An Architecture for Mining Massive Web Logs with Experiments*

András A. Benczúr[1 2] Károly Csalogány[2] András Lukács[1 2]
Balázs Rácz[3 1] Csaba Sidló[2] Máté Uher[1] László Végh[2]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences
MTA SZTAKI, Kende u. 13-17., 1111 Budapest, Hungary
[2] Eötvös University, Pázmány P. stny. 1/c, 1117 Budapest, Hungary
[3] Budapest University of Technology and Economics, Egry J. u. 1., 1111 Budapest, Hungary
{benczur,cskaresz}@ilab.sztaki.hu, alukacs@sztaki.hu,
bracz@math.bme.hu, scsi@ludens.elte.hu, matus@cs.elte.hu,
veghal@eotvos.elte.hu
www.ilab.sztaki.hu/websearch

Abstract. We introduce an experimental web log mining architecture with advanced storage and data mining components. The aim of the system is to give a flexible base for web usage mining of large scale Internet sites. We present experiments over logs of the largest Hungarian Web portal [origo] (www.origo.hu) that among others provides online news and magazines, community pages, software downloads, free email as well as a search engine. The portal has a size of over 130,000 pages and receives 6,500,000 HTML hits on a typical workday, producing 2.5 GB of raw server logs that remains a size of 400 MB per day even after cleaning and preprocessing, thus overrunning the storage space capabilities of typical commercial systems.
As the results of our experiments over the [origo] site we present certain distributions related to the number of hits and sessions, some of which is somewhat surprising and different from an expected simple power law distribution. The problem of the too many and redundant frequent sequences of web log data is investigated. We describe our method to characterize user navigation patterns by comparing with choices of a memoryless Markov process. Finally we demonstrate the effectiveness of our clustering technique based on a mixture of singular value decomposition and refined heuristics.

1 Introduction

Today there is an emerging demand of Internet and network related Service and Content Providers to collect the valuable service usage data and process it using data mining methods to answer questions regarding security, service improvement or financial issues. Medium to large sized companies can easily produce log files of extreme sizes (up to hundreds of gigabytes per month). Collecting, keeping these log data sets in a storage-efficient and easily accessible way suitable for direct processing by several types of data mining algorithms is a challenging problem.

* Support from NKFP-2/0017/2002 project Data Riddle and OTKA and AKP grants

We present an architecture for web log mining designed for the largest Hungarian Web portal [origo] (www.origo.hu). The site [origo] that among others provides on-line news and magazines, community pages, software downloads, free email as well as a search engine. The portal has a size of over 130,000 pages and receives 6,500,000 HTML hits on a typical workday, producing 2.5 GB of raw server logs that remains a size of 400 MB per day even after cleaning and preprocessing, thus overrunning the storage space and analysis capabilities of typical commercial systems.

The system includes several modules to provide a complete service for web log handling and analysis. The modules for collecting and filtering provide the preprocessed data for the advanced modules of a refined storing system and an OLAP-like statistic unit. The statistic system provides a fast on-line service for basic statistical aggregation and detailed short-term queries.

Module of storing provide a base for long-term *storage* in a form appropriate for direct processing by *data mining* algorithms. A novel preprocessing and high density compression technique for log files is used in the module.

To demonstrate the usability of our architecture for web log handling we introduce the results of our first experiments over the site [origo]. The experiments include overall statistics, identifying important sequences in the click pattern, and modelling user behavior.

Certain distributions related to the number of hits and sessions are presented, some of which is somewhat surprising and different from an expected simple *power law* distribution.

For analyzing click-streams first we apply large-scale *frequent sequence* analysis to the data sets. The problem of setting the minimum support and the problem of interpreting the output are investigated in details. We will briefly concern the usefulness of the *closed frequent itemset* concept and *association rules*.

Another approach models navigational behavior by assuming that the user has a limited finite memory of past pages and the choice of the next hyperlink is more or less determined by the last few pages visited. We consider the support of frequent (strict referrer) sequences. We compare their support to the postfix shorter by one element, we measure the *bias* from a memoryless *Markov process*.

As an example for *clustering* of massive data, we used the web log data to construct the user-document matrix for clustering of the users and the also the documents of the site [origo]. Our main tool for clustering based on linear algebra, the *singular value decomposition*. The quality of clusterings were demonstrated in the following way. In case of document-clustering we compared the results with the existing topics hierarchy of the site in question. A cluster of users, on the other hand, can be represented by the documents downloaded by the members of the cluster.

This paper is organized as follows: In Section 2 we describe the main properties and tools of the system for collecting, storing and analyzing web log data. Section 3 contains our experiments on the web log data collected from the site [origo]. Section 3.1 contains overall statistics about the number of hits and sessions. Our observations on click-stream analysis of web logs can be found in section 3.2 (finding frequent sets of downloaded pages) and in section 3.4 (Markov and non-Markov properties of the

clicking process). In section 3.5 we introduce our method for clustering users with linear algebraic tools (SVD).

