Sessions, Quests, and Long Term User Search Behavior

A thesis submitted in fulfillment
of the requirements for the degree of
Master of Science

by

David Mehrzadi

Supervised by
Prof. Dror Feitelson

School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

May 2011

Abstract

Today search engines are the worldwide source of retrieving information. People use them daily in various aspects of life for different aims such as business, entertainment, informational need, navigational search, etc. The importance of Web searching motivates us to understand Web searching trends and user behaviors. Figuring out user search behavior characteristics can help us to improve the user experience. To achieve this goal we need to use search logs. As the log files contain highly detailed interaction of users, this information can be used to reveal the users behavior. The length of the log file, i.e, the period of the record, can affect the behaviors that can be revealed, if at all, and as we want to study the user behavior in long term we used the longest log file available in literature, which is the AOL log.

The log files are just the records of user interactions, in the first step, we are interested in grouping related activities together and finding the boundaries between these groups. It is accepted to call these groups ”sessions”. There are two major ways for detecting session boundaries. The first is based on the interval of time that passed between the user queries. It is popular to use a global threshold for all the users to identify the sessions, where its value varies in different studies. The intervals longer than the threshold are thought to show breaks between sessions.

Using a global session threshold does not consider each user personally and rather all the users are the same, so it introduces artifacts that may affect the analysis. We show that using a global session threshold, regardless of its value, impacts the session durations and we do not obtain a natural distribution. Thus we suggest an algorithm that provide an individual threshold based on each user’s search activity pattern. We compared our algorithm findings with Human evaluations and received high precision and recall. We checked the impacts of our individual suggested thresholds on the number of queries in a session and the session length distribution. Finally we checked the correlation between them.

The second way for combining the log record entries is based on the similarity between queries. We first used a simple containment method and after that we used n-grams to find the similarity. All this was done after removing the stop words. Using the n-gram method we built heat maps that show the similarity between all the pairs of user queries. This enabled us to recognize different search patterns such as repetition, editing and etc.

Finally we merged these two different concepts of combining the user activities. Using our personal threshold and the n-gram method we received high precision and found out that most of the time these two methods agree with each other. We also noticed the topic changes in sessions that contained more than one quest.

As a part of revealing user behavior, we also studied classifying queries at the preliminary part

of our research. We concentrated on navigational queries, suggesting a simple method to identify them and evaluated their different properties.

Our findings can be used in future evaluations such as modeling the long-term user behavior, user search strategies in long-term, etc.

Acknowledgments

I would like to thank my supervisor Prof. Dror Feitelson for his guidance and help through this research work.
Also I would like to thank my family for their encouragement and support throughout my studies.

Contents

Chapter 1

Introduction

The development of the Internet and web in recent years provides access to a rich collection of information resources and services to the global community. Web search has become a central instrument for satisfying information needs, and largely replaced other media in searching such as newspapers, magazines, etc. Major search engines such as Google, Yahoo, etc are now among the largest companies in the world.

As the Web gets larger and more complex, it is more essential for users to find relevant information efficiently. The web's growth creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of users. Searching in the web is used extensively by users for many different purposes, so we need to engineer proper search systems that help users find what they need quickly and intuitively, and to improve the search experience for consumers. Engineering a search engine is a challenging task and to achieve this goal we need to understand the requirements.

Search engines are designed to serve human users and enable them to get the precise results they seek. To help improve searching on the web we need a better understanding of users characteristics: how they search, what are their behaviors, what are their search strategies. By understanding user search behaviors we will better understand the user requirements of search engines.

To achieve good understanding we need to analyze many users activities while searching. Regrettably there is no way to record all the aspects of the user activity, since users may search from home, work, cafe-nets, etc. Thus the only way to record users activity is logging their search queries as they are received by the search engine. This is augmented with some other parameters such as issuing time, URL and rank of viewed pages, etc.

Search engine logs contain the user activity and are a useful source of information about user behavior. This can be used to obtain a better understanding of web based information seeking and user's information needs. Analysis done using such logs can affect the design of search engines to improve the user experience and to provide more accurate results to improve their performance.

Several search engine logs are available for research from sources such as Excite, Altavista, etc. They cover variable periods of time, from hours to days or a couple of weeks. The AOL log is the longest published log that is accessible to researchers, and covers three month of activity. Previous work done using these logs mostly focused on short term user behavior and tried to characterize search parameters such as query length, number of viewed pages, number of clicked URLs, etc.

We concentrate on long term user behavior. We define two different concepts, of *session* and *quest*, and divide the sequence of user activity according to them as a way to reveal the user's activity while searching. Specifically, we identify sessions by suggesting per-user thresholds on the time intervals between each successive queries recorded in the log, and quests are based on the similarity between query terms, which help us to find the relationship between the queries in the search process. Moreover we consider the relation between these two concepts using the combination of our metrics. This is useful in characterizing search patterns.

We also try to categorize the users queries into different groups according to intent such as navigational queries. We suggest a simple method to identify navigational queries. Applying this method, we found longer time intervals after navigational queries. We exploit the long-term coverage of the log to identify queries that are related to news events, by noting the combination of high popularity in one week and relatively low popularities at other times.

This research tries to provide the first steps toward long-term behavior studies and to show related characteristics, it can be used a basis for different aspects of this topic for more deep analysis in the future, such as attempts to model the long-term user behavior, user search strategies in long-term, and periodic search in long-term.

Chapter 2

Related Work

Logs from web search engines record the activity of users. Available logs cover different periods of activity from one day to a couple of weeks or months and contain thousands of users. Using long logs that cover a couple of months gives us the possibility to study behavior of users for a continuous period that can help to find out different usage patterns which are impossible to reveal in short logs of one day or a week.

Most of the logs contain timestamps that mark the query issuing time but none of them indicates when the user stopped viewing the results, so the first challenge is to divide the user activity into a sequence of intervals to have a better view of users time-consuming activity and to classify user search patterns. These intervals are often referred to as sessions and there are various definition for them.

Silverstein et al. [27] define the session as a series of queries by a single user made within a short range of time. Jansen et al. [16] define it as the entire series of queries by a user over a number of minutes or hours. A session could be as short as one query or contain many queries. It was then refined by Jansen and Spink [13] as the entire series of queries submitted by a user during one interaction with the Web search engine.

An important part of the analysis is dividing the queries of a log into sessions. This segmentation is based mostly on the time-gap between query issuing or query reformulation. Silverstein et al. [27] used a global threshold of 5 minutes to separate sessions. Using global session thresholds is common and they are used with different values such as 10, 15, and 30 minutes [8, 10].

Using global thresholds for all users is not always appropriate. Although we know that it is impossible to find the real session lengths, we believe that using arbitrary values as global thresholds dose not identify the session boundaries. This was claimed also before by Murray et al. [22], who suggested a personal threshold using a hierarchical agglomerative clustering algorithm.

Using the content of the queries and looking for topic switches was also suggested as a way to identify sessions. Spink et al. [4] and He et al. [11] used this idea to define search patterns. There is also parallel usage of these ideas: He et al. [11] use the combination of search pattern and global threshold to identify sessions.

Another aspect that has been studied using query logs is classifying the queries according to intents or topics. This division aims to clarify the interaction of the user with the search engine. Broder [7] divided queries into three types of Navigational, Informational, and Transactional. This division is widely accepted and used by researches, although it was also expanded with some added sub-categories [17, 25]. Brenes et al. [5] used another idea in classifying queries, a grouping criteria based on the popularity of the query submitted to the search engine.

Statistics such as query length, session length, number of clicked results, etc. were used to characterize user behavior by Jansen et al. and Silverstein et al. [15, 27]. Even user location was considered: Spink et al. [30] compared US users with Europeans.

There are several query topic taxonomy definitions. Spink et al. [29, 31, 28] attempted to classify queries according to topics using 11 topical categories. Using this, they show how the focus of what people search for changes with time. Beitzel et al. [3] studied the query popularity changes on an hourly basis at topic categories. The number of categories and their sub-categories varies: Ross and Wolfram [26] used 30 categories and Chuang and Yang [24] used a taxonomy with 15 categories and 85 sub-categories.

All the researches mentioned above concentrate on user search behavior in short periods and do not evaluate the user search behavior in long term.

It is probable that search engine companies such as Google, Yahoo, etc. have conducted some additional research in this field but they are saved as business secrets and not revealed.

Chapter 3

Analysis

3.1 Choosing a Data Log

In our research we try to characterize user behavior so we preferred to use a log which contains the user's interaction with the search engine for a long period of time, for more than a couple of hours or days. In particular, using long logs allow us to find different patterns and phenomenas such as periodic searches or high rate news searches as we see later.

We choose to work with the AOL log, as it is yet the longest log published. We assume that there are search engine companies such as Google, Yahoo, etc. that have longer and bigger logs, but as they are not publicly published researchers have no access to them and their volume is unknown. The AOL log consists of about 30 million queries from about 650,000 users over three months (01, March, 2006 to 31, May, 2006). In this log all the queries of a user are gathered and given an anonymous identifier. The log is sorted by these anonymous user IDs. As in most logs, AOL contains the following fields: Anonymous-ID, Query, Query-Time, Item-Rank, Clicked-URL. Queries are case shifted and most punctuation marks are removed. Query-Time indicates the query issuing time for searching. Item-Rank is the rank of the item in the results list that the user clicked. Clicked-URL is the domain portion of the URL of the clicked item. The last two fields appear only if the user clicked on any item in the suggested results list, so tuples in the log can be divided into two groups of events: those where the user just searched the query and did not click on any results, so they contain just the first three fields mentioned above, and those queries that have been clicked on thus containing all the five fields. In the case that a user clicked on more than one item from the results list there would be multiple tuples correspondingly in the log. If a user asks for the next results page a tuple with the same query and a later time-stamp will be added to the log. It is possible to identify different queries and requests for new pages using different combinations of the first three fields.

In total the AOL log has 657,426 unique user ID's and contains 36,389,567 lines of data. 16,946,938 of them are queries without user clicks and the rest (19,442,629) are queries with user click-through. It contains 21,011,340 new queries (with or without click) and 7,887,022 requests for next-page of results. It has 10,154,742 unique (normalized) queries. These figures are from the README file accompanying the data. The AOL data log is divided into ten parts and sometimes

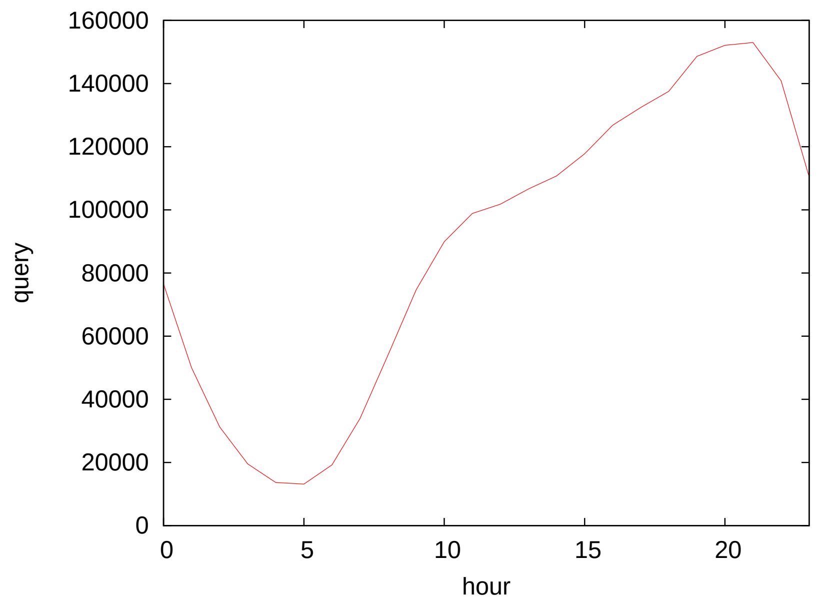

Figure 3.1: Number of web searches at each hour of day

we used one arbitrary part.

In the following section we analyze basic characteristics of the AOL search engine data log as the opening of our process of modeling user behavior in web search.

3.2 Number of Queries at Different Hours of Day

The log is supposed to mostly represent human users although some queries are probably issued by robots. As we put emphasis on human user behavior, it is desirable to remove user's activity that are non-humans (robots). The simplest method is to find these users by high rate of query issuing. In our study we use Duskin's analysis [9], which worked with the AOL log and classified the user to three groups of human, robots, and unclassified users that had no sure evidence to fit them to either of the first two groups. He supplies files with the ids of users for the last two groups. Although this classification may not be perfect, we removed all these users before any of our analysis to assure that our analysis is done on human users only. As the cleaned log contains mostly human users activity, we expect to see some pattern that specifies the preferred web search hours, and specifically a daily cycle of activity. Thus we simply counted the number of queries at each hour of the day for the full duration of the log.

