Impact of Sequence Mining on Web-Page Recommendations in an Access-Log-Driven Recommender System

Nguyen ThaiSon and Lukas Siemon

Information Search and Retrieval,
Lecturer: Francesco Ricci,
Academic year 2011-2012 - 2nd Semester,
Free University of Bolzano
`http://www.inf.unibz.it/~ricci/ISR/`

Abstract. While a lot of existing web solutions could benefit from recommender systems, this usually requires (complex) data collection and/or integration into the underlying database system. Since not all administrators are willing or capable of doing this, we are analyzing the capabilities of access log driven collaborative filtering recommender systems to generate web-page recommendations. We are using clustering and sequence mining to build a prototype that aims to understand the user long-term behavior. We then analyze the limitations and benefits of this system by running it on real world data. Our evaluation shows that by taking the order of user page accesses into account we can improve the quality of recommendations. Since we were restricted to data from one web-page in the domain of MMORPGS, further investigation is still required.

Keywords: recommender system, clustering, sequence mining, implicit feedback, collaborative filtering, long-term recommendation

1 Introduction

Recommender systems (RS) are indisputable very powerful and can improve the user experience a lot. On the other hand, recommending has to be done carefully, since "wrong" recommendations can confuse and annoy the user. Hence it is advised that any RS should be field tested. Reducing the amount of recommendations can aid the quality. [2]

While RS are very useful to deal with information overload [8], implementing a content-based RS into an existing website is not a trivial task. Assume one wants to utilize an existing system, which takes some data input and generated recommendations: The first step is to identify the RS "engine" that suits the purpose best. The RS will now require existing data to work with and this data might not be available or not available in a format that the RS can work with. Generating and converting the data can be very time intensive and can add significant overhead to the system.

The next step is then to find a way to present the generated recommendations effectively inside the web-page. This is of great importance, otherwise the user will not accept the recommendations or get annoyed. There are a number of papers dealing with this problem, e.g. [6].

In this project we focused on generating web-page recommendations for users that were "frequently using a specific web-domain to acquire knowledge". The requirements we set for the system was that it could reliable identify each user and that, to a certain extend, the pages the user visited reflect his "knowledge" of what the content of the domain describes.

One goal we wanted to achieve was to make the process of data-collection simple and lightweight (e.g. leaving a minimal footprint to the existing source-code and producing little overhead for the server). A second goal was to generate recommendations based on a long term understanding of the user. We tried to model the user access behavior in such a way that it represents the type of user and use this type to predict future accesses.

In the following we will simplify a recommender system as a process that consists of three independent steps: Collecting user generated data, generating recommendations based on this data and showing the recommendations to the user.

In our generic prototype we focus on making the first two steps easy, reusable and accurate. The biggest simplification is the data generation. We simply stored page access logs. This can be done very easy and lightweight with Google Analytic[1] by using Java-script and Cookies. We will give an example of this in Chapter 4.

The core part of the prototype then consists of 1) computing similarity values between pages based on user sessions 2) clustering the pages by these similarity values using agglomerative hierarchical clustering[2] 3) finding similar users by comparing cluster access sequences and 4) generating recommendations for the users and evaluating them.

The prototype was run on access logs that were collected from a database website of an MMORPG [11]. We will discuss why we have chooses this domain and our results in chapter 5.

The further parts of this report consist of 1) an overview of some related work, 2) a further motivation and an abstract description of our prototype, 3) a detailed description of the system including the technologies we used, 4) an evaluation of our system using real-world data and 5) our conclusion.

2 Related Work

The idea of treating accesses to web-pages as "purchases" and creating recommendations based on association rules and collaborative filtering (CF), is no novelty. Association rules have successfully been used in recommender systems

[1] `http://www.google.com/analytics/`

[2] This step creates user access pattern, while also providing the notion of page- and pattern-similarity.

that predict which web page can be useful to a user. A limitation of the approach is the distribution of access logs and hence limitation of recommendations to a certain domain. [7][9]

[1] describes a hybrid system that utilizes data mining techniques to generate web-page recommendations. The system crawls and analyzes web-page content and references users access behaviors to create recommendations. The mined association rules are used to predict the current session behavior of the user. In our approach we try to utilize the data further to predict the behavior in the next sessions.