2 Goals and the architecture

The [origo], as a market leader portal in Hungary, records over 6.5 millions of hits per day. This means a data volume of 2500 MByte per day text file logs that results in over 80 GByte data per month.

In the current solution, weblogs are processed at each http frontend server and each day separately. The raw log files are compressed via the *gzip* utility, and stored on tapes. Although zipped files require less storage capacity, reusing archives of a size of 21 GBytes per month remains infeasible. Currently the only means of analyzing weblogs consist of standalone fixed report generators and Perl scripts, with the drawbacks of limited flexibility and usability. They can answer only a limited number of questions by pre-generated reports, and customization typically results in rewriting the utility.

Our aim is to design and build a data warehouse based solution offering a flexible base for ad-hoc analysis, OLAP-like queries and data mining components. Long term storage of log files is also included in the database architecture. Notice that an OLAP architecture design for weblog mining is a challenging task; the statement given in 1998 [20] remains valid: “due to the important size and the ever exploding nature of web log files, the construction of multidimensional data cubes necessary for on-line analytical processing and knowledge discovery, is very demanding and time consuming”.

A great deal of previous works handle the special task of web usage mining and the analysis of the so-called clickstream data. We design the system on the base of the Click Fact Table model. In his latest book [10] Ralph Kimball describes the models of Click Fact Table and Session Fact Table considered now as an industrial standard. These models provide a good base for OLAP-like ad-hoc queries and for flexible statistical reports. The prime aim of the design is to produce a (Fact Table Model based) clickstream data mart system enhanced with integrated data backup and data mining modules. The logical architecture of the system is depicted on Figure 1.

The main flow of the clickstream data is shown on the figure. The prime data sources are the web server log files. They contain all the requests the web servers got. A hit has standard attributes like client host, request time, request code, http request (containing the page url), referer, agent identification string, and contains session identification cookie as well. The cookie collection in the log files simplifies the processing of session related information. The session identification by the usual Common Log Format or Extended Common Log Format should be less accurate, handling cookies makes individual users more traceable [4].

Additional data sources include the editorial system of the [origo] portal describing the web page hierarchy and the page properties.

The first step is to preprocess the raw log data. We decided to handle clickstream data with the granularity of the page hits as the finest data granularity. Thus we reduce the amount of data and introduce an initial filtering stage that does as well cleaning of the raw data. In this step we also deal with the task of making data more understandable, completing missing information, doing reverse DNS lookups, identifying session-

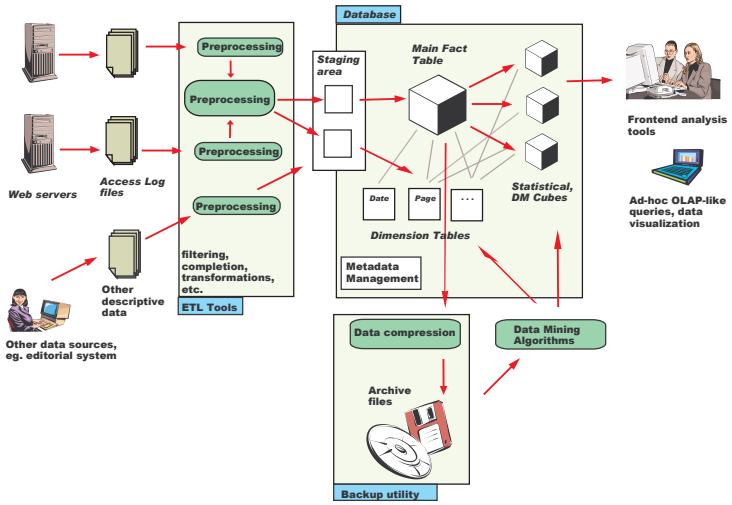

Fig. 1. The architecture of the web log analyzer

level attributes, etc. After the first phase the data is ready for loading into the database, through a staging area.

The database schema is designed according to the relational click fact table model. The dimensions are designed to handle all the usual descriptive information as well as the extra information from the editorial system. The central fact table contains all the page hits. During the construction of the schema we have paid attention to the huge data volumes. It was necessary to evaluate the element count and element count growth of the dimensions. Processing a sample log helped to determine the problematic dimensions, and to estimate the fact table size. If some of the dimensions grow too large or too fast, we had to make a compromise: smaller dimension granularity means less analysis flexibility, but it can keep the data amount on a manageable level. Another compromise is that the central fact table contains only a short time period (eg. two weeks or a month) of data on page hit level.

To enhance the analytical performance of the system we build statistical data cubes that contain aggregated data from the main fact table. The main idea is to store high resolution data only for a limited time period. The statistical cubes contain data for a long time period, but with less dimensions or filtered data. The determination of these aggregation conditions is based on the needs of analysis and on the evaluation of the cube size and the cube data density. We design a metadata-structure for data propagation to the statistical cubes. Appropriate descriptive metadata on the cubes can be useful for developing a general OLAP-like analyser tool as well.

