of each user who is rating a museum, then the ratings of all users and all museums are normalized where for a rating >= 3 it will be 1 and if <=2 it will be 0 so that the calculation of collaborative filtering is good. Here the rating results will be weighted to 80% and the review results will be added as the results of the recommendations. If the value is greater, it will be more likely to be included in the recommendation.

For content based we make a recommendation system based on the type most liked by the user, if the user bookmarks/favorites a certain type of museum then the results that will appear in the recommendation system are all museums of that type for example: if a user bookmarks a blockbuster museum with the type of art & children, what will be recommended is the museum with the type of art / children and the maximum number of recommendations is 5, so if the results exceed 5, the type with the most bookmarks will be searched and then sorted in descending order [7].

APPLICATION CONNECTION USING API AND REST API

An application programming interface (API) is software that connects applications together. The purpose of the API itself is to be able to share data between one application and another and to make the process of developing and creating applications faster because separate functions are available and there is no need to create these functions again (Figure 8). The Representational State Transfer (REST) API is an architectural design that resides within the API. The way the REST API works is that the REST client will access data on the REST server. Each piece of data will be identified by a Universal Resource Identifier (URI) or a global ID (Figure 9). REST APIs consist of several formats, such as JSON, XML, and text. Recently, JSON has been the most widely used format. JSON consists of key-value pairs separated by a comma [8].

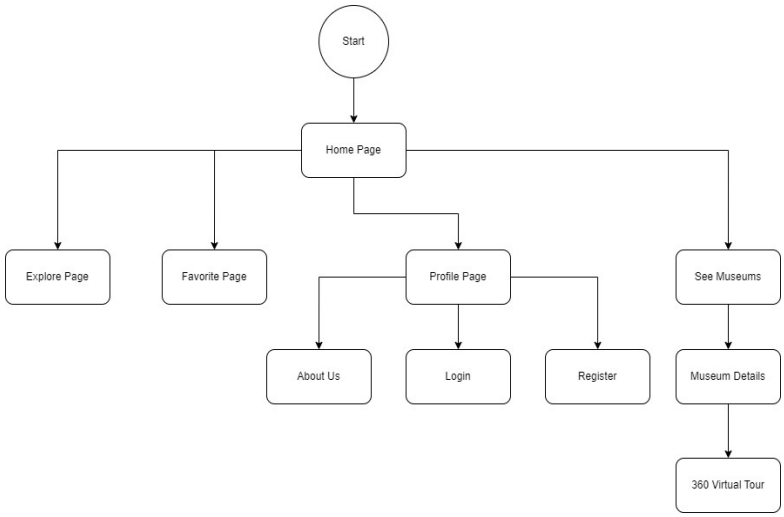


FIGURE 8. Flow Museum Insight

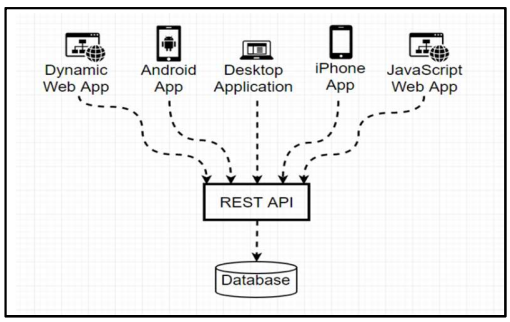


FIGURE 9. REST API overview

For the API register, we will receive input in the form of JSON, which contains 4 fields: name, email, password, and password confirmation. This API register will check whether the email entered is already in the database or not; if it is, then the account creation will be rejected; if it is not, then it will proceed to the next checking stage.

For API login, we will accept input in the form of JSON, which contains two fields: email and password. If the email and password sent by the user match those in the database, then we will return a JSON object containing a JWT (JSON Web Token). JWT is a long string used to authenticate requests sent by logged-in users (Figure 10). We set the validity of this JWT for 1 day, aka 24 hours. So, if the user does not open the application for more than 24 hours, then they will have to re-login. The following is an example of the output of the login API if the user has successfully logged in.

FIGURE 10. JSON Web Token

For the refresh API, it will update the JWT token every time they open the app. If they open the application for less than 24 hours, they will get the latest JWT, and the old JWT will expire. This is very important, so they don't have to type in their email and password to log in every 24 hours. For the logout API, we will accept JWT input and deactivate the JWT so that it can no longer be used to request resources from other APIs. For the museum API, there will be 2 APIs: the first is the list museum API, which will return all lists of museums available in the application. This API will be useful for displaying all museums in the Home section of the museum application. The second API is the museum details API, which will receive the museum ID parameter and return museum details such as the name, description, location, and the museum's. For the API profile, it will return the name of the user so that on the profile page, they can see their name. All APIs are only accessible if they include a valid JWT.

We tested the application prototype on 50 museum visitors who were visiting museum at the time. The test results are reported in Table 1. It is possible to infer that the application created entices people to come and learn more about the Museum Insight (80%).

TABLE 1. Respondents' Responses toward Respondents' Interest in the Application

	Very Not Interesting	Not Interesting	Interesting	Very Interesting	Total Respondents	Respondent Percentage
Age 12-15	0	0	5	15	20	40.00%
Age 16-19	0	0	2	18	20	40.00%
Age 20-35	0	0	3	7	10	20.00%
Total Respondents	0	0	10	40	50	100.00%
Percentage Answer	0.00%	0.00%	20.00%	80.00%	100.00%	

CONCLUSION AND DISCUSSION

We use the LRU algorithm, because the LRU algorithm implements a recent ranking of users based on the most recently submitted user access. Sorting based on the user's last input choice is intended to speed up the acquisition of the latest data accessed by the user, so that in the future a selection data will be stored which is sorted on the basis of the user. It is hoped that this access will provide convenience and speed for users to access data that is most relevant to users.

ACKNOWLEDGMENTS

We thank the Petra Christian University Research and Community Service Institute for funding this research. We also thank Museum in Surabaya for the permission given to take pictures and collect data. We are grateful for the very useful input and discussion from Petra Christian University Mobile Computing Research Group colleagues.

REFERENCES

1. G. Petrus B.K., “Simulation and Study of Replacement Strategy in NDN,” Master’s Thesis, Bandung Inst. Technol., 2020.
2. S. Bagchi, S. S. Bhat and A. Kumar, "O(1) time sorting algorithms using spiking neurons," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 2016.
3. N. Alzakari, A. B. Dris and S. Alahmadi, "Randomized Least Frequently Used Cache Replacement Strategy for Named Data Networking," 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2020.
4. Y M. A. P. Putra, H. Situmorang and N. R. Syambas, "Least Recently Frequently Used Replacement Policy Named Data Networking Approach," 2019 International Conference on Electrical Engineering and Informatics (ICEEI), Bandung, Indonesia, 2019.
5. Ruchin Gupta and Narendra Teotia. Article: Least Recently Used Page Replacement using Last Use Distance (LRUL). International Journal of Computer Applications 84(2):8-10, December 2013
6. Setiawan, Alexander, Rolly Intan, and Ivan Gunawan. "Data traffic synchronization application in electronic map." AIP Conference Proceedings. Vol. 2691. No. 1. AIP Publishing, 2023.
7. Rostianingsih, Silvia, Alexander Setiawan, and Michael Boentarman. "Customer clustering using K-means clustering: Supporting customer relationship management system." AIP Conference Proceedings. Vol. 2691. No. 1. AIP Publishing, 2023.
8. Setiawan, Alexander, et al. "Virtual application technology of citizen journalism based on mobile user experience." Journal of Physics: Conference Series. Vol. 1502. No. 1. IOP Publishing, 2020.