(User) context is any information that can be used to characterize the situation of any person, place, or object which is relevant to the interaction between a user and an application. This includes the user and application themselves. [3]

User context has been used before for recommendation generation [4]. Since this context is not always available, it might be observed from the previous user actions. Grouping the user actions can help to identify meaningful patterns in the user behavior if the training data is not large enough or to sparse [5].

The analysis of recent sequence accesses has been mentioned in the domain of music recommendations, where certain user context, like mood, is usually short lived [5]. There are however no reasons why this should not be used to observe longer lasting context. In our prototype we try to combine techniques described above to create a system that "learns" the user and recommends based on the long-term user development.

3 System Setting and Overview

The idea that we tried to capture in our RS is that individual user actions can describe the development of this user over time, e.g. what the user knows and is interested in at a given point in the past. As an example of a domain that our RS can work with let us consider some powerful graphic software. The software has so many features, that usually one user can not know or access all the features. There are different types of users that will use the software: One person might be interested in web design and learns about the different slicing tools and techniques. Another user will use an electronic drawing board to create concept art. Yet another user creates pix-elated art and has a very different set of skill. Knowing about the different users and their preferences can be useful to generate recommendations.

The graphic software will likely have a web-site that provides:

- Information about products and free and (paid) plug-ins for the software
- Documentation and Troubleshooting for the software
- Different forums that provide further information and allow for discussions
- Information about releases and known problems
- API access for the software

A user visiting the website will most likely want to learn "more" about certain parts of the program or give feedback. Some examples might be: Learning about a certain feature the program provides, learn how to use a feature, learn how to improve known techniques, learn about "best practice", learn about plug-ins, and so on. Recommendations could predict these needs and suggest certain topics in advance. If users have found a topic interesting, chances are high that a similar user will find it interesting as well. The main goal of these recommendations would be to show previously unknown, relevant topics to the user, that are a "logical consequence" of what the user knows already and was interested in before.

Assuming that the user has registered the program and is logged into the website, one can easily log and identify all user accesses.

Our system can then deduce the "type" of user by processing the sequence of accesses. Similar users are retrieved by comparing these sequences.

An abstract view on our integrated prototype can be found in Figure 1.

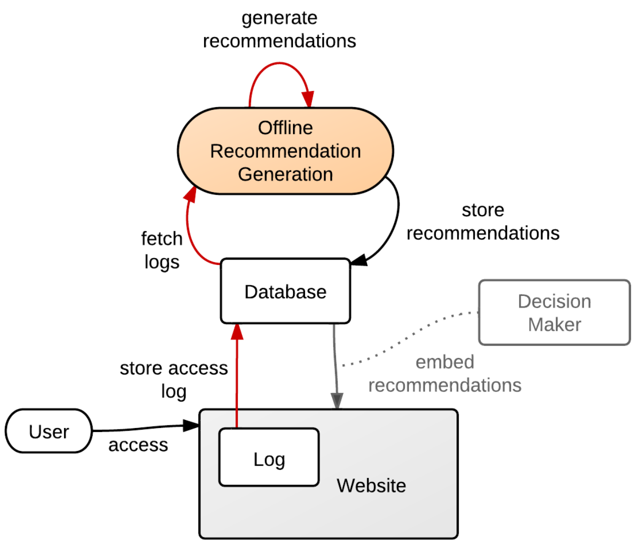


Fig. 1. An abstract view on the prototype that we implemented. The red arrows highlight the parts of the process that we focused on.