For the long period web log storage we use a novel compression module. The module is designed to store all kind of log files efficiently, compress the log data with a compression rate exceeding that of the general data compression utilities by using par-

ticular properties of the log file structure. The module receives a general, descriptive meta-structure for log data (field type definitions and other field properties) in order to compress the data by selecting the appropriate methods from a large set of compression methods, like gamma-code, delta-code, Huffman-code etc. As a final step the compressed data is archived.

The compression utility gets data from the database, from the main click fact table, and can restore this data later.

The compression module has a number of desirable properties. Decompression runs in linear time and may begin by almost random access in the middle of the compressed file. Accidental errors of the compressed data remain local. As the compression is made separately on fields, and the decompression results a parsed memory image of the data set, the compressed data is directly ready for data mining tools.

Data mining algorithms such as association rule and sequential pattern mining or clustering can be implemented using the compressed, long range archive data as special, standalone applications. The clickstream database provides a good opportunity to store and analyze the results of these methods. The goal is to make data mining results accessible in the OLAP-like query framework, to make this knowledge visualizable, easy to use. Hence we design data mining cubes, data-mining dimensions or dimension hierarchies like frequent page sequences or user clusters. Using these new elements in data collection and in the analytical framework can make data mining results more understandable, the OLAP-like ad-hoc queries more flexible.

As a future work we plan to implement session fact table-like components, to enhance the analysis flexibility. The general framework and the experiences through building the weblog architecture would give us a good base to design other log file processing modules as well, like mailserver logs or search engine logs.

3 Experiments

3.1 Download and session statistics

For the sake of illustration we describe the October 2001 web logs (current figures have roughly doubled since then). The total number of unique IP addresses was 875,651, having a total of 18,795,106 requests which can be grouped into 5,702,643 sessions. (There should be at least 30 minutes between two sessions.) The site contains 77,133 unique pages which have been downloaded at least at once.

Pages over the [origo] site consist of a dynamically generated frame with fixed elements (links to other topics and services) as well as links to the hot news of the current and other topics. Page URL remain valid forever and contain the same main content with the current frame generated at request; old pages are, however, after a while only reachable by searching the archive and not by following hyperlinks.

In Figure 6 we show the distribution of page access count, with the count on the horizontal axis and the corresponding density in percentages on the vertical axis. The distribution is Zipf-like with a $\gamma = -1.2$ exponent. The dashed line is the fitted line using least squares method.

Figure 7 shows a similar graph for the number of pages visited by a user. The distribution appears to be a combination of two Zipfians, in the lower range with a larger exponent $\gamma = -1.5$, while in the higher range a smaller $\gamma = -2.5$.

Next we identify user sessions by starting a new session after 30 minutes of idle time. In Figure 8 we show the distribution of session length with density in percentages over the vertical axis and a logarithmic scale on the horizontal axis.

The experiment shows that in the Origo web log, the average sesion length is about 7 minutes with 1919 seconds of deviation. The median session length is 0 second, which means that half of the session contains one request, and less than 50% of them contains more than one.

The distribution of time elapsed between adjacent hits by a user is seen in Figure 9. The graph consists of several peaks. The first one corresponds to redirects. The next peak at 14 seconds might correspond to real user intention: after seeing the page, quickly selecting the desired link. The peak at 64 seconds might then characterize users who read the page content before continuing. Peaks at 1745 and 3486 seconds are explained by the refresh meta-tag of pages (`www.origo.hu/index.html` has a refresh meta tag of 1740 second). Another maxima appear at multiples of days that may describe users who read [origo] news (or perhaps have Internet access) at a regular time of the day. The average time gap between two request is about 1 day, and the median time gap is about 10 minutes. In other words half of the visits come more than 10 minutes after the last access, and half of them come within 10 minutes.

3.2 Clickstream analysis

Click stream analysis has a vast literature; approaches falling into indentifying important sequences in the click pattern and modeling user behavior. A very early approach [6] utilizes association rule mining [2]; frequent sequences [12,17] and longest repeating subsequences [15] can be considered as variants of this approach.

Another approach models navigational behavior by assuming that the user has a limited finite memory of past pages and the choice of the next hyperlink is more or less determined by the last few pages visited. Various studies investigate how well length three [18] or longer [14] sequences predict user behavior, comparing patters from weblogs with Markovian processes of certain order or consider hidden Markov models [19].

First we describe our frequent sequence mining experiments including algorithmic issues and setting the minimum support. In section 3.3 we turn to the challenge of handling the huge size of frequent sequences that makes it impossible for a human to interpret the results. We briefly concern some useful space-saver specialization of the basic definition, as well as the usefulness of the 'closed frequent itemset' concept as well as those of association rules. Finally in Section 3.4 we sketch a novel approach to filter the relevance of longer sequences by measuring how much of their first element is "remembered" when the user makes the last selection.