Figure 3.1 shows that web searches are done at any hour and never stop. We can see that the lowest rate of searches is at 4:00 to 5:00 o'clock at morning when most people are asleep. Then it starts to arise during the day till it reaches its highest rate at about 20:00 PM to 21:00 PM, and then decreases again. We can say that the preferred time for searching is at evening after the day's work time.

To see if we have any interesting pattern also for web search at each minute or second of the

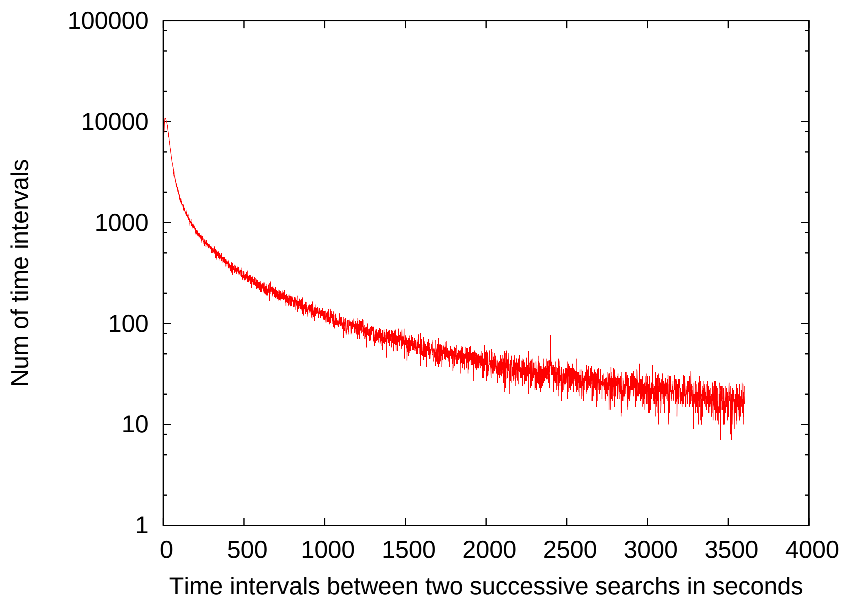

Figure 3.2: Time intervals histogram in logarithmic scale in one hour

day, we have also counted number of queries at each minute and second of the day, but as expected these look like random noises and don't follow any pattern.

3.3 Time Interval between Two Successive Queries

Another interesting point is the rate of issuing queries by users. The time interval distribution of adjacent queries of users can help us understand the session (which will be defined later) time length of users. To do this we traversed the log tuples and found the time difference between each two adjacent queries of each user. The adjacent log entries of users that had the same time-stamp with identical query were removed. This way we also find the most frequent interval, which is the interval of time with the highest number of occurrences.

At first we determined the maximal interval length to be 60 minutes and ignored longer intervals as we thought although these queries are sequential, passing an hour to issue the second query does not point to continuous search activity. Figure 3.2 (note Y axis is in log scale) shows the time intervals between adjacent queries with maximum length of an hour. As we see the curve is smooth and continuous so classifying the time intervals to different long or short groups would not be easy since the graph shows no threshold that divides it. It is opposite to what is expected, a bi-modal distribution which divides the intervals into long and short groups of intervals which would point to intervals that are part of a continuous search activity or intervals that are between these continuous series of activity, what are usually referred to as time intervals in or between sessions.

As we can not exclude the existence of users who searched continuously after more than an hour, and also to have a wider view, we set the maximal time interval to one day (24 hours) and repeated the previous process. Our findings are illustrated in Figures 3.3 and 3.4 (note X axis is in

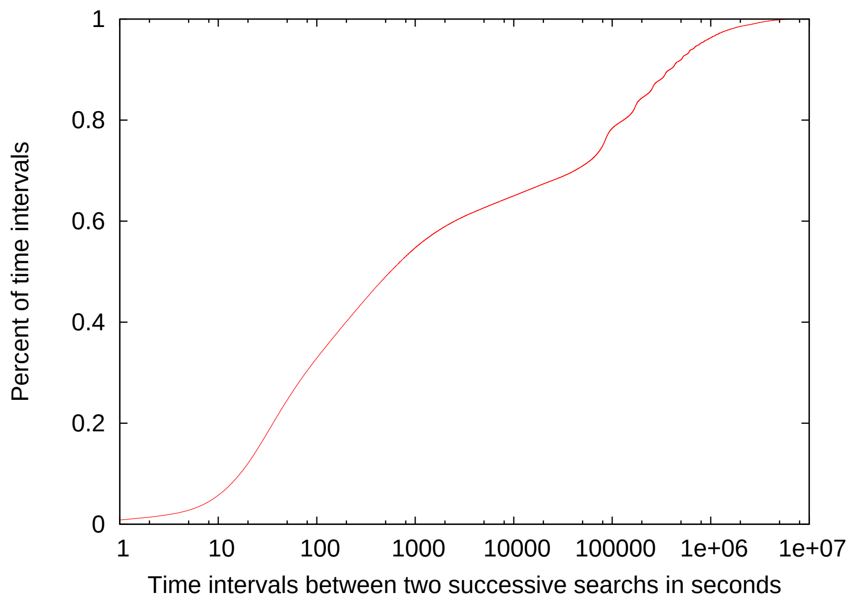

Figure 3.3: Percentage of time intervals between searches

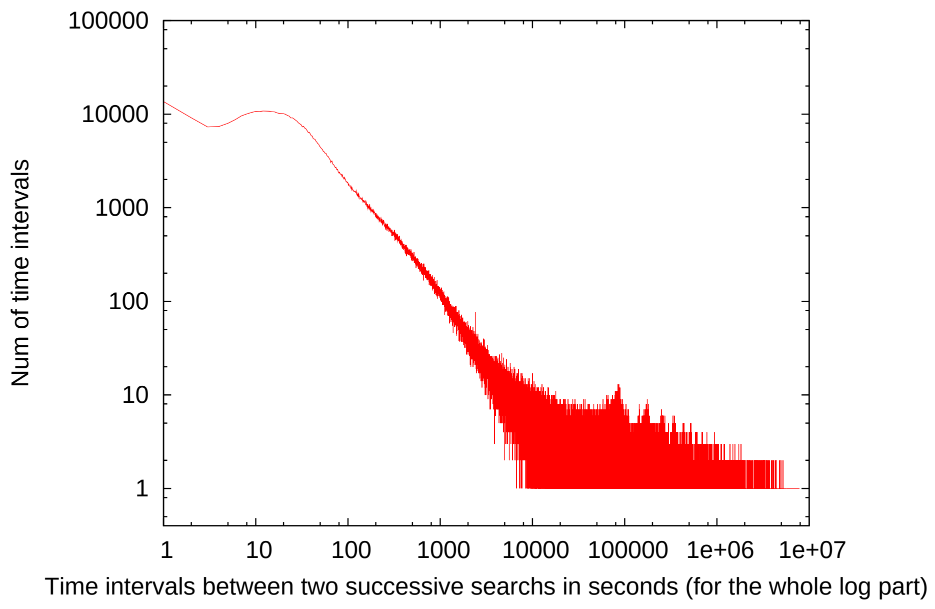

Figure 3.4: Time intervals histogram in log-log scale

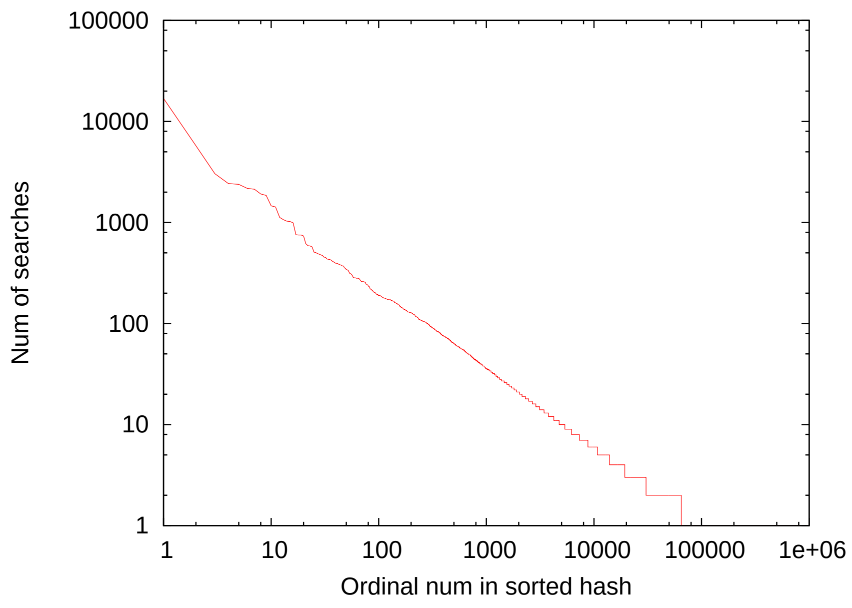

Figure 3.5: Queries and number of times they were searched in log-log scale

log scale). These Figures show all the time intervals for any length even longer than on day. We can see that about 30% of intervals are shorter than one minute and less than 40% of them are longer than one hour. We can see also that there exist intervals that reach our maximal boundary (which is 24 hours) but as the intervals get longer their number decreases. Figure 3.4 shows a histogram of the intervals between queries on log-log axis. This indicates that the distribution actually has a heavy tail.

3.4 Most Popular Queries and Their Rate

In order to find the most popular queries we traversed the log and for each query recoded the number of times that it was searched by different users. In the case that a user searched a query more than one time we just count his first search. Then we sorted the queries according to the number of times they were searched, giving the first rank to the most searched query term. Figure 3.5 illustrates this (this figure is based on one part of the AOL log which was choose randomly). This figure is in log-log scale and shows a straight line that is an indicator of a Zipf distribution which is common for such cases.

We see that there are few queries that have high searching rates and most of the queries were not searched too often. The most popular query is ’-’ with about 17,000 hits, which we think is typing typo or some preprocessing search engine fault. The first twenty highest ranked queries appear in table 3.1.

We also observe terms such as ’http’ and ’.com’ with high ranks. We tend to attribute these also to some preprocessing search engine fault, but their popularity obeys the Zipf distribution according to graph 3.5.

Rate	Hits	Query
1	16925	-
2	5752	google
3	3046	mapquest
4	2431	google.com
5	2382	yahoo.com
6	2176	ebay
7	2137	yahoo
8	1918	internet
9	1852	myspace.com
10	1461	www.google.com
11	1425	http
12	1123	www.yahoo.com
13	1065	weather
14	1030	myspace
15	1022	.com
16	992	map quest
17	757	american idol
18	751	mapquest.com
19	751	www.myspace.com
20	736	ebay.com

Table 3.1: The first twenty highest ranked queries

3.5 Classifying Queries into Different Groups

We tried to classify the queries into three categories: popular-queries which are queries that are always asked, news-queries which are asked during a specific week depending on news events, and the rest.

We divided the three month log data into thirteen weeks and retrieved the first hundred most searched queries of each week. Then we gathered all these queries and found for each query the number of times (weeks) it appeared as a top-100 query. We got the diagram shown in Figure 3.6.

We can recognize a bimodal distribution of queries in the graph. The first high column represents the queries that were included just for once in the first hundred high searched weekly queries. The last column are the queries that were included during all the thirteen weeks in the first hundred most searched weekly queries. We believe that obviously the first column and also the second and third column are mainly news-queries. Actually the news queries are usually popular for just a few days, but if the news event occurred at the end of the week it would appear popular for the next week too.

Table 3.2 shows some examples of news-queries that were popular during one, two, or three weeks. The Week column shows the week the query appeared. The query's rank and number of

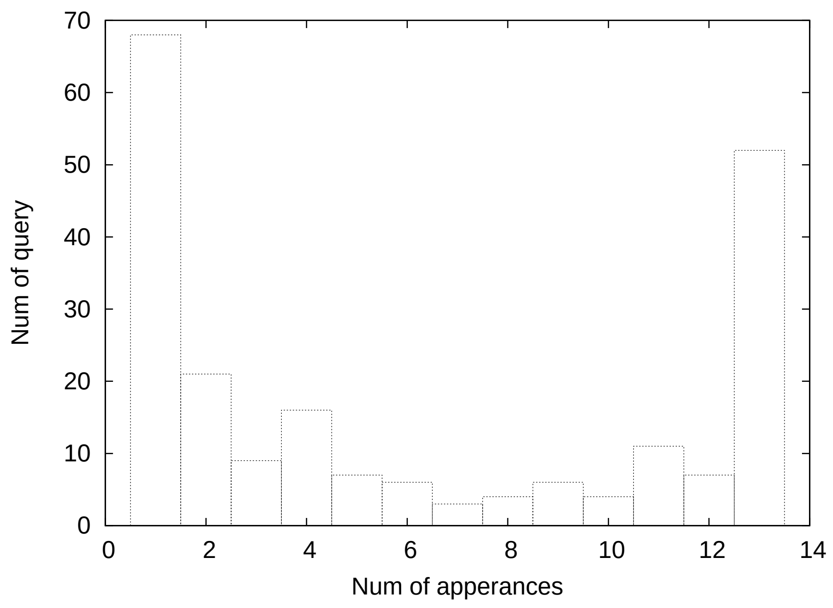

Figure 3.6: The appearance of first 100 most searched weekly queries in other weeks

searches are in the Rank and Num columns. The last two columns are the averages of the weeks the query appeared. Note that the AOL log starts on 1, March, 2006 which is Wednesday, and in our week division we added these first days to the first week. The news-queries are ordered according to the maximal week rank.