4 Technologies and System Details

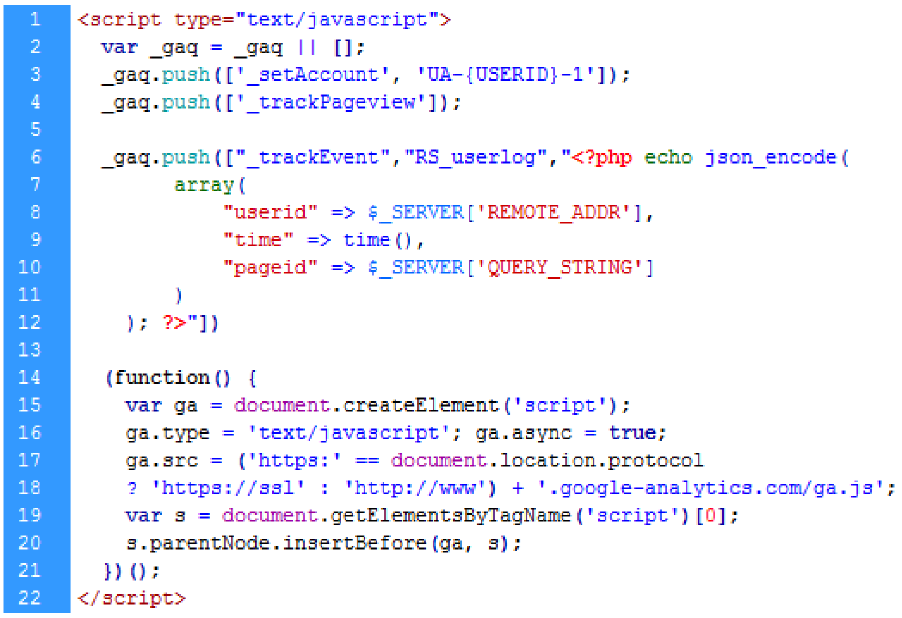
Our prototype consists of three parts:

– Data Collection (using Google Analytic Events)
– Generating Recommendations (using clustering and sequence mining)
– Evaluating the Recommendations

These parts shall now be described in more detail, including the different technologies we used.

4.1 Data Collection

Since we wanted to built a lightweight system, we used Google Analytics to log user page accesses. Once a Google Analytics account is set up one can simply add the code[3] as displayed in Figure 2 to the head part of each page. Google Analytics will store the tracked events in a cookie file and only flush every now and then[4]. Since Java-script is client side, the only overhead for the server, is sending the additional data volume[5].



```
<script type="text/javascript">
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-{USERID}-1']);
  _gaq.push(['_trackPageview']);

  _gaq.push(["_trackEvent","RS_userlog","<?php echo json_encode(
        array(
            "userid" => $_SERVER['REMOTE_ADDR'],
            "time" => time(),
            "pageid" => $_SERVER['QUERY_STRING']
        )
    ); ?>"])

  (function() {
    var ga = document.createElement('script');
    ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol
    ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(ga, s);
  })();
</script>
```

Fig. 2. Google Analytic Event code to track page accesses. Similar to the one used in our prototype.

The events are now stored on a separate Google server and can be conveniently accessed from a Java program through the Google Analytics API[6]. What the Java program can now work with, is a set of *userid, timestamp, pageid* - triples. A typical sample of the logs for one user can be seen in 3. This leads us to the next step, processing the data.

[3] For further details see `https://developers.google.com/analytics/devguides/collection/gajs/eventTrackerGuide`

[4] This means the logs are not accessible in real-time.

[5] If pages are created dynamically, updating the time-stamp has to be done client side and things get a bit more technical. This was done for our data collection, but the details are not of great importance for this report.

[6] Further information on how to set this up can be found at `https://developers.google.com/analytics/devguides/reporting/core/v2/gdataJava`

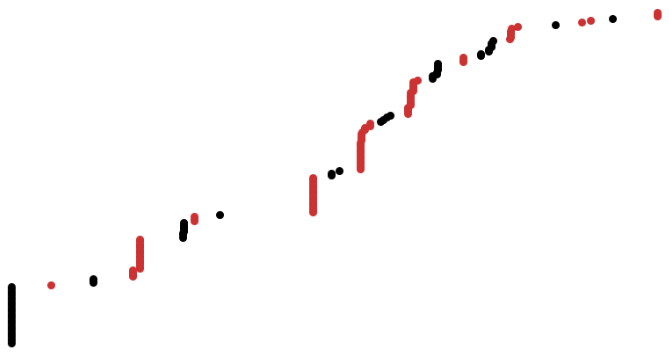


Fig. 3. Visualization of a typical user access log. The x-axis is the time of access and the y-axis the number of hits so far. The colors separate the different sessions that were formed by our system.

4.2 Generating Recommendations

When we first analyzed the data format, we came up with different strategies to measure user similarity. Since the information gain from a single page visit is very low, it was clear to us that we had to use some kind of Data Mining technique. After reading [9], which gives a good general view on data mining in RS, we decided to combine two techniques: clustering and sequence mining.