While the output of frequent sequence mining gives meaningful insight of navigational patterns, it is typically used as the first step of *association rule discovery*. An association rule $X \rightarrow Y$ holds with *support* s and *confidence* c, or (c, s) in short, if s percentage of all the users visited all pages in set X, and c percentage of these users

also visited all pages in set Y. Usually we set a minimum constraint for both parameters: min_supp and min_conf respectively to select only the most relevant rules of the dataset.

Challenge: infeasible Apriori data structures We find that algorithm *Apriori* [2], a descendant of algorithm *AIS* [1] using ideas from [11], is infeasible for web log sequence mining with low minimum support and *heavy users* with a large number of page visits. We overcame this difficulty by designing a new internal data structure that holds considerable supplemental information aside the trie, and redesigning the counting process around this data structure. Due to space limitations we omit discussion and use the results of the modified algorithm.

The key problem with a textbook implementation of algorithm *Apriori* is that a combinatorial explosion is observed starting from the 5-element frequent sequences. Since on the average 50–60% of the candidates proved to be frequent sequences themselves, the explosion is not explained by the time spent on the counting of non-frequent candidate support. A more detailed analysis showed that over 85% of the time was spent on 20% of the log file which belonged to the accesses of *heavy users* with a large number of page downloads.

The combinatorial explosion for medium-size sequences caused by the heavy users can be explained on a theoretical base. The counting process of Apriori, while examining the sequence of one user, for each element of the sequence uses as many trie descend-lookups as the number of perviously visited trie nodes. With short user sequences (like market baskets at the time of the development of Apriori) this poses no problems at all. However, web logs can hold user sequences of up to several thousands of accesses, making the counting process infeasible for web logs.

pass	count of		run time		
	candidates	frequents	(new)	(original)	(or., 80%)
1	14718	358	-	0:00:04	-
2	128164	4030	0:11:12	0:00:45	0:00:25
3	62452	13801	0:09:07	0:07:09	0:01:36
4	62324	27017	0:15:13	0:25:09	0:04:03
5	72050	37939	0:18:31	0:59:04	0:08:20
6	74273	42521	0:22:50	1:43:52	0:14:09
7	70972	43465	0:24:46	2:26:37	0:18:55
8	70541	45373	0:20:05	3:05:05	0:22:30
9	75082				0:24:57

Table 1. Comparison of encoding and decoding time on the sample web logs

The statistics of a sample run are summarized in Table 1. Run times were measured on a 1.4 GHz AMD Athlon based machine with 3.5 GB of RAM running linux. The minimum support was set according to the next subsection so that there are numerous nontrivial frequent documents.

Setting the minimum support The key problem of mining a news site's web log is that the data set is extremely heterogenous. The most relevant difference is between the static pages (main page, directory indices) and the content (news, articles) themselves. **The support of a static page is higher by one to three orders of magnitude.** This can be explained by on one side the aging of the news and articles, on the other side the recurrence of the index pages in one user's clickstream.

This has a dramatic impact on the reasonable min_supp values. The main page gets 28% of the hits, but there are only 8 documents with over 1% of support. The 50 documents with over 0.1% support contain only 3 content page, everything else is either a directory index or some ad support page. On the other hand to get a fair coverage of the data set by frequent documents one has to set the minimum support to as low as 0.01%. This is quite far from the textbook or synthetic dataset definition of 'frequent'.

The impact of the minimum support on the number of frequent documents and the total hits collected by them can be seen on figure 2.

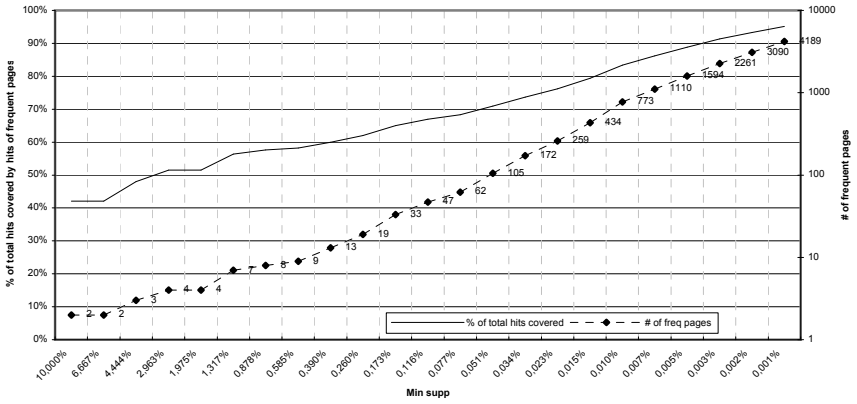

Fig. 2. Number of frequent pages for different min_supp values

3.3 Challenge: frequent sequences are too frequent

As seen in Figure 3, there are just *too many* frequent sequences for a human to read through in hope of understanding the data set. Also, the information in frequent sequences is highly redundant. Due to the low min_supp value, the output is flooded with all combination of the directory index pages. We have to either manually filter the output or look for better concepts than plain frequent sequences.