Checking a sample shows that these are indeed news-queries: ”david blaine” was at rank 33 in the tenth week and had lower ranks in seventh and ninth weeks. Blaine is a magician and endurance artist. A quick glance at Wikipedia will reveal that on first May, 2006, ”Blaine was submerged in an 8 feet diameter, water-filled sphere (isotonic saline, 0.9% salt) in front of the Lincoln Center in New York City for a planned seven days and seven nights, using tubes for air and nutrition”. We see that the date it happened is appropriate to week we assigned. Undoubtedly it is a news-query that survived for more than one week.

At the third place stands ”anna benson”, she filled for divorce on March 31, 2006, this was enough to place her at rank 49 at fifth week.

Another news-query is ”kentucky derby”, in weeks 9 and 10 and ranked for 194 and 50 respectively. The 2006 Kentucky Derby was the 132nd running of the Kentucky Derby horse race and was held on May 6, 2006. As before weeks are assigned correctly. ”barbaro” the winner of the 2006 Kentucky Derby, another news-query, shattered his leg two weeks later on May 20, 2006, in the 2006 Preakness Stakes, this event caused his name to appear for two weeks with ranks of 64 and 248.

On April 21, the 55th Miss USA pageant was held. Accordingly, ”miss usa” met rank 118 in that week. While we just checked some of the queries from table 3.2, undoubtedly the rest of the events are also related to some news event.

Obviously the queries in the last three columns of Figure 3.6 are the popular-queries. It may happen that in some week they did not reach enough hits to be in the top hundred queries, and thus

Query	**Week**	**Rank**	**Num**	**Average Rank**	**Average Num**
whitney houston	5	10	142	10	142
david blaine	7 9 10	462 430 33	5 6 44	308.3	18.3
anna benson	5	49	35	49	35
kentucky derby	9 10	194 50	10 36	122	23
nfl draft	9	54	25	54	25
barbaro	12 13	64 248	26 9	156	17.5
ncaa	2 3 4	108 69 169	23 32 16	115.3	23.6
memorial day	13	72	21	72	21
vanessa minnillo	4	78	27	78	27
the simpsons live action	4	81	25	81	25
chris daughtry	11	92	21	92	21
miss usa	8	118	14	118	14
taylor hicks	12 13	365 126	8 16	245.5	12
grey's anatomy	11	165	14	165	14
preakness	12	167	13	167	13
ncaa brackets	2	167	17	167	17
indy 500	13	198	11	198	11
mothers day poems	11	216	12	216	12
george mason university	4	327	10	327	10

Table 3.2: Examples of news-queries

they are counted only in eleven or twelve weeks.

To achieve a better classification of the queries that fluctuate around the threshold we decided to repeat this process with the first five hundred most searched queries of each week, see Figure 3.7. This would let the queries to appear in more weeks. For example the query ”movies” succeeded to be in the first hundred most searched weekly queries only in week thirteen, but after expanding the range of queries, it appears in all the weeks. The query ”nick.com” that did not appear at all before now appears in nine weeks. Queries such as ”espn.com” and ”tattoo” succeeded to improve their appearance respectively from one to eleven and from three to eight weeks.

We see that by expanding the threshold we got many more news-queries, as the first and second column almost multiplied themselves seven times. The last three columns that represent popular-queries have slower growth and multiplied themselves almost between three to four times.

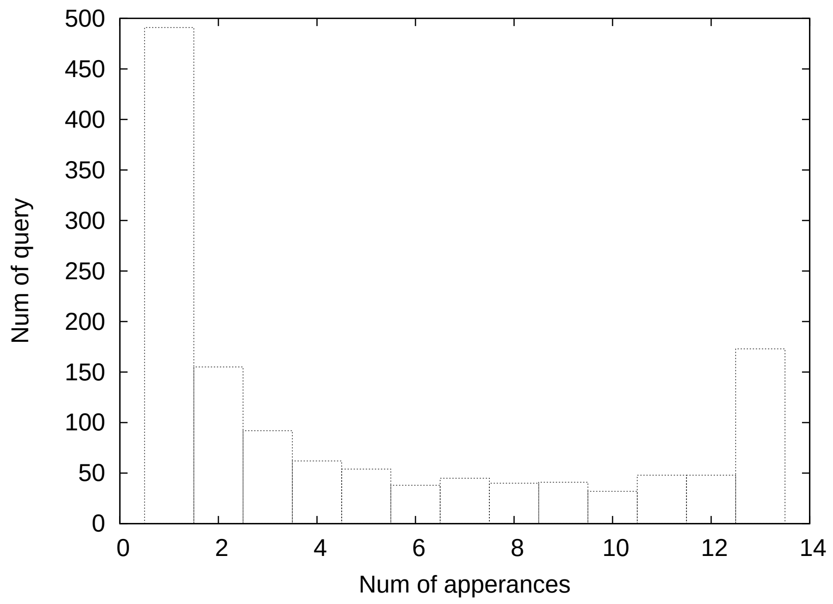

Figure 3.7: The appearance of first 500 most searched weekly queries in other weeks

Table 3.3 shows the fifty top ranked queries in the log. The first column is the average rank of the query during weeks it succeed to be in the first 500 most searched queries during the thirteen weeks. The second column is its average number of its appearances during these weeks. The last column shows the number of weeks the query appeared. We observe some changes in query ranks after expanding our boundary from 100 most searched weekly queries to 500 most searched weekly queries. We see that if we consider the week popularity to rank queries, some news-queries often succeed to be in twenty first.

Table 3.3: Top ranked queries in the log

Average rank	**Average Appearance**	**Query**	**# Weeks**
1.7	738.9	google	13
2.9	344.07	mapquest	13
4.8	268.6	yahoo.com	13
5.3	260.2	google.com	13
5.7	255.2	ebay	13
6.1	245.6	yahoo	13
7.8	209.3	myspace.com	13
8.8	176.3	internet	13
9.7	1778.2	-	13
10	142	whitney houston	1
10.07	148.3	www.google.com	13
12.2	127.07	http	13

12.6	114.7	www.yahoo.com	13
13.8	107.5	myspace	13
14.6	101.5	map quest	13
14.7	104	weather	13
15	96.5	profileedit.myspace.com	2
17.9	84.5	.com	13
19.1	75.8	ebay.com	13
20	82	recent	2
20	71	sp.trafficmarketplace.com	1
20.1	74.7	www.myspace.com	13
20.5	77.5	american idol	13
21	65	slirsredirect.search.aol.com	1
21.4	68.3	mapquest.com	13
22.6	59.07	dictionary	13
25.0	57.1	www.google	13
27.2	53.6	m	13
29	47	org.co-motion.com	1
29	47	iframe.adultfriendfinder.com	1
31.3	46.6	southwest airlines	13
31.7	50.3	ask jeeves	13
31.7	66.3	mycl.cravelyrics.com	9
32	46	hotmail.com	13
33.3	46.3	com	13
33.5	45	bank of america	13
33.7	45.6	walmart	13
34.2	43.8	msn.com	13
34.6	43.6	white pages	13
35	44.7	maps	13
36.4	44.7	ask.com	13
38	57	games.myspace.com	1
38.5	44.6	www.	12
39	38	preferences	1
39.07	40.1	my space	13
39.2	40.3	home depot	13
41	41	dir.porthalcyon.com	1
41.3	39.1	om	13
41.7	39	yellow pages	13
43.3	37.9	travelocity	13
44.07	38.3	my	13
46.4	39.1	msn	13

46.4	36.4	im help	13
46.4	35	target	13
48.9	34.07	goggle	13
49	35	anna benson	1
49	32	search	1
50.1	35.3	www	13

3.6 Manually Classifying the Popular Queries

To expand our query categorization we decided to classify manually the first 250 popular queries, which were ranked between 1.7 to 202.5, into four different categories of navigational, informational (indirect), news event, and strange queries. This partition was done by one examiner using his personal knowledge and Internet help in cases where he was suspicious about the query.

We define our categories as following: Navigational queries are mostly one phrase queries that are aimed at a specific entity (meaning that the query has only one satisfactory result). This means there is a unique page that the user is looking for. Examples include ebay, msn, bankofamerica.

Informational (indirect) queries are queries that have more than one appropriate result, and it is expected that the user would search again at the page he receives. Examples are weather, dictionary.

News event queries are queries that are generated during a limited time period due to some specific event, often celeb names such as whitney houston, david blaine, anna benson.

Strange queries are queries that have no meaning or they are ambiguous enough that we could not classify them to the previous three categories. ’-’, ’www.’ and ’internet’ belong to this group.

We found out that 67.6% (169) of the queries were classified as navigational, 12.5% (38) were classified as informational, 6% (15) were classified as news queries and the rest 11.2% (28) were classified as strange queries. Although this sample is small, it shows that the most popular queries of search engines are dominated by looking for some specific web page that the user has in mind and wants to access.

This finding that a sizable portion of queries are navigational persuaded us to perform a deeper study of navigational queries.

Chapter 4

Navigational Queries

Navigational queries are queries that look for a specific website or web page of an entity (company, institute, or brand). These queries can be one or more terms that contain the name or referral the user seeks, and can be also a complete or partial URL query.

4.1 Recognizing Navigational Queries

Manually classifying queries can be done on small data, but it is not suitable for logs that contain millions of queries. Moreover, we want to be able to recognize queries automatically. Given the AOL log entries we needed some method to identify the navigational queries.

There are different ways of recognizing navigational queries that were suggested by researchers separately or with methods to recognize other kinds of queries (informational, transactional). Kang and Kim [19] offered four different methods to distinguish between navigational and informational queries, or more precisely what they call ”topic relevance” and ”homepage finding”. Three of them need training collections, and the last used part of speech (POS) tagging, so they used the ten gigabyte WT10g collection. (http://ir.dcs.gla.ac.uk/test_collections/wt10g.html)

Lee et al. [21] tried to distinguish between navigational and informational queries using clickthrough data and anchor texts. They used the clickthrough data to compute click distributions and average click number for each query. They need a large amount of data to compute these two criteria so they used the anchor-link distribution which is similar to the click distribution but gathered from a collection of web pages. Nettleton et al. [23] and BaezaYates et al. [2] used machine learning methods to categorize queries by intent, although the categories are not identical to Border's [7]. Jansen et al. [14] used simple rules to clarify the intent of each query. They assume that navigational queries contain names of companies or organizations, contain domain suffixes, or have less than three terms. Actually performing some of these criteria to automatic queries classification requires external information collection [6].

Our suggested method is simple and also avoids the overhead of other methods in collecting data in advance to help in recognizing the navigational queries.

As we know navigational queries look for a single page of a brand or company, we assume that these queries contain terms that are related to the sought page URL. This means that the company

ID	Query	Date	Time	R	Clicked URL
6819	www.clum.com	2006-03-08	18:50:13	1	http://www.clum.com
11754	www.bigbearadventures.com	2006-04-03	15:40:49	2	http://www.bigbearadventures.com
17759432	www.bravotv.com	2006-03-24	20:00:31	1	http://www.bravotv.com
12973	www.usbank.com	2006-05-20	07:24:00	1	http://www.usbank.com
12973	www.aib.edu	2006-03-01	17:29:34	1	http://www.aib.edu
5621609	www.bmssports.com	2006-03-15	18:18:07	1	http://www.bmssports.com

Table 4.1: Examples of full URL navigational queries

ID	Query	Date	Time	R	Clicked URL
4575	fox	2006-03-09	23:52:59	3	http://www.fox.com
8569434	walmart	2006-05-18	16:57:43	1	http://www.walmart.com
9721032	orbitz	2006-03-26	15:10:09	1	http://www.orbitz.com
9722033	mapquest	2006-03-24	12:55:52	1	http://www.mapquest.com
11385	wifelovers	2006-05-05	10:52:06	1	http://www.wifelovers.com
24969595	mrislc	2006-05-31	21:49:56	1	http://www.mrislc.com
997	cartoonnetwork	2006-05-26	14:46:38	1	http://www.cartoonnetwork.com

Table 4.2: Examples of one term navigational queries

name or brand or some expansion of them tend to appear in the URL itself. Therefore we suggest the following to recognize navigational queries. First, if the query contains more than one term then we removed all the white spaces to create a single term. Then we check if the query is contained in the URL that was clicked (in case it exists). For those queries that have full or partial URL structure we took the domain part of the URL to check if it is contained in the clicked page URL. This avoids URL typing mistakes such as misspelling or confusion between org, com, and net. Following are some examples of this process.