Our idea is that a user can be characterized by what types of pages (clusters) he accessed over time. In theory this can represent the user quite well and our experiments suggest that sequence mining can increase the precision of the recommendations even further. We will discuss this observation in Chapter 5.

While developing our idea and after every step, we analyzed the extracted information and tried to identify existing problems. In the following we will go through the different steps of our algorithm and explain their purpose.

Preprocessing. The main purpose of preprocessing is to reduce the noise in the data. When analyzing our data we discovered that a lot of users accessed two or more pages in a very short time period. We identified a number of reasons for this behavior: 1) The user opened a lot of different tabs to look at all of the pages[7]. 2) The user quickly observed that the first page was not what he was looking for and continued browsing. 3) The user used the previous pages to quickly navigate to the desired page.

For 2) and 3) it is helpful to only consider the last page hit, while for 1) it is hard to guess what the user was looking for. In our prototype we discarded page hits that preceded another page hit by only a few seconds as we found that this had a positive influence on the generated recommendations.

Page Clustering. Since many pages usually represent similar information it is only natural to group them together. We did not have any way to do this

[7] This typically happens when the result representation of a search does not give enough or not the right information.

semantically, so we decided to group the pages by statistical co-occurrences: The similarity between two pages was generated by looking at how often two pages were accessed together. The assumption that we make is that pages that are accessed together represent topics of interest that belong together.

To compute the page similarity, we divided each user time-line[8] into sessions. To generate the sessions we split the time-line whenever a gap was detected that was bigger than a certain threshold. We then counted in how many sessions two pages p_1, p_2 of the set of all pages P were accessed: $Acc(p_1, p_2) \in \mathbb{N}$. And in how many sessions a page was accessed: $Acc(p_i) \in \mathbb{N}, i \in \{0, 1\}$. The similarity Sim of two pages was then computed by (1).

$$Sim(p_1, p_2) = \frac{Acc(p_1, p_2)}{Acc(p_1) + Acc(p_2) - Acc(p_1, p_2) + 1} \tag{1}$$

The $+1$ was added to prevent division by zero and to punish low $Acc(p_1, p_2)$ values as these might occur by chance. This means that $Sim(p_1, p_2) \in [0, 1)$. We further added a threshold $c_S \in \mathbb{N}_+$ and set Sim to zero if $Acc(p_1, p_2) < c_S$.

To cluster the pages we used agglomerative hierarchical clustering [10]. Hierarchical clustering was chosen since the amount of clusters was unknown. To decide whether to merge two clusters $A, B \subset P$ we used the Unweighted Pair Group Method with Arithmetic Mean [12], e.g.

$$\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} Sim(a, b) \tag{2}$$

As a threshold we experimented with different values. If the threshold is too small, a lot of pages are clustered, but the clusters might not actually be meaningful. If the threshold is too large, only strong (and very obvious) clusters are considered. This will then result in safer recommendations, but these might not be as interesting for the user.

Clustering is the slowest part of our prototype and the implementation can still be improved upon. The complexity of the clustering technique we implemented is known to be $O(n^3)$. Since the computation is done offline, this is still acceptable.

Scoring Clusters. Not all clusters are as meaningful as others. A cluster that has been accessed by most users is obviously not very meaningful when distinguishing users. We measured the importance $Score(A) \in [0, 1)$ of a cluster A by the percent of users that had accessed at least one cluster, but had not accessed this cluster.

Similar users. To generate recommendations we used Collaborative Filtering. This meant we first had to find similar users. To compute the similarity between two users we combined a number of different attributes.

For every user U_i we stored each cluster the user accessed and the first access to this cluster. This gave us a sorted list L_i of accessed clusters. We then only considered those users for recommendation that had at least accessed a certain

[8] The user time-line is the sequence of page accesses for a particular user over time.