First we give ideas of alternative definitions that may reduce the number of frequent sequences. A novel relevance filtering method is descibed in the next subsection.

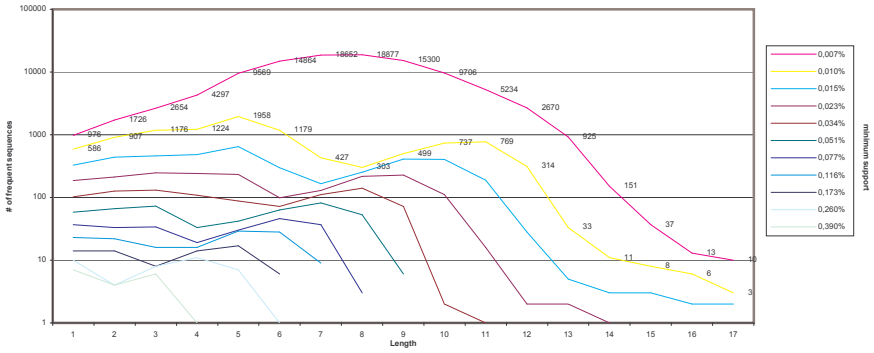

Fig. 3. Number of frequent sequences for different min_supp values
Note that the minimum support values are also logarithmic (with a quotient of 1.5), so the equidistant partition of column '1' shows the power law of the 1 element frequent sequences (e.g. frequent documents).

closed frequent itemsets A frequent itemset X is *closed* if when taking all users who downloaded all pages in X we cannot find any more pages all of them downloaded too. This is equivalent to the following: we cannot find a $Y \supset X$ larger set which had the same support. A third equivalent definition is that a frequent itenset is closed if it does not stand on the left hand side of a 100% association rule. In theoretical studies this is a very important concept, as there are examples where the count of non-closed freqent itemsets is exponential in the count of closed frequent itemsets. However, in real world datasets *there are no 100% association rules valid*[4], so by the the third definition all frequent itemsets are closed.

weakening closedness We could weaken the requirement of the association rule being 100%. If we do this to the extreme and filter out all frequent itemsets which are on the left hand side of an association rule of any confidence, then we get the *maximal* frequent itemsets (or sequences). This approach can reduce the number of frequent sequences by 50-70%, but still cannot reach the human understandable size. On the other hand, this gives us an idea on how many association rules hold (without considering the possibility of a certain itemset to be on the left hand side of more association rules).

referer-strict sequences We get a more refined and weblog-specific definition if we require the user download equence to contain a frequent sequence with each element's referer being the previous element of the frequent sequence. PLease note that this way we still note require the frequent sequence to be a consecutive subsequence of the containing download sequence of the user. We allow 'browse loops', e.g. when the user

[4] ...usually, of course. Whenever there is the slightest applicability of an independence-based probabilistic model on the data source (for example where users actions are independent of each other) the probability of an itemset to be closed is exponentially small in the support of it. As we set a minimum constraint on the support, this is practically zero.

turns into a dead end or looses his way through the site navigation and returns to a previously visited page and continues on the path of the frequent sequence.

3.4 Memory of navigation

Both Markovian and frequent sequence mining approaches suffer from time and storage limitations, as noted for the former approach in [16,14]. When mining frequent sequences we set the minimum support threshold and receive, as output, all sequences visited by at least the prescribed percent of users. By setting the threshold high, however, we get too little and obvious information, those related to visiting various index.html pages. And as we decrease the threshold we very quickly face an exceedingly large amount of output that needs data mining within the data mining output such as relevance measuring [5], clustering [8] or base selection [21].

We suggest a different combined approach. As suggested in [12], we consider the support of frequent (strict referrer) sequences. We compare their support to the postfix shorter by one element, thus measuring how much information is remembered of the first Web page when making the decision to visit the k-th. Formally stated, we measure the *bias* from a memoryless Markov (or order $k-1$ Markov) process,

$$r(x_1, x_2, \ldots, x_k) = \Pr(x_k|x_1, \ldots, x_{k-1}) - \Pr(x_k|x_2, \ldots, x_{k-1}) \cdot \Pr(x_2, \ldots, x_{k-1}|x_1)\,.$$