First we present examples of normal navigational queries where the only required editing is removing spaces. The recognition of these navigational queries can be divided to three sub categories. Table 4.1 shows queries that contain the full URL that was later clicked. These reflect a situation where users write the URL in the search box rather than directly in the browser's navigation bar. The R letter in tables represents the rank in the result list.

Table 4.2 shows some examples of the queries that contain domain term from the clicked URL. Here the users save the need to type the ”www.” and the ”.com”.

Table 4.3 shows queries where we need to remove spaces and convert them to one term to recognize them.

Table 4.4 shows some examples where more editing is needed, to correct confusion between suffixes such as org, com, and net when typing the URL query.

Table 4.5 shows examples of navigational queries that our method finds where it is not enougth to simply check that the query is contained in the clicked URL, since the query has more terms or misspelling.

ID	Query	Date	Time	R	Clicked URL
997	proformance insurance	2006-03-13	08:05:24	10	http://www.proformanceinsurance.com
19395	summation technology	2006-03-09	10:41:10	1	http://www.summationtechnology.com
125284	bank of america	2006-03-13	19:30:47	1	http://www.bankofamerica.com
814345	home depot	2006-05-24	23:04:25	1	http://www.homedepot.com
8020	virginia natural gas	2006-03-28	16:00:58	1	http://www.virginianaturalgas.com
11062189	michele watches	2006-05-30	22:21:48	2	http://www.michelewatches.com

Table 4.3: Examples of navigational queries needed to remove middle spaces

ID	Query	Date	Time	R	Clicked URL
90255	www.mysticseaport.com	2006-03-20	16:08:32	1	http://www.mysticseaport.org
879803	www.caringbridge.org	2006-03-01	17:03:34	2	http://www.caringbridge.com
47870	www.netscape.net	2006-05-12	23:19:45	1	http://www.netscape.com
41441	www.bugzone.com	2006-03-29	22:16:17	1	http://www.bugzone.net
24658241	www.harpercollege.com	2006-05-30	15:54:30	1	http://www.harpercollege.edu
165141	www.ntta.gov	2006-05-13	19:34:29	1	http://www.ntta.org
2715425	www.wyandotte.gov	2006-04-06	23:31:39	1	http://www.wyandotte.net

Table 4.4: Examples of navigational queries with confusion between suffixes

4.2 Time Interval Length after Navigational Queries

As we want to characterize the long term behavior of users we checked if the time interval distribution of navigational queries would be the same as the distribution of non-navigational queries. This would show if the time intervals between queries are correlated with user intent. Using the former navigational recognition method we found all navigational queries, and measured the time interval to the next query issued by the same user. We also measured these intervals for non-navigational queries. Figure 4.1 contains both distributions. (note the figure is sparse)

ID	Query	Date	Time	R	Clicked URL
203999	www.npr.org help index	2006-03-02	18:31:00	1	http://www.npr.org
218649	http www.amn.org	2006-03-01	23:54:03	6	http://www.amn.org
416589	www.compasspoint.con	2006-04-05	17:49:30	1	http://www.compasspoint.org
48072	www.webquest.com	2006-04-21	20:09:16	7	http://webquest.org
416589	disneymovies.go.com	2006-04-04	23:43:45	1	http://disney.go.com
23090415	www.molottery.state.mo.us	2006-05-06	18:43:33	1	http://www.molottery.com
24569686	www.walmart.com	2006-05-30	14:02:32	3	http://www.walmartstores.com
24609672	dr.phil.com	2006-05-26	10:38:51	1	http://www.drphil.com
24926619	www.disney.com	2006-05-31	20:53:01	5	http://radio.disney.go.com

Table 4.5: Examples of navigational queries needed to be checked deeply

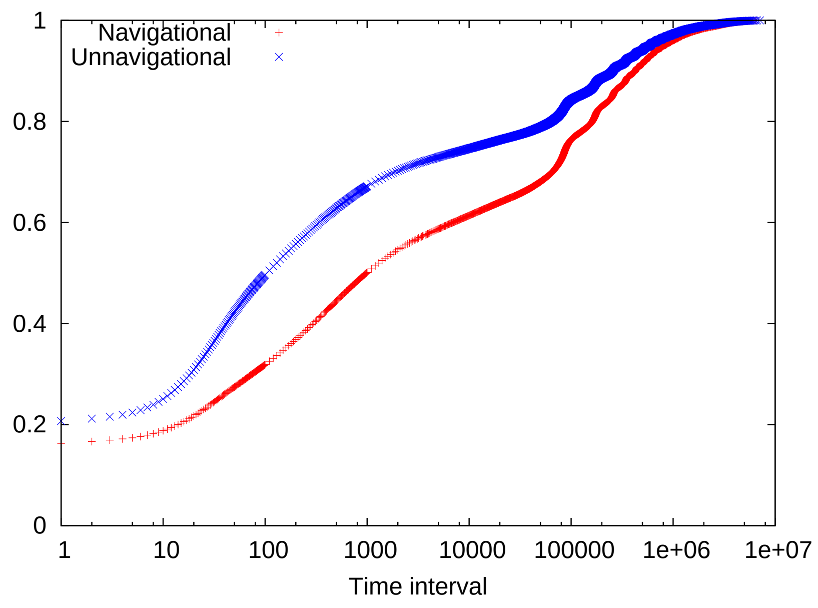

Figure 4.1: Time interval distribution of un/navigational tagged queries

As we can see in the graph, the navigational time interval curve is lower than the non-navigational one, which means that after issuing a navigational query it takes a longer time for the user to issue another query. This matches our expectation since we assume that when a user seeks a specific site and finds it, he will spend more time at that site before issuing another query compared to a user that uses the search engine for information needs and wants to check different sites that were suggested by the search engine.

4.3 Fraction of Navigational Queries in Web Searches

Knowing the portion of navigational queries in web search logs would help us to know roughly the percentage of navigational queries issued by users. We randomly picked one part of the AOL log, and using our navigational recognizing method found that out of 3,813,394 queries, 459,411 of them were tagged as navigational which is 12.047%. This was performed without any user filtering while each entry in the log was considered independently. We repeated this with regard to unique queries, and this time we had 824,287 different queries where 64,842 of them were tagged as navigational, which is 7.86%. We also counted the sequence of different queries per user; this way the total queries were 1,337,875 and 168,859 of them were tagged as navigational which is 12.62%. This shows that at least one out of ten queries issued by user is navigational, although this rate depends on our method and could vary using other criteria.

This ratio of navigational queries is much smaller than that one found in the first 250 popular queries in Section 3.6. This indicates that in the top popular queries there is a high concentration of navigational queries. Our finding are in the range found by Jansen et al. [14], who found that about 10% of queries are navigational. Broder [7] choose 1000 random queries and after filtering, inspect the first 400 and determined 20% of them to be navigational. (the paper does not say if the classification was manual or by a program)

We have also used our method to verify the first 250 popular queries that were tagged manually in Section 3.6. To do this we collected all the tuples that contained one of these queries from one arbitrary part of the AOL log. We found that out of 249 unique popular queries (after removing the '-' query), 171 were tagged as navigational queries, which is 68.67% (if dividing by 250 it is 68.4%). The remaining 78 unique queries just appeared at popular queries. This rate is a little higher than the manually tagged rate, which was 67.6%.

Given the big difference between the general rate of navigational queries and their rate in the popular queries, we checked the relationship between navigational queries and popularity. We choose one arbitrary part of the AOL log and ordered all the queries by the number of times they appeared. To prevent users that search one term more frequently than others from impacting on our analysis we just count each user once, meaning the queries were actually ranked by the number of times they were searched by different users. We then divided these ranked queries into logarithmically sized groups, since we expect to see a higher rate of navigational queries in the first groups. We thus divided them as follow: the two first groups each contain 250 queries, the third group contains the next 500 queries, the fourth group contains the next 1000 queries, and the groups double their size till the last group that contains the all the queries left (its size is not double of the previous group). Then for each group we gathered all the unique queries from the arbitrary chosen log part, and using our navigational recognizing method found all the navigational queries they contained. Figure 4.2 shows the portion of navigational queries recognized at each of these groups (note the X axis is in log scale). We see that for the first two groups that contain together the first 500 most popular queries the rate is almost the same and range from 76.4 to 77.6. The rate of navigational queries in the next 500 popular queries decreased to 68.4. As the groups contains less popular queries, the portion of navigational queries also decreases. The last four groups have the lowest ratio of navigational queries which is almost the same value that is roughly 6.2.

4.4 Non/Multi Clicked Navigational Queries

Another interesting point about navigational queries we have observed in the AOL log was that there are some complete or partial URL queries (such as: www.x, x.com, etc.) that were not clicked at all by the user, or for the same query the user clicked on different URLs from the list of search results. Naturally, we do not expect such behavior if we classify these queries as navigational. In a navigational query we would like to see that the user clicks on the suggested URL and does this at most once, otherwise we can not be sure to classify the query as navigational. From now on we refer to this kind of queries as URL queries (or semi-navigational). To check the aspects of this phenomenon we decided to find the portion of these URL queries out of the total user's different queries and also to check how many of these URL queries were not clicked at all. This is illustrated in Figure 4.3, where the X axis represents the percentage of URL queries out of all queries of the user, and the Y axis shows the percentage of URL queries that were not clicked out of the URL queries.(note the graph is sparse randomly and only 20% of information is represented)

In this graph we see high concentration at the top left corner which shows that typically few