threshold of clusters. The factors that we considered important for the similarity between two users U_i, U_j were further 1) the clusters C_{ij} that U_i and U_j had both accessed 2) the total amount of clusters L_i that a user U_i had accessed 3) the score of a clusters and 4) the positions $O_i(c), O_j(c)$ of a cluster c in the lists O_i, O_j of first access that U_i and U_j made to elements of C_{ij}, e.g. "$C_{ij} \cap L_i$" and "$C_{ij} \cap L_j$". This led to the formula

$$Sim(U_i, U_j) = \left(\frac{1}{\#C_{ij}} \cdot \sum_{c \in C_{ij}} \frac{Score(c)}{\sqrt{|O_i(c) - O_j(c)| + 1}} \right) \cdot \frac{\#{C_{ij}}^2}{\#L_i \cdot \#L_j} \cdot f(i,j) \tag{3}$$

with

$$f(i,j) = max\left\{0, \ \frac{-4}{\#C_{ij} + 1} + 1\right\} \tag{4}$$

to compute the similarity between two users. The first part of (3) accounts for the similarity in the sequence of common accesses to clusters. A visualization of the similarity between two cluster positions can be seen in Fig. 4 (1). It was chosen since a similarity very close to zero was not desirable. The second part considers the amount of clusters that are actually overlapping relative to the ones that could overlap. And the third part punishes if the total amount of common clusters is low. A visualization of this can be seen in Fig. 4 (2).

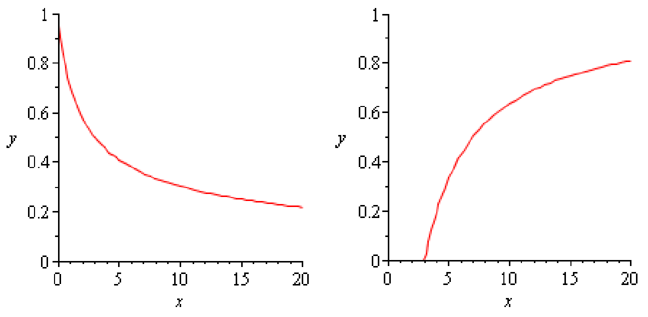


Fig. 4. Visualization of the distance function $1/\sqrt{x+1}$ between two cluster (left) and visualization of the penalty function $max\{0, -4/(x+1)+1\}$ for small cluster overlap (right).

Let us consider an example to better understand the formula: Assuming we have the sequences $L_1 = [A, B, C, D, X]$ and $L_2 = [B, Y, C, A, D]$. Then we compute $\#C_{12} = 4$ and $U_1 = [A, B, C, D]$ and $U_2 = [B, C, A, D]$. For the first part of $Sim(U_1, U_2)$ we get

$$\frac{1}{4} \cdot \left(\frac{Score(A)}{\sqrt{|1-3|+1}} + \frac{Score(B)}{\sqrt{|2-1|+1}} + \frac{Score(C)}{\sqrt{|3-2|+1}} + \frac{Score(D)}{\sqrt{|4-4|+1}} \right) \tag{5}$$

$$= \frac{1}{4} \cdot \left(\frac{Score(A)}{\sqrt{3}} + \frac{Score(B)}{\sqrt{2}} + \frac{Score(C)}{\sqrt{2}} + \frac{Score(D)}{\sqrt{1}} \right) \tag{6}$$

We notice that the position of D matches and hence this part gets the highest weight. For the second part we get

$$\frac{4^2}{5 \cdot 5} = \frac{16}{25} = 0.64 \tag{7}$$

which reflects that the overlap for the cluster access is relatively large. The third part computes to

$$max\left\{0,\ \frac{-4}{(4+1)+1}\right\} = 0.2 \tag{8}$$

This is a relatively small value since there are only four clusters overlapping. If there were fewer clusters the value would even be zero. The meaning of this is that a recommendations is only useful if it is based on a suitable large amount of comparisons.

In the next step we generated a list of nearest neighbors (NN) for each, user using a threshold to prune distant neighbors. Recommendations were only generated if a certain amount of neighbors was found. Clusters were recommended when a certain percentage of NN had accessed a cluster, but the user itself did not access it. For our evaluation we only considered the top recommendation most NN agreed on.

4.3 Evaluating the Recommendations

While evaluating the system by asking the user directly would have been the preferred way, we did not have the time to implement and run this method.

Since we recommended clusters and the recommendations were based on long-term user actions, having an expert analyze the result sets for "quality" would have been a difficult and tedious work. Some recommendations were based on more than a hundred page accesses.