For example consider the frequent strict referrer sequence

```
www.origo.hu/szoftverbazis/jatekok/akcio/index.html
www.origo.hu/szoftverbazis/jatekok/akcio
www.origo.hu/szoftverbazis/jatekok/akcio
```

with value $r = 0.22$, meaning that visiting the action game index page makes it more likely to follow a second game page. The meaning of such a rule needs careful understanding. The user is very unlikely jump directly to a particular game page, unless using search. The likely entry point is thus the index—or another game page. Hence this rule in fact means that it is *less* likely that the user will visit a third game page. As another example,

```
www.origo.hu/search/index.html
www.origo.hu/search/index.html
www.origo.hu/index.html
```

has a negative bias -0.088, meaning that after subsequent searches we are less likely to go back to the index page and continue with reading news articles—we are likely to have a target in mind and are less intended for browsing the site. Pages

```
www.origo.hu/adsminisites/usa
```

consist of image sequences; all such sequences are frequent of length up to 12 and have positive bias that exponentially diminishes (-0.07, 0.07, 0.004, 0.013, 0.013, 0.011, 0.010, 0.0087, 0.0082, 0.0072, 0.0067). Note that the length three sequence has negative bias. The length two sequence has a positive bias for continuation with another world

news article, these two observations together meaning that a large number of users quit after seeing two pictures, however those who continue are even more likely to continue as they see more and more. These observations, for the example of the above image sequence, explain a large tail in session length and may partly explain the Zipfian distribution observed in Section 3.1.

The method of selecting sequences with large (positive or negative) bias filters out most of the frequent sequences. For example we may safely discard rules such as "many users immediately quit reading world news and turn to local news",

```
www.origo.hu/index.html
www.origo.hu/nagyvilag (world news)
www.origo.hu/itthon (home news)
```

since the bias 0.0003 means a purely random bahavior that may equally likely happen if they have already seen a number of news, regardless of topical categories.

As indicated by Figure 4, the bias of frequent sequences decays exponentially as the length of sequences increases. Sequences with positive bias appear to have a larger tail, however the maximum in this case is attained at the above sequence of images and is characteristic to such sequences.

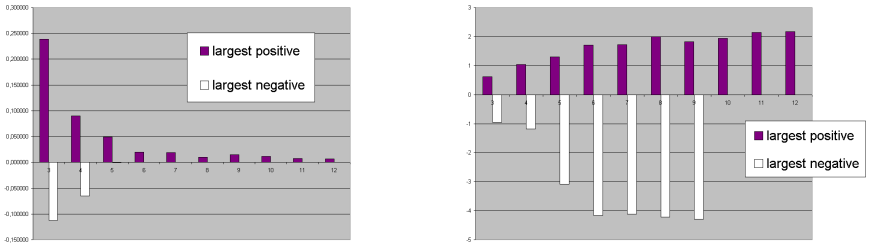

Fig. 4. Value of the largest positive and negative bias for a sequence of length 3 to 12, on a regular (left) and logarithmic (rigth) scale

3.5 Clustering

We use clustering by singular value decomposition, a method well described in the literature when applied to graph partitioning [3,13,9] and latent semantic indexing over the document-word matrix [7].

We adapt the algorithm described in [13] for the document-user matrix instead of a graph adjacency matrix. The algorithm first computes the first k singular vectors and projects users or, respectively, the documents into a k-dimensional space. Over the reduced size projection we apply the k-means clustering algorithm. While singular vector computation is expensive, few vectors yield clusters of poor quality. Typically for k clusters at least k dimensions are suggested; however even projecting to higher dimensions we often obtain a single huge cluster.

Since k-means often fails to produce balanced size clusters, we use two heuristics to produce a "reasonable" split of the data. First, index pages with very large number of visits often keep points close and result in huge clusters; these elements can safely be discarded when found. Second, by starting with a certain dimension k, we continue computing increasing number of singular vectors until we reach the desired split in k-means.

In case of document-clustering we compared the results with the prefixes of the URLs (the domainname and the first tag of the path). A cluster is homogeneous if a few prefixes dominate the cluster. Figure 5 shows an increasing homogenity as we move down in the hierarchy. Each color in the pie charts corresponds to an URL prefix downloaded by the users in the cluster (e.g. software `www.origo.hu/szoftverbazis` is red, sport `www.origo.hu/sport` is green).

A cluster of users, on the other hand, can be represented by the documents downloaded by the members of the cluster. These documents shows the interest of the set of users. On the bottom of Figure 5 we see the homogenity increasing down in the hierarchy, similar to document clusters. Notice however that clusters of users are less homogeneous, since a typical user is interested in several topics and is easily distracted—nevertheless the prime goal of a portal site design is to make users spend more time there.

Acknowledgments

Our thanks to the team at Axelero Inc., especially to István Szakadát, Gábor Kiss and Tibor Takács.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.
3. C. Alpert and S. Yao. Spectral partitioning: The more eigenvectors, the better, 1994.
4. J. Andersen, A. Giversen, A. H. Jensen, R. S. Larsen, T. B. Pedersen, and J. Skyt. Analyzing clickstreams using subsessions. In *Proceedings of the third ACM international workshop on Data warehousing and OLAP*, pages 25–32. ACM Press, 2000.
5. S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. pages 265–276, 1997.
6. M. S. Chen, J. S. Park, and P. S. Yu. Data mining for path traversal patterns in a web environment. In *Sixteenth International Conference on Distributed Computing Systems*, pages 385–392, 1996.
7. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