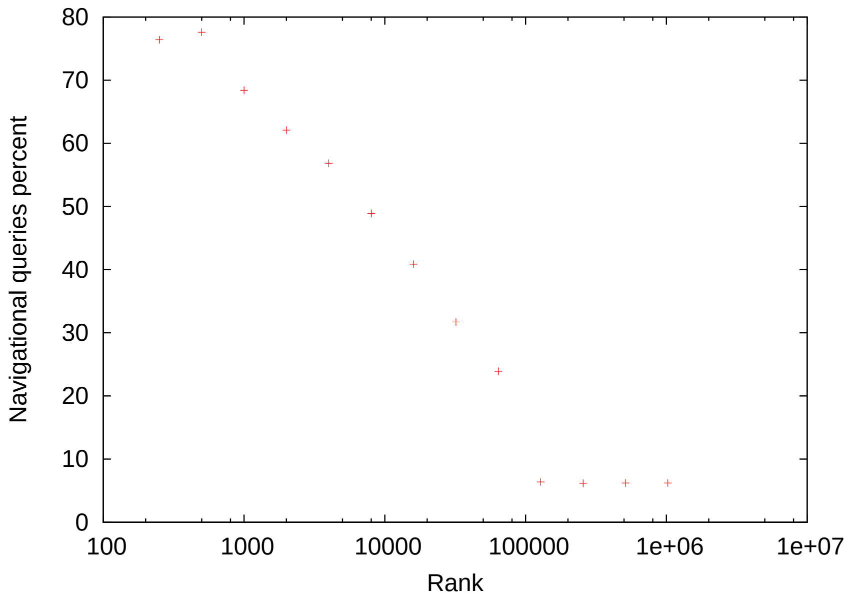

Figure 4.2: Popular queries and their navigational query ratio

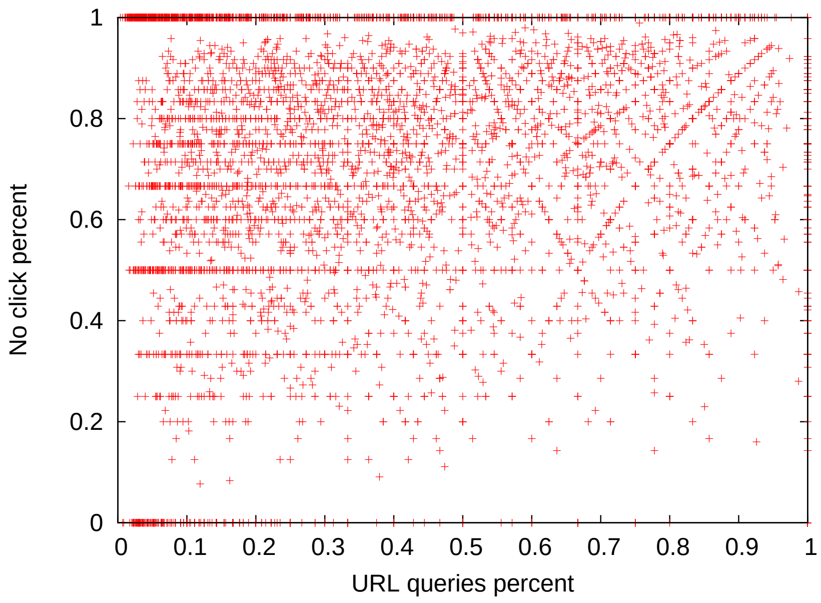

Figure 4.3: URL queries percent and their non click percent

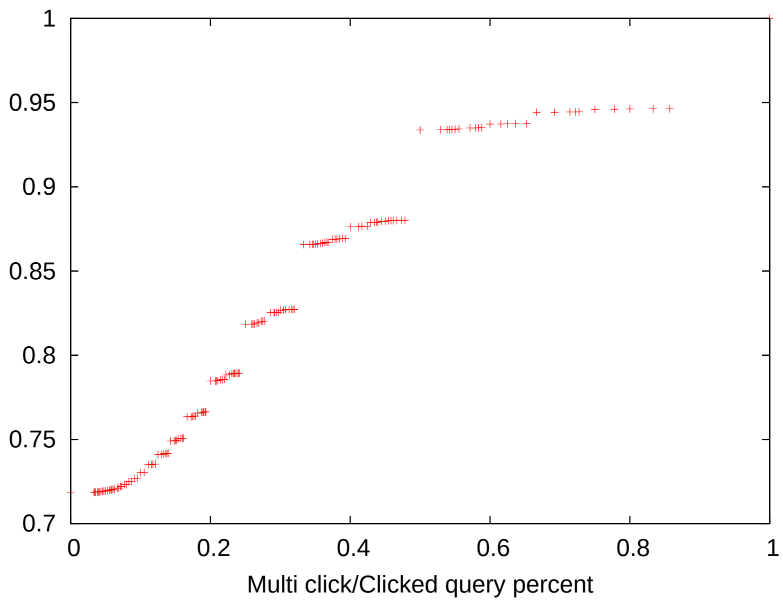

Figure 4.4: Multi clicked percent out of URL clicked queries

of the queries were URL queries, and they were not clicked. This is surprising for the URL (navigational) queries as noted before. The parallel horizontal lines at the left of the figure are just simple fractions such as 1/2, 1/3, 2/3, etc. The curves at the right are also an artifact of such discretization.

The other surprising event we checked was the appearance of multi clicked URL queries. We decided to find the percent of these multi clicked queries out of the URL queries for each user. Figure 4.4 shows the CDF of this percentage. The curve in the graph is not continuous since it represents the fraction of multi clicked out of clicked queries. As these are discrete and sometimes small numbers, there is a concentration of cases with the same ratio like 1/2, 1/3, etc. leading to discontinuities in the graph at these values.

We can see that more than 70% of the URL queries were clicked only once, and less than 10% of users had clicked more than once on half of their URL queries. This shows that multi clicking on URL queries is not very common. We speculate that it may happen when the URL query does not define the web page uniquely or correctly and the user needed to check several suggestions, or when the user searches for some page which is not the main page and the search engine returns the main page and more specific (deep) pages such that the user may need to click on them to check them.

We also sorted URL queries according to the number of times they were clicked on different suggested URLs, meaning the first set contains the URL queries that were not clicked at all, the second set contains the URL queries that were clicked once or more but on the same URL, the third set would contain all the URL queries that user clicked on two different URLs once or more times and so on. We collect this data for each user separately and then aggregate it to prevent the influence of specific clicking behavior of one user on the whole data. The results are shown in graphs 4.5, 4.6.

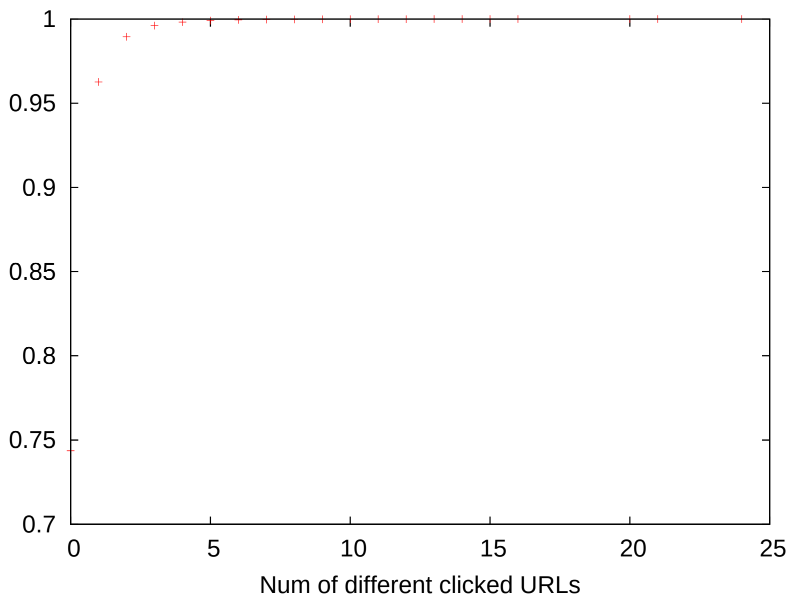

Figure 4.5: URL queries click distribution

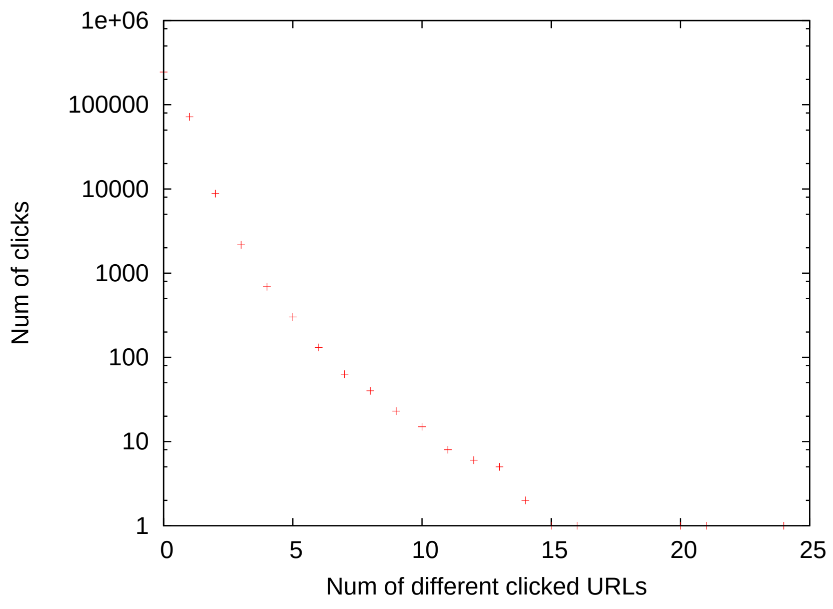

Figure 4.6: Queries percent and their non click percent

We can see that about 74% of URL queries were not clicked at all, and 22% of them were clicked only on one URL. The rest, which is about 4%, were clicked on two different URLs. The portion of URL queries that were clicked more than two different URLs is less than 1%.

Figure 4.6 shows the histogram of the previous sets. We can definitely point to the second mark on the graph as navigational queries since it represents URL queries with one click on them. Regarding the many zero-click queries, we actually do not know the reasons why users searched for URLs and did not click on the search engine suggestion. Maybe just a glance on the suggested pages was enough to decide that they are not relevant. Or maybe the user stopped his search to drink coffee, answer the phone, etc. or simply got tired and left the search. In any case these are characteristics of user sessions, and knowing them could help us analyze the user behavior better.

We have seen that a notable portion of queries are navigational, it is guessed that navigational queries have short sessions, so it would be interesting to find out the distribution of sessions and check it in more details.

Chapter 5

Sessions

A session in web search can be the sequence of queries issued by a user during one day or from the start of using the browser till closing it, or the sequence of queries with short inactivity gaps between them.

There are various definitions in the literature for sessions. Silverstein et al. [27] defined the session as ”a series of queries by a single user made within a small range of time; a session is meant to capture a single user’s attempt to fill a single information need”. Jansen et al. [16] defined it as ”the entire series of queries by a user over a number of minutes or hours. A session could be as short as one query or contain many queries”. Later Jansen and Spink [13] refined it as ”the entire series of queries submitted by a user during one interaction with the Web search engine”.

We define a session as the sequence of queries issued by a user such that the time interval between them is less than or equal to some session threshold. Session identifies a continuous progress of searching in the web, which depends on the time that passed between queries in a sequence of web searches. Relating multiple queries into one session according to log records can be done in various methods. The most common way is using the time gap between queries to combine them into sessions. Another session boundary detection method suggested is using the content of queries to reveal any topic change and thus start of a new session. We call the sessions that are combined using this approach as quests, and represent related methods in next Chapter 6. Moreover we check the relationship between these two query search concepts in Chapter 7.

The search engine logs are the only source to be used for session detection analysis. Most of the logs contain basic information such as: the query, the time it is issued, anonymous-ID, etc. But none of them point to start or end of sessions. These log entries compose a sequence of queries for each user. Thus there is needed a decision to combine them into sessions. The most common method is using a global threshold to verify the continuation of the session based on the time gap between the queries. In this chapter, we present the usage of such a method with different values as session threshold, but finally we claim that using a global threshold for all users is not appropriate, and there is a need for a personal threshold for each user, to fit best to the user activity. Other alternative detection methods can be found at Chapter 2.

5.1 Determining the Session Threshold

There are various methods in the literature to determine the session threshold. The most common and easy way is using one global threshold for all the users and assuming that intervals longer than this value point to session breaks. This method assumes that intervals can be classified into two groups, those that are shorter and are between activities in a session, and those that are longer and are between different sessions. The threshold length was set by different researchers to various values such as 5, 10, 15, and 30 minutes [27, 8, 10].

Another method is suggested by He and Goker [10]. They claim that the threshold should be set with regard to its effect on the number of sessions that will be identified, because if the threshold is too short then short sessions are separated into individual queries. While increasing the threshold value, the number of sessions is decreased and finally stabilizes. They finally used a global 10-15 minutes threshold. Huynh and Miller [12] provide a mathematical model for this idea.

Another idea is using a set of possible thresholds instead of one single threshold. This idea was implemented by Arlitt [1] while analyzing the world cup 98 site. There are also methods that try to find personal session thresholds for each user according to the time intervals in their activity. We present them in Section 5.3.

The determination of session thresholds affects many aspects of user behavior analysis, such as session length, division of the user activity into different sessions, and the number of queries in each session. Thus it has high importance. We try to see the effects of using a global session threshold on these parameters. When using a global session threshold for all users, choosing the right value can be done according to the time interval distribution, which shows the number of each different time interval between two adjacent queries of all users, see Figure 3.3. As we noted before, the curve in the figure is continuous and there are no breaks that explicitly divide the time intervals, so choosing any global threshold would not be perfect for all the users and may not identify the sessions correctly. We decided to choose 20 minutes as our constant global threshold since most of the search time intervals were less than 20 minutes.

5.1.1 Session Threshold and Number of Queries in a Session

Using our session threshold we can divide the continuous activity of users into different sessions with various lengths. This enables us to study the relationship between the session length and the number of queries in sessions. To find the distribution of number of queries in each session, we counted the number of different queries that occurred in each session for each user. Gathering this information we received the graph shown in Figure 5.1 (note the figure is in log-log scale). This shows that sessions with one query are the most common. Besides that most of the sessions include less than ten queries and as the number of queries in a session increases the number of such sessions is reduced.

Figure 5.2 (note the figure is in log-log scale) shows the survival probability of sessions to have more than a given number of different queries. We can see that less than 1% of the sessions would have more than 10 queries, and the chance that a user performs 20 queries is about 0.05%. Moreover this figure shows that users tend to have short sessions as 90% of the sessions contain three or less different queries.