Instead we split the page visits into train and test sets. To ensure that additional pages for each user are found in the test set, we split our data by removed the last few sessions of each user. The exact parameters we used, the results we found and an interpretation of them is given in the following section. To fine tune the system, we split the train data again.

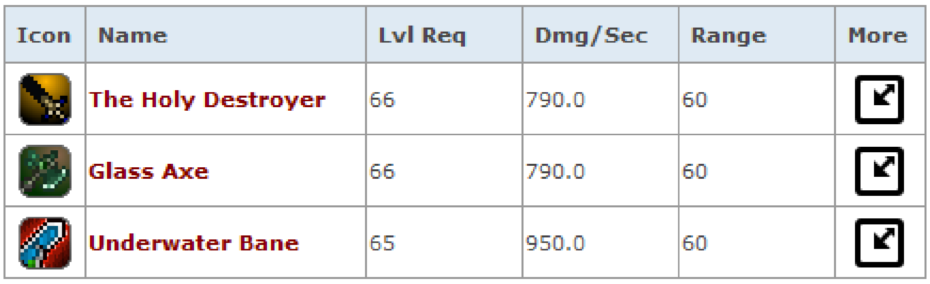
5 Evaluation

To evaluate the system we used access data logs from a MMORPG database website [11]. The motivation here was that progressing users will access the pages that are relevant to their character. For example a user who progressed with his character (e.g. "gained a level"), might be looking for a new weapon or a new skill that his character can now use. Or he might be interested in knowing what actions his character needs to take to obtain a skill or weapon.

Like the graphic software that was discussed before, this is an example of a domain our RS can be applied to. The connection can easily be drawn: In the graphic program the user acquires knowledge, while in the MMORPG the character of the player progresses (”acquires items and properties”).

There are a number of decisions that a user can take when evolving their character and hence the sequence of accesses and the accesses itself can be very different, while groups of similar behaving users also exist.

A typical cluster that was generated by our system looked like Fig. 5. One can easily see why these items are related: They are all weapons with similar attributes. Other cluster were formed because they had similar names and were retrieved in the same search.



Icon	Name	Lvl Req	Dmg/Sec	Range	More
	The Holy Destroyer	66	790.0	60	↙
	Glass Axe	66	790.0	60	↙
	Underwater Bane	65	950.0	60	↙

Fig. 5. A typical cluster retrieved by our prototype. An access was logged when the user clicked on the ”more” button

We fine-tuned several parameters to match our test database and applied some filters to reduce the noise (e.g. characters that did not progress were excluded from the data). From two month of data collection we got 13893 page hits this way. After pruning the last two sessions of all users that had accessed more than 20 pages, we were left with 11391 page to generate our recommendations.

When clustering we got 49 clusters with a total of 146 pages. Out of all users 121 had accessed more than 50 pages, but only 36 users had accessed 8 or more clusters (which was our threshold to consider them). Not for all 36 users recommendations were generated, since for some there were not enough NN.

To study the impact of considering access sequences, we generated recommendations with and without removing the first part of the user similarity formula (see formula 3), e.g. taking the distance between the clusters in the sequences into account and not doing so. The results can be seen in Figure 6. The rows show the results for considering the top 1,2 and 3 recommendations. Users that didn’t have enough recommendations or did not access clusters in the test data were excluded. Precision P and Recall R were computed by

$$P(L) = \frac{1}{\#U} \sum_{u \in U} \frac{\#(L(u) \cap T(u))}{\#L(u)} \tag{9}$$

$$R(L) = \frac{1}{\#U} \sum_{u \in U} \frac{\#(L(u) \cap T(u))}{\#T(u)} \tag{10}$$

where U are all the users that have recommendations, $L(u)$ are the recommended clusters for user u and $T(u)$ are the clusters that u accessed in the test data and not in the training data.

One can see that both tables the F1 Score does not change too much. Precision decreases and Recall increases in both experiments as was expected. In the first table the F1 Score is always larger.

Recom./User	User	Precision	Recall	F1 Score
1	22	0.636	0.337	0.441
2	16	0.469	0.424	0.445
3	7	0.381	0.440	0.409

Recom./User	User	Precision	Recall	F1 Score
1	22	0.455	0.187	0.265
2	15	0.333	0.274	0.301
3	8	0.250	0.344	0.289

Fig. 6. Values generated using the unchanged similarity formula (top) and values generated when not considering the sequence of accesses, e.g. removing the first part of the similarity formula (bottom). Values are rounded to three decimal places.