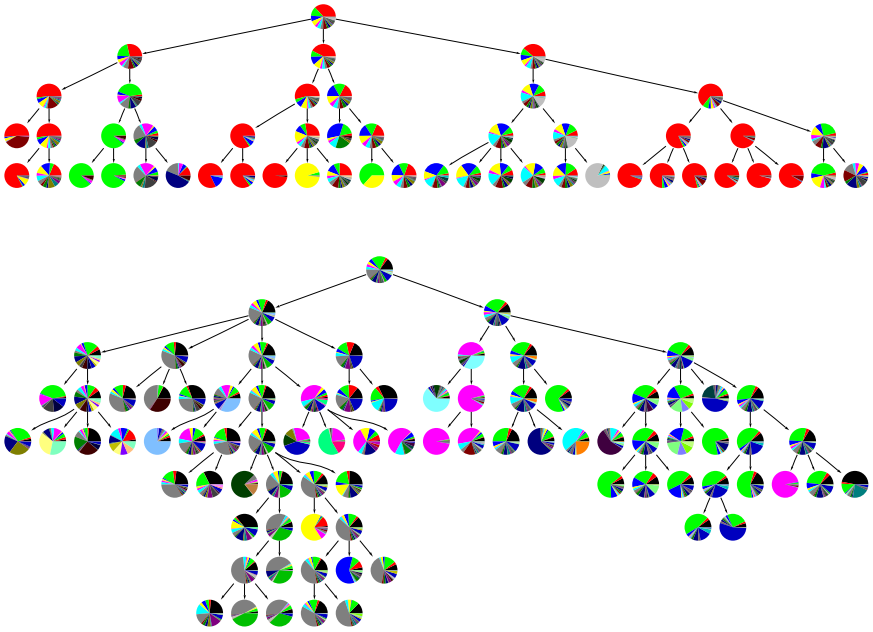

Fig. 5. The top of the hierarchy of the document (top) and user (bottom) clusters. Each color means different prefix in the URLs of the given cluster, and downloaded by the users in the cluster, respectively

8. E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, 1997.
9. R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *IEEE:2000:ASF*, pages 367–377, 2000.
10. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., 2002.
11. H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In U. M. Fayyad and R. Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases(KDD-94)*, pages 181–192, Seattle, Washington, 1994. AAAI Press.
12. B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *2002 IEEE International Conference on Data Mining (ICDM'02)*, 2002.
13. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001.
14. P. Pirolli and J. E. Pitkow. Distributions of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2(1-2):29–45, 1999.
15. J. E. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USENIX Symposium on Internet Technologies and Systems*, 1999.
16. A. Schechter, M. Krishnan, and M. Smith. Using path profiles to predict http requests. In *Proceedings of the 7th World Wide Web Conference, Brisbane, Australia*, 1998.
17. M. Spiliopoulou. The laborious way from data mining to Web log mining. *International Journal of Computer Systems Science and Engineering*, 14(2):113–125, 1999.
18. X. Sun, Z. Chen, W. Liu, and W.-Y. Ma. Intention modeling for web navigation. In *Proceedings of the 11th World Wide Web Conference (WWW)*, 2002.
19. A. Ypma and T. Heskes. Categorization of web pages and user clustering with mixtures of hidden markov models, 2002.
20. O. R. Zaïane, M. Xin, and J. Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Advances in Digital Libraries*, pages 19–29, 1998.
21. M. Zaki and M. Ogihara. Theoretical foundations of association rules, 1998.

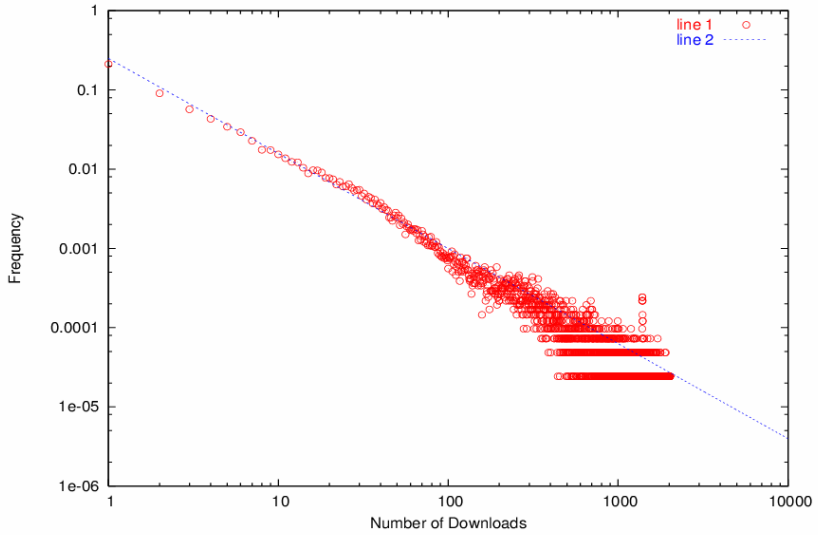

Fig. 6. Page Download. The horizontal axis stands for percentage of web documents on site. The vertical axis represents the percentage of pages being downloaded the given times