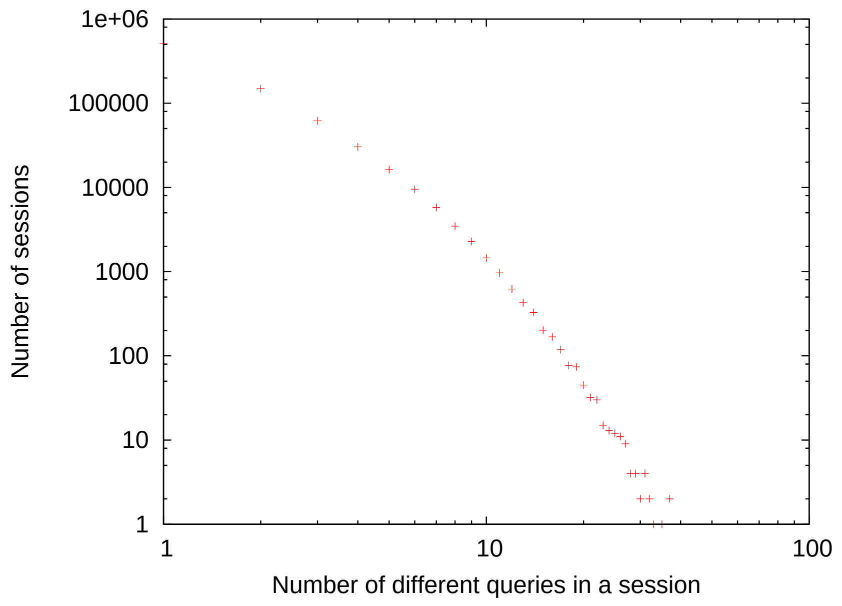

Figure 5.1: Different queries number distribution at sessions

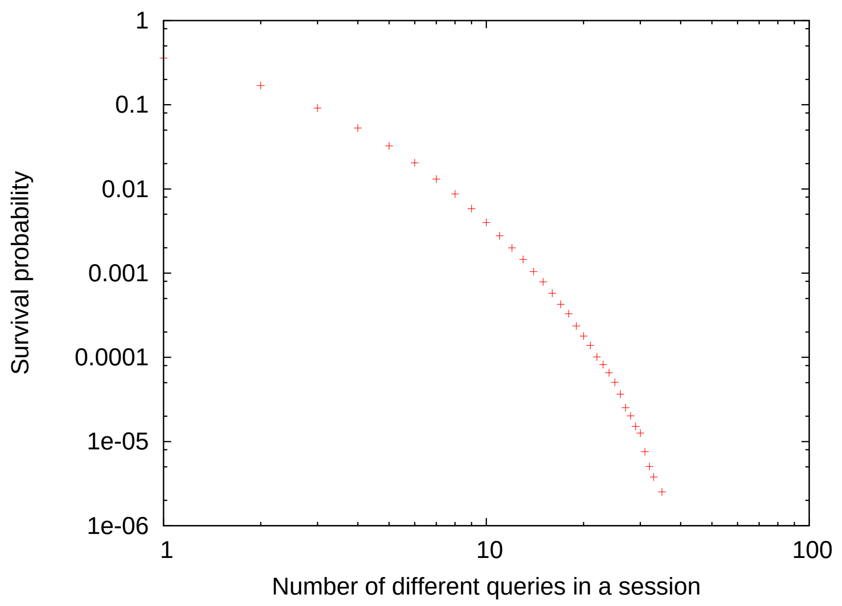

Figure 5.2: Distribution of different queries in sessions

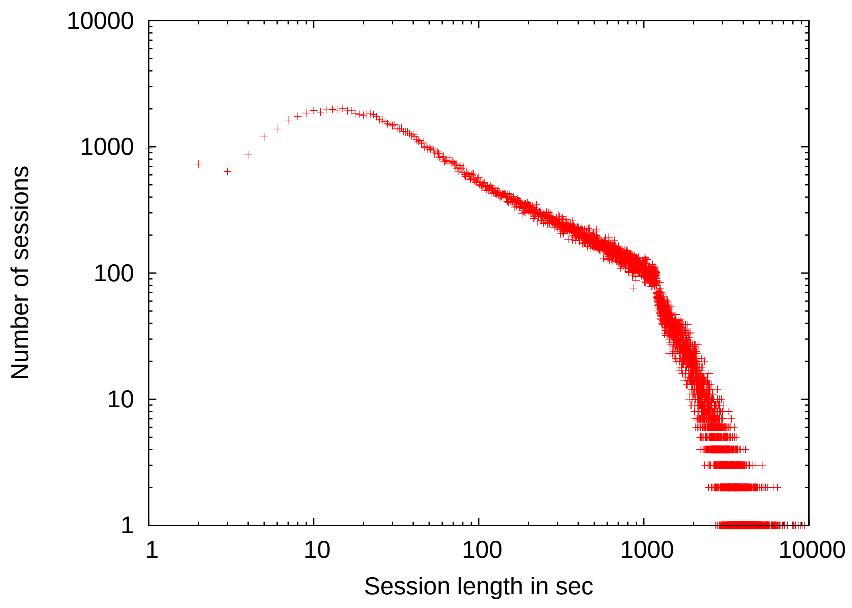

Figure 5.3: Sessions length distribution using 20 minutes (1200 sec) threshold

5.1.2 Session Threshold and Sessions Length Distribution

Another aspect that is affected by the session threshold is the session's length distribution. Using our session threshold (20 minutes) we tried to find the session length distribution of users, so we traversed one arbitrary part of the AOL log and saved for each user all of his different session lengths. See Figure 5.3 (note the figure is in log-log scale).

As we see in Figure 5.3, most of the session lengths are up to about one minute. This matches the observation we noted before that users tends to have short sessions. Moreover as the sessions get longer their popularity drops. This reduction continues in a natural way till we arrive to a break at sessions with a length of about 20 minutes. Having such a break at the value of the session threshold we used is suspicious and does not sound natural. After the break we observe a sharp decrease in the number of sessions.

The next Figure 5.4 shows the survival probability of having sessions with different lengths. We can see as before that most of the session lengths are short and less than 10% of sessions are longer than 20 minutes. About 40% of sessions are shorter than 10 seconds. The slow decrease in number of sessions continues to about 1200 second and after that there is a fast reduction.

5.1.3 Sessions Length and Number of Queries Correlation

Knowing the relationship between session length and the number of queries in sessions is interesting, so we checked the correlation between these two parameters. This would help us to verify if the session contains more queries as it gets longer. For each session of each user we kept its length and number of queries issued. Plotting these together we received the scatter plot of Figure 5.5 (note the graph is sparse and represent 20% of all information). We see a high concentration at low

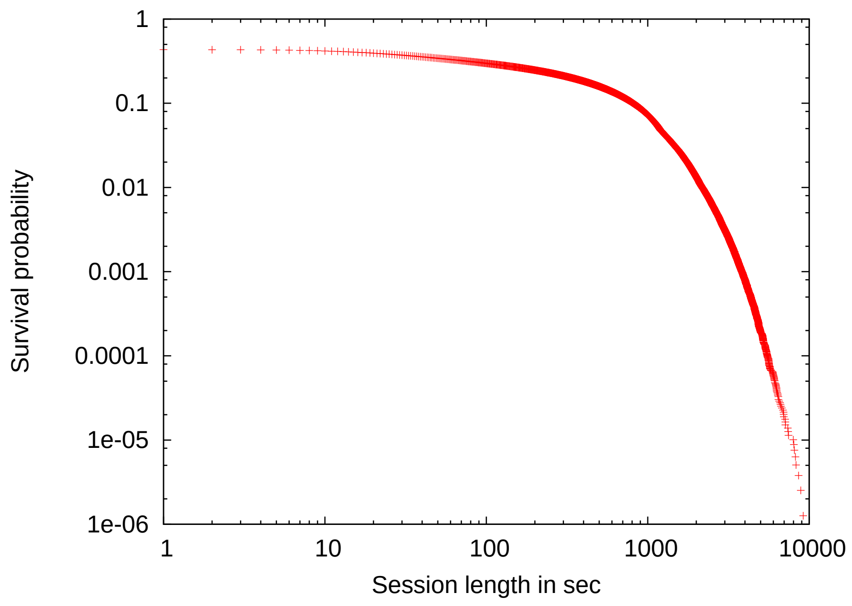

Figure 5.4: Percent of different length sessions using 20 minutes (1200 sec) threshold

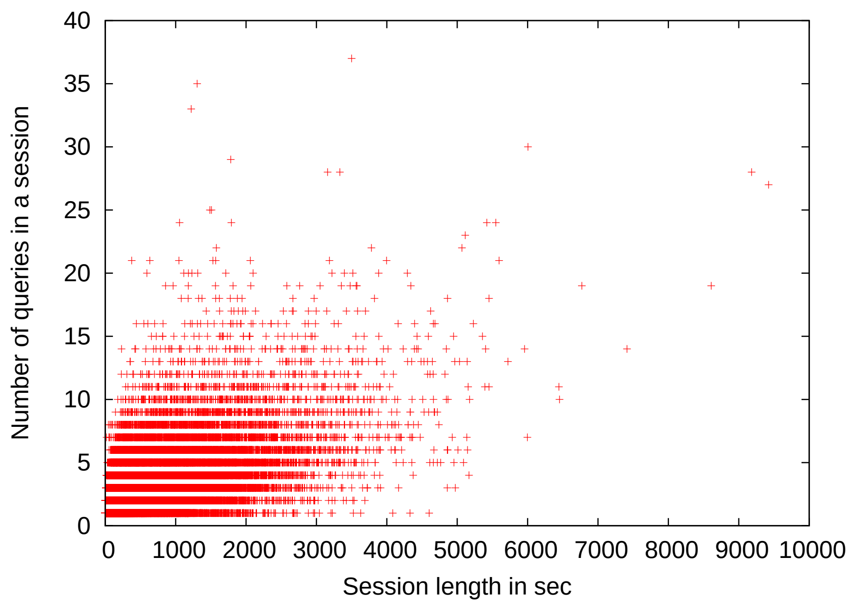

Figure 5.5: Correlation between session length and query number at sessions

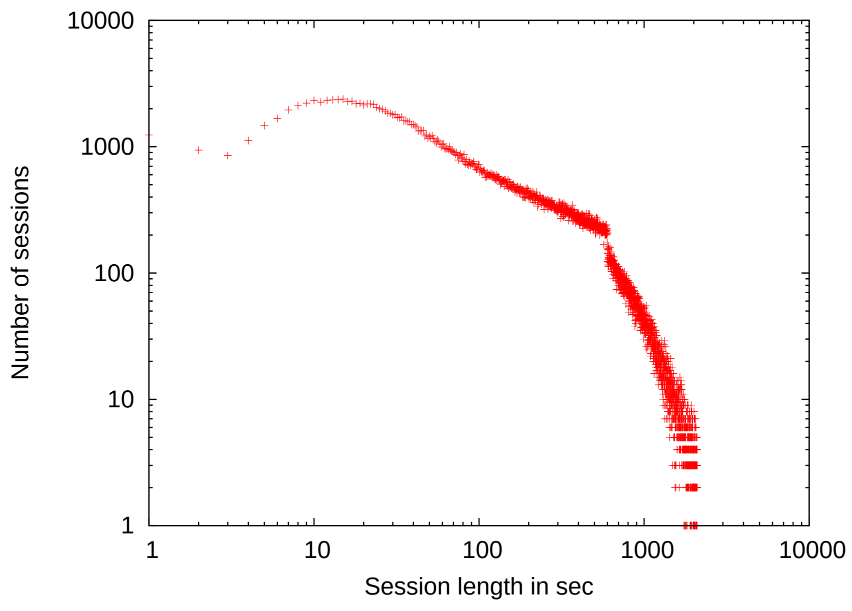

Figure 5.6: Sessions length distribution using 10 minutes (600 sec) threshold

values but it is hard to identify a linear relationship. Actually the correlation coefficient between these two parameters is too low and stands on 0.000335.

5.2 Global Session Thresholds and Session Length Distributions

Looking at the previous figures, especially the session lengths distribution of Figure 5.3, we see that setting the session length threshold at 20 minutes has a direct affect on the sessions length distribution: we can clearly distinguish the break at the 20 minute point while we expect to have a smoother transition at that point. To be sure that our assumption that this break is related to the session threshold is correct, we decided to check what happens if we change the session length threshold to 10 minute and 60 minute. In case that the break is part of the natural distribution we expect to see it in the same place even after changing the session threshold to lower and higher values. Figure 5.6 shows the session lengths when the session threshold is set to 10 minutes, and Figure 5.7 shows the same when the session threshold is increased to 60 minutes.

As shown in Figures 5.6 and 5.7, after changing the session threshold value to 10 minutes and 60 minutes the break point moved respectively towards 10 minute and 60 minute and did not vanish. The 10 minute graph is very similar to 20 minute graph except the break point which is earlier, but the 60 minutes graph is a little lower than the curve in the 20 minutes graph which means that rising the threshold caused there to be less sessions at each session length. In other words some separate sessions with the 20 minutes threshold are combined together with the 60 minute threshold and counted as one session. This shows that choosing the session threshold value

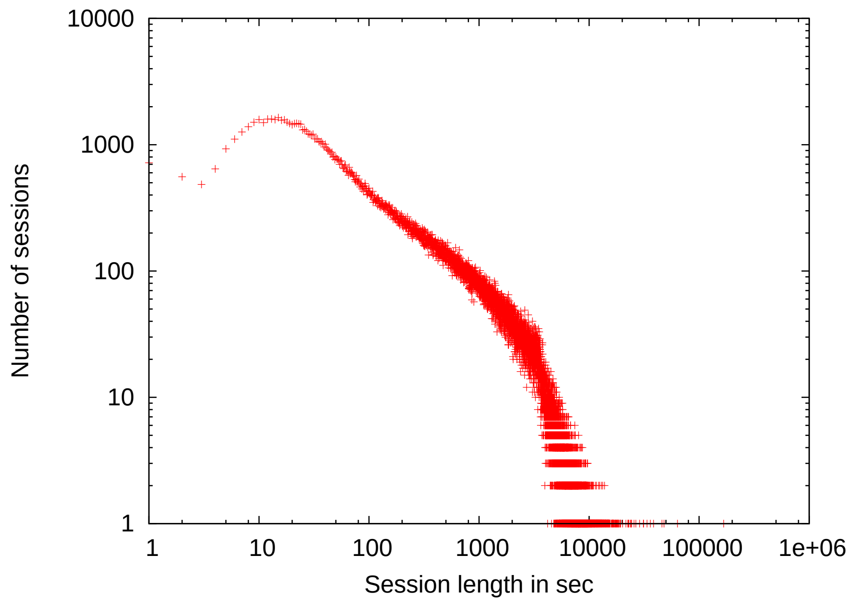

Figure 5.7: Sessions length distribution using 60 minutes (3600 sec) threshold

has a direct affect on the session length distribution and changing its value results in break point movement. This means that using a global session threshold with any single value would not give us a good session threshold as there would always be sessions that it would be too long or too short for them.

Figure 5.8 substantiates this claim. This figure shows the distribution of time intervals for some users that are classified into three different groups, such that for each group some different threshold is suitable. The X axis of these plots is the interval length, in a logarithmic scale. The Y axis is randomized, being used to displace the points relative to each other so as to expose their density.

For the users in the center column, it appears that 30 minutes is an appropriate threshold, while for the left-hand column a lower threshold may be better, and for the right-hand column we prefer a higher threshold. This motivates the idea of adjusting the thresholds and using a customized value for each individual user.

5.3 Suggesting Individual Session Threshold

We have seen that using a global session threshold is not appropriate for all users, so we thought to suggest an individual session threshold for each user. This idea was exploited before by Murray et al. [22], who proposed a hierarchical agglomerative clustering algorithm to find the session thresholds. In their algorithm, they first order the intervals in ascending order, then for each one find the quotient of the interval divided by the standard deviation of all shorter intervals that appeared before. The quotient is maximized for the interval which is significantly longer than the others, and the threshold is set between this interval and the preceding one. This algorithm may

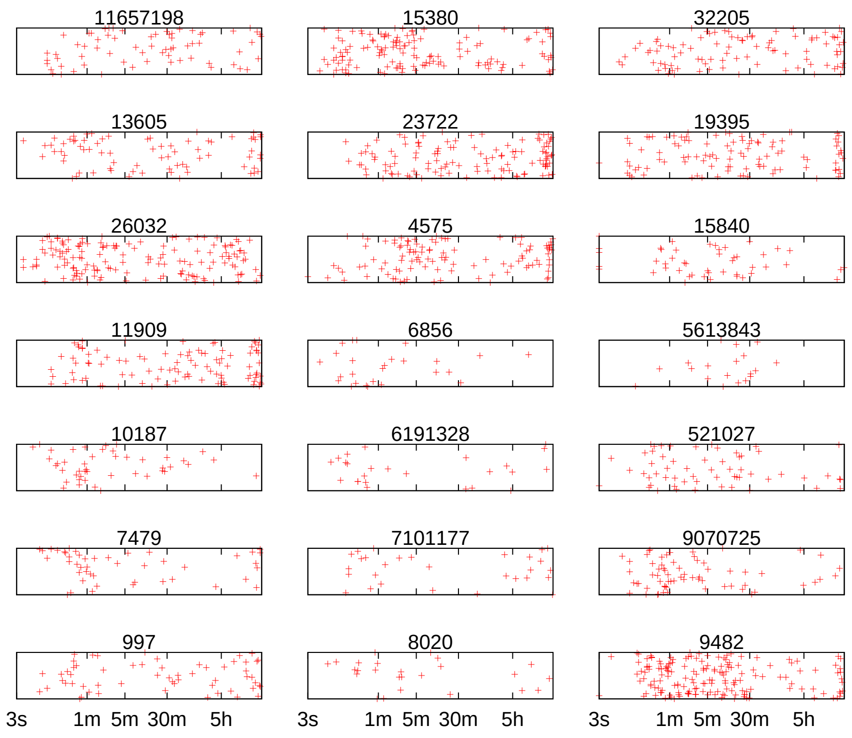

Figure 5.8: Texture plot of distribution of time intervals between queries

be problematic in case the maximum appears at unreasonable values such as a couple of minutes that is too short, or several hours which is too long. Moreover it is explicitly based on assuming a bimodal distribution of intervals, meaning short intervals with little variability and long ones also with little variability, so that the first long interval will maximize the variance set. But in case that long intervals are themselves varied, including time intervals of a few hours to a couple of days or weeks, there may be several high values such that the first one would not necessarily identify the first large interval.

To avoid the above mentioned problems, we suggested a simple algorithm, which codifies common sense and domain knowledge rather than relying on more abstract statistical parameters. Our algorithm also assumes that there is a bimodal distribution of time intervals, meaning many short intervals that represent gaps between actions within the same session, and then a group of longer intervals that represent session breaks, but we actually group the long intervals into sets in powers of two.

Our idea is to classify the user time intervals into groups of different length, and grade them according to how common they are while considering other parameters such as adjacency of common length interval groups, breaks (no activity) in interval groups, and continuous increasing or decreasing of sub groups of time intervals. We try to find a natural session threshold that expresses well the user session threshold in his session length distribution and fits it as best as possible.

As a first step we choose small group of users to study them in details. We retrieved 50 random users that had at least 20 queries to guarantee that each user had a minimal level of activity. As

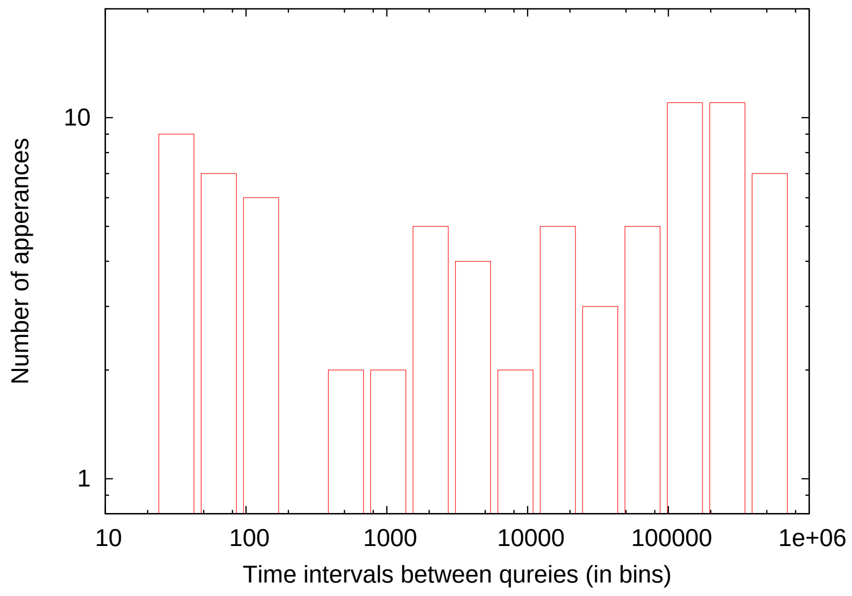

Figure 5.9: Time intervals matched to different length groups for user id: 997

before, we found the time interval between each two adjacent queries of each user, and the popularity of each of these intervals. We then divided the time durations between adjacent queries into groups of time intervals, starting at 32 second and increasing in powers of two, meaning our first time interval group starts at 32 second, next is 64 second and so on. Time intervals were matched to intervals based on the top bound, meaning time intervals that their length was below or equal to 32 seconds were assigned to the first group (32-group), those that were longer than 32 seconds but shorter or equal to 64 seconds were fitted to the second group (64-group) and so on. The obtained data was illustrated in graphs to have some visual help in the first steps of constructing an algorithm for determining the session threshold. Figures 5.9, 5.10, 5.11, 5.12 are some examples of the different graphs that were made for each of above 50 users. These graphs helped us to make better decisions between different versions of our algorithm in the development process.

Using the data that these graphs represent and concentrating on the behavior of different users, we suggest the following algorithm to evaluate an individual session threshold. Detailed pseudo-code of the algorithm is given in the following Subsection 5.3.1.

5.3.1 Individual Session Threshold Algorithm

Based on the data, we suggest the following algorithm for finding a suitable threshold. See Figure 5.13.

In our algorithm, we limited the potential session threshold to be between 32 seconds to 8192 seconds (about 2 hours and 16 minutes) to ensure a reasonable time interval as threshold. We refer to the number of intervals that fall in a group's range of time length as the group value.

For each user the algorithm grades each of his groups according to the max value of any group to its left and any group to its right (these groups start with 64 second and end with 65536 second).

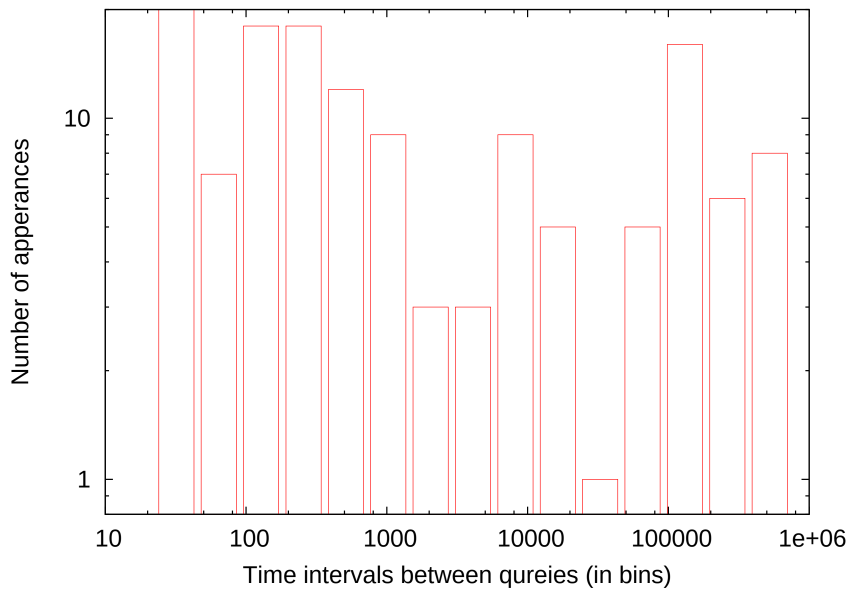

Figure 5.10: Time intervals matched to different length groups for user id: 15380

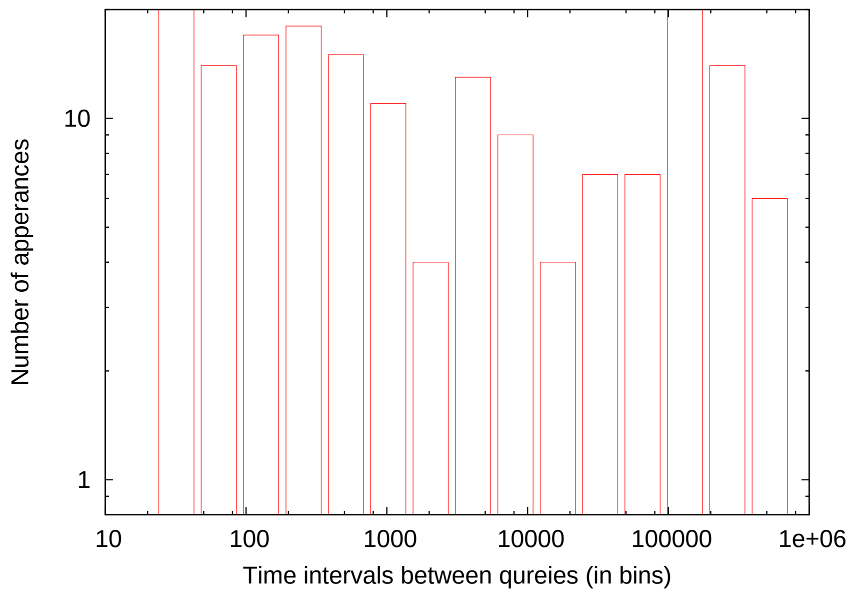

Figure 5.11: Time intervals matched to different length groups for user id: 8690

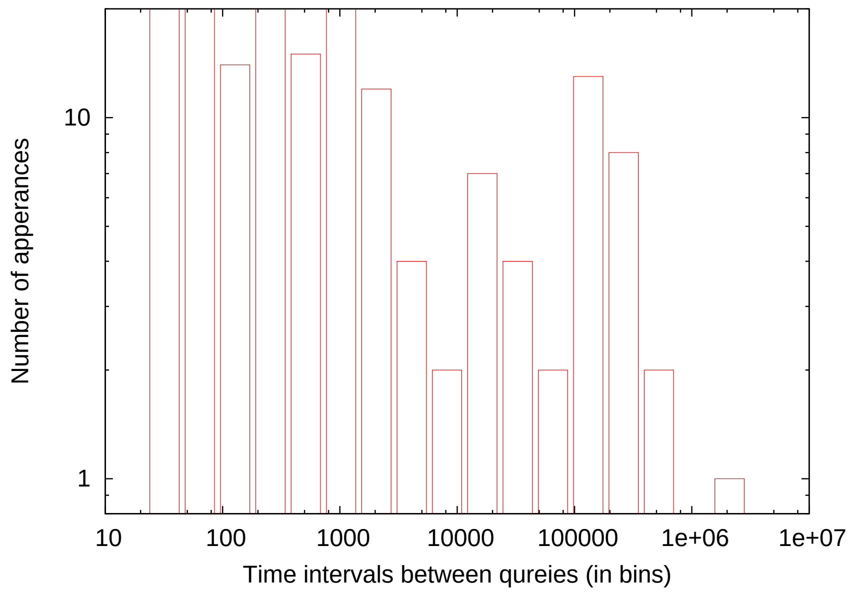

Figure 5.12: Time intervals matched to different length groups for user id: 9482

The algorithm looks for the minimum value between the short time intervals and the long time intervals. The group will have a higher grade as the difference is a bigger, since if there is a little difference between its left or right max it shows continued decrease or increase and then this interval is not the best threshold candidate. If the group value is zero, it means there were no two adjacent searches with interval time in this length range, the group is given the highest grade. Finally the algorithm assigns a grade to each group, and the group with the maximal grade is our suggested threshold for the user. In a case that two groups have the same value the algorithm suggests the group that is closer absolutely to 1200 seconds (20 minutes).

5.3.2 Individual Session Threshold and Sessions Length Distribution

We have run our algorithm on the log and found for each user his session threshold. Using these individual session thresholds we recorded as before all of the user's different session lengths, and found the session length distribution of all users, see Figure 5.14.

We can see here that the sessions length curve is smooth and there is no break at any specific place. This shows that using individual thresholds avoids dividing or connecting sessions by choosing non appropriate global short or long thresholds.

Figure 5.15 shows the probability of having sessions with different lengths. Using individual thresholds we have a lower probability for long sessions. About 50% of sessions are of one second.

```
Input: t[i] is timestamp of i'th query
// find intervals
for i=2..N
    d[i-1] = t[i] - t[i-1]
// create histogram
for i=1..N-1
    bin = 1
    lim = 32
    while (d[i]>lim)
        bin++
        lim *= 2
    hist[bin]++
// assign scores in desired range
for bin=5..9
    maxLeft = max( hist[2..bin-1] )
    maxRight = max( hist[bin+1..12] )
    score = 0;
    if (hist[bin] <= 2/3*maxLeft) score++;
    if (hist[bin] <= 1/2*maxLeft) score++;
    if (hist[bin] <= 1/3*maxLeft) score++;
    if (hist[bin] <= 1/6*maxLeft) score++;
    if (hist[bin] <= 2/3*maxRight) score++;
    if (hist[bin] <= 1/2*maxRight) score++;
    if (hist[bin] <= 1/3*maxRight) score++;
    if (hist[bin] <= 1/6*maxRight) score++;
    if (hist[bin] == 0) score += 5
    s[bin-4] = score
// find maximal score
lim = threshold = 512
max = s[1]
for bin=2..5
    lim *= 2
    if (s[bin]>max)
        max = s[bin]
        threshold = lim
if ((threshold==512) && (s[1]==s[2]))
    threshold = 1024
```

Figure 5.13: *Pseudo code of algorithm for setting the threshold.*

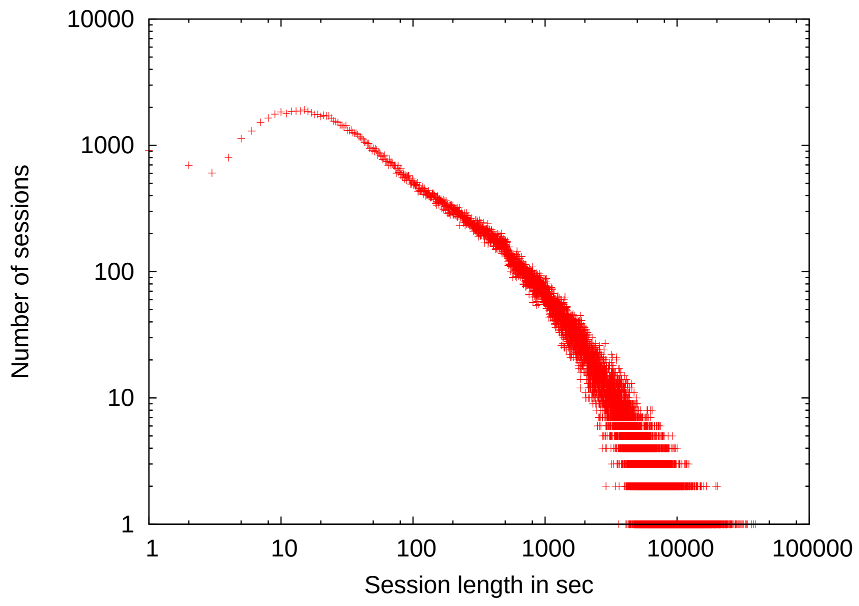

Figure 5.14: Sessions length distribution using personal thresholds

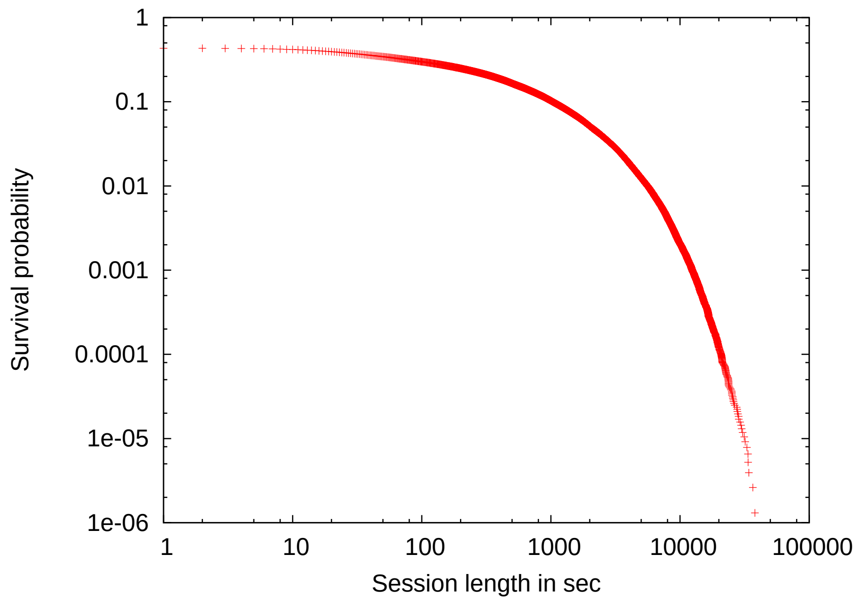

Figure 5.15: Percent of different length sessions using personal thresholds

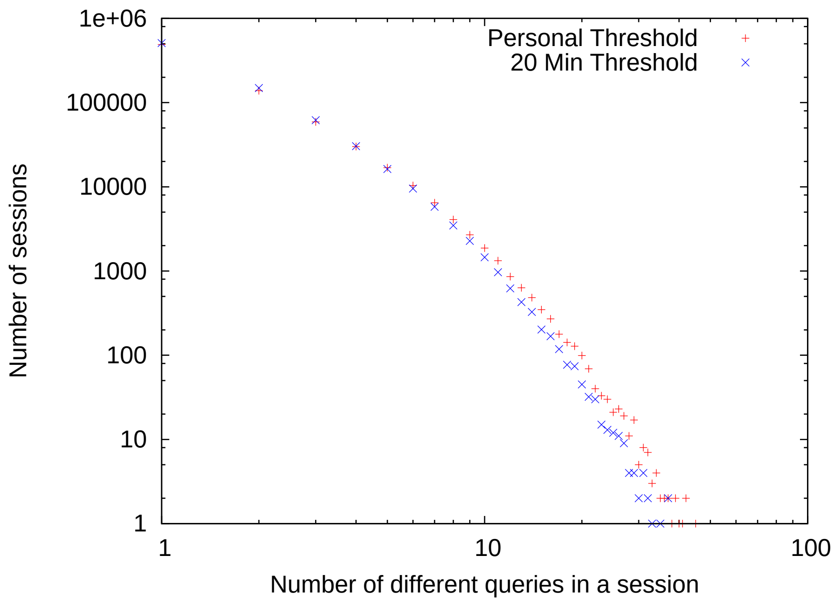

Figure 5.16: Different queries number distribution at sessions

5.3.3 Individual Session Threshold and Number of Queries in a Session

The Figure 5.16 (note the figure is in log-log scale) shows the distribution of number of different queries in the sessions using two different session detection methods, global threshold with value of 20 minutes and the individual threshold, and Figure 5.17 (note the figure is in log-log scale) shows the survival probability of sessions to have more than a given number of different queries using these two methods.

As we can see for low numbers of different queries, the global method is a little higher but as the number of the different queries is increased, the individual method has higher values. The difference in the low number of different queries between these two methods is then spread between the higher values in the individual threshold method.

Actually we do not expect great changes, since using the individual threshold algorithm mostly affects the sessions at the break point, the region with the session threshold value, and the rest of sessions are changed less.

Sessions Length and Number of Queries Correlation

The following Figure 5.18 (note the graph is sparse and represents 20% of the information) is the correlation between session length and query number for sessions using the individual thresholds.

As before it is not easy to identify a linear relationship between them, and the correlation coefficient is still very low and stands at 0.00046.

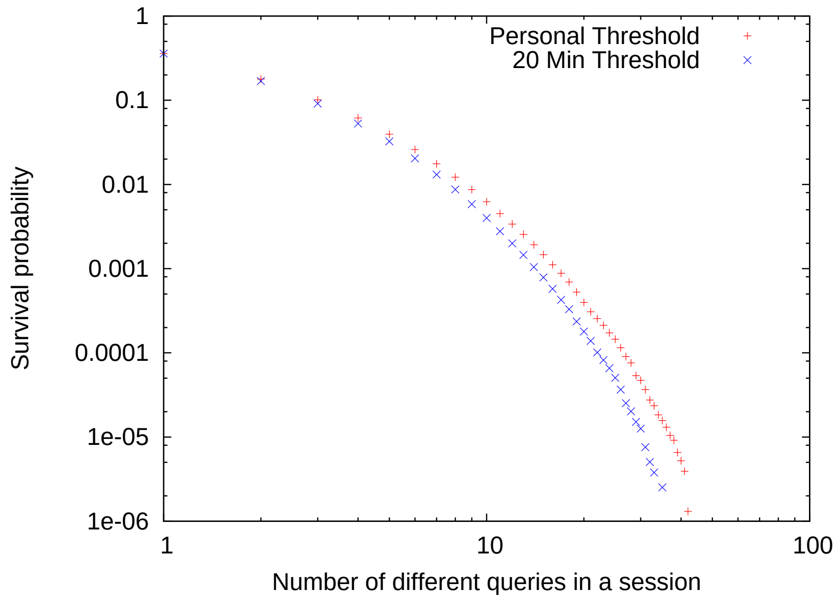

Figure 5.17: Distribution of different queries in sessions

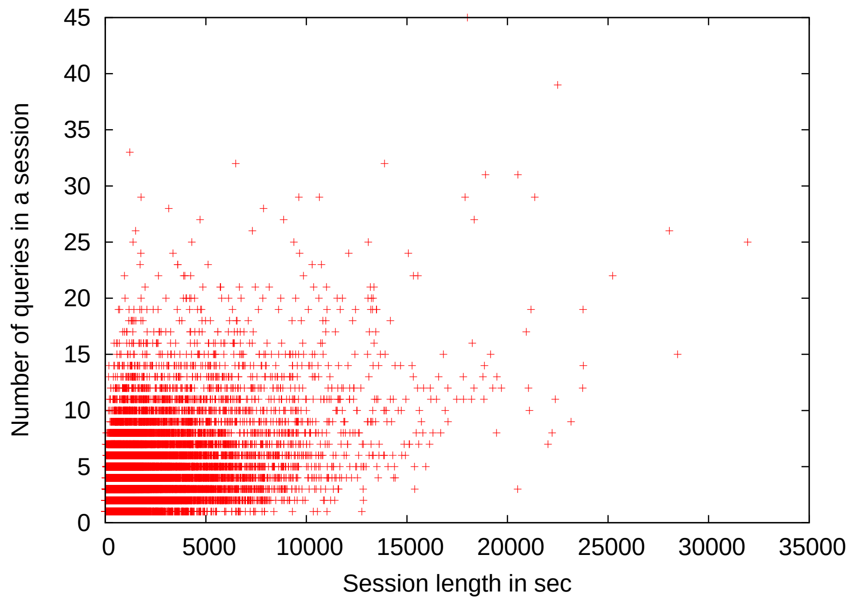

Figure 5.18: Correlation between session length and query number at sessions