When manually looking through the recommended clusters we observed that a lot of them were "safe" recommendations, e.g. items that every user would be using at a certain point. Other recommendations were based on the class of character the user had chosen (a sword for a warrior, a wand for a magician).

We then changed different parameters in our prototype, like the necessary similarity for NN to be considered. The difference between the confirmed recommendations between "sequence checking" and "intersection checking" was not always as significant, but in general the sequence checking performed better.

Since we only used one specific dataset and the set of recommendations was not very large, we have to be careful when interpreting the result. It seems to suggest that recommendations can benefit from analyzing the sequence of user actions in certain situations.

In the future we would like to evaluate the system by presenting the users with recommended clusters and letting them evaluate the usefulness of the shown clusters. This will allow for a more sophisticated evaluation of the system: Even though the user visited a cluster that we recommended, this doesn't mean he would be happy to see the recommendation. The cluster might have been an obvious choice or only useful together with other pages that were not recommended.

6 Conclusion and future work

We have shown how to implement effective, lightweight user access logging and utilize the information to build a recommender system. Our prototype and test-data then illustrated that sequence mining of long term user behavior can be useful when generating or refining recommendations.

While the prototype that we have built requires a lot of data to work properly and the generated recommendations are sparse, the required information is easy to collect. It would be nice to analyze the capabilities of this system further and test it with larger and different data sets. More conclusive results are also required to tune the prototype further and to identify different impacting parameters in different datasets.

The prototype still requires a lot of fine tuning before it works with an existing system. It could highly benefit from automatic parameter generation, which would allow for much faster integration.

An interesting area for research is when to suggest recommendations. Even if we are able to understand and predict the long term user behavior, this doesn't mean the user is already ready for the recommendation. He might need to progress before certain content becomes useful. In our example recommending a weapon with high level requirements to a low level player is not necessarily helpful, even though the player might eventually end up using this weapon.

References

1. Choochart Haruechaiyasak, Mei-Ling Shyu, Shu-Ching Chen: A Data Mining Framework for Building A Web-Page Recommender System. IRI 2004: 357-362
2. Francesco Ricci; Lior Rokach; Bracha Shapira; Paul B. Kantor, ed. (2011). Recommender Systems Handbook. ISBN 978-0-387-85819-7.
3. Dey, A. K.: Providing Architectural Support for Building Context-Aware Applications. Ph.D. thesis, College of Computing, Georgia Institute of Technology (2000)
4. Mark Setten, Stanislav Pokraev, Johan Koolwaaij: Adaptive Hypermedia and Adaptive Web-Based Systems In Adaptive Hypermedia and Adaptive Web-Based Systems, Vol. 3137 (2004), pp. 515-548, doi:10.1007/978-3-540-27780-4_27
5. Negar Hariri, Bemshad Mobasher, Robin Burke: Context-Aware Music Recommendations Based on Latent Topic Sequential Patterns. *Manuscript submitted for publication.*
6. Nicolas Jones and Pearl Pu. User Acceptance Issues in Music Recommender Systems. Technical report, EPFL, 2008.
7. Niwa, S.; Doi, T.; Honiden, S.; , "Web Page Recommender System based on Folksonomy Mining for ITNG &#146;06 Submissions," Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on , vol., no., pp.388-393, 10-12 April 2006
8. P. Maes, "Agents that reduce work and information overload," Communications of the ACM, 37(7):30-40, 1994.
9. Schafer, J.B., (2005), "The Application of Data-Mining to Recommender Systems," in Encyclopedia of Data Warehousing and Mining, Wang, J. (Editor). Information Science Publishing.
10. Wikipedia contributors. "Hierarchical clustering." Wikipedia, The Free Encyclopedia, 9 May. 2012. Web. 5 Jun. 2012.
11. Wikipedia contributors. "Massively multiplayer online role-playing game." Wikipedia, The Free Encyclopedia, 6 Jun. 2012. Web. 8 Jun. 2012.
12. Wikipedia contributors. "UPGMA." Wikipedia, The Free Encyclopedia, 10 Jan. 2012. Web. 5 Jun. 2012.