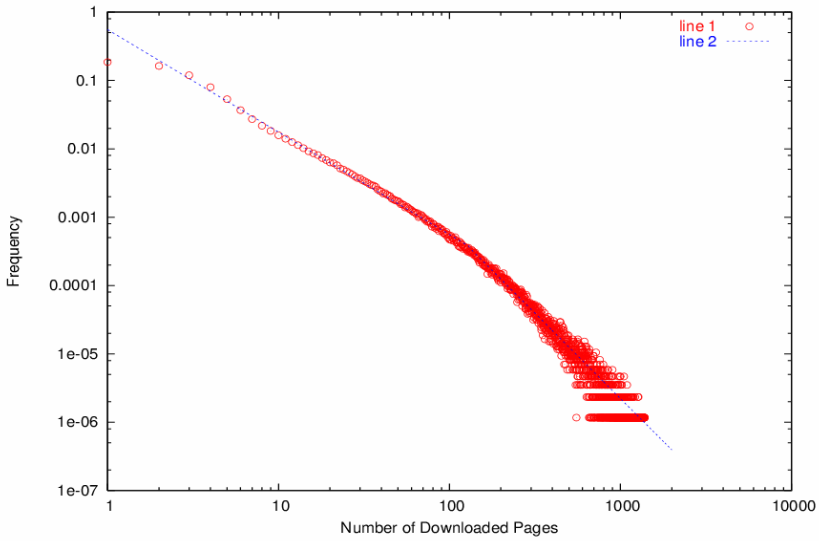

Fig. 7. User Download. The horizontal axis stands for frequency of users and the vertical axis represents the

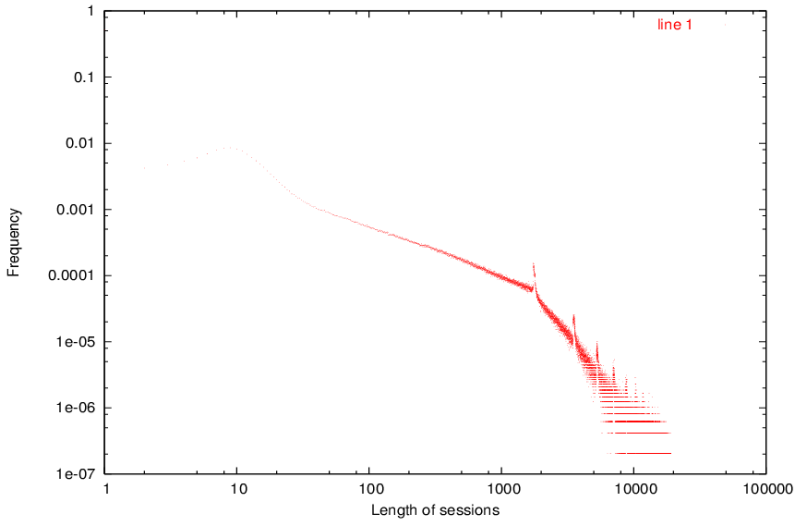
line 1
Frequency
Length of sessions
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1
10
100
1000
10000
100000

Fig. 8.

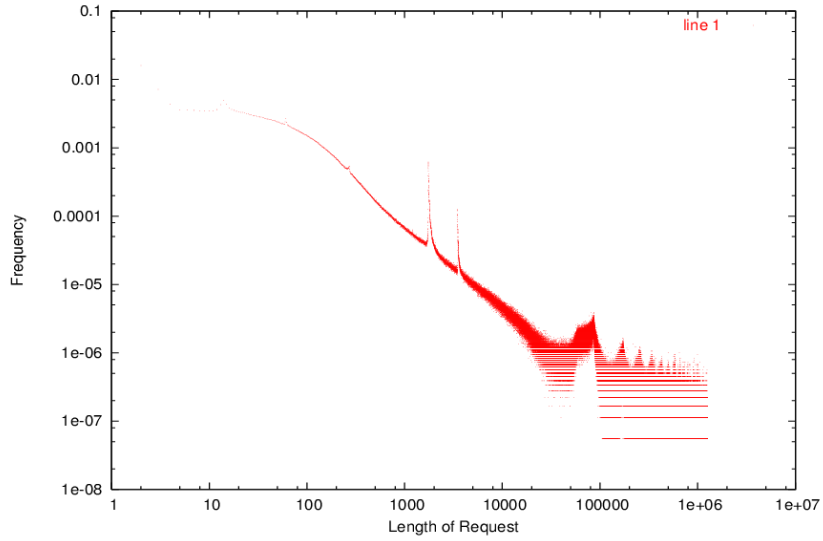
line 1
Frequency
Length of Request
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
1
10
100
1000
10000
100000
1e+06
1e+07

Fig. 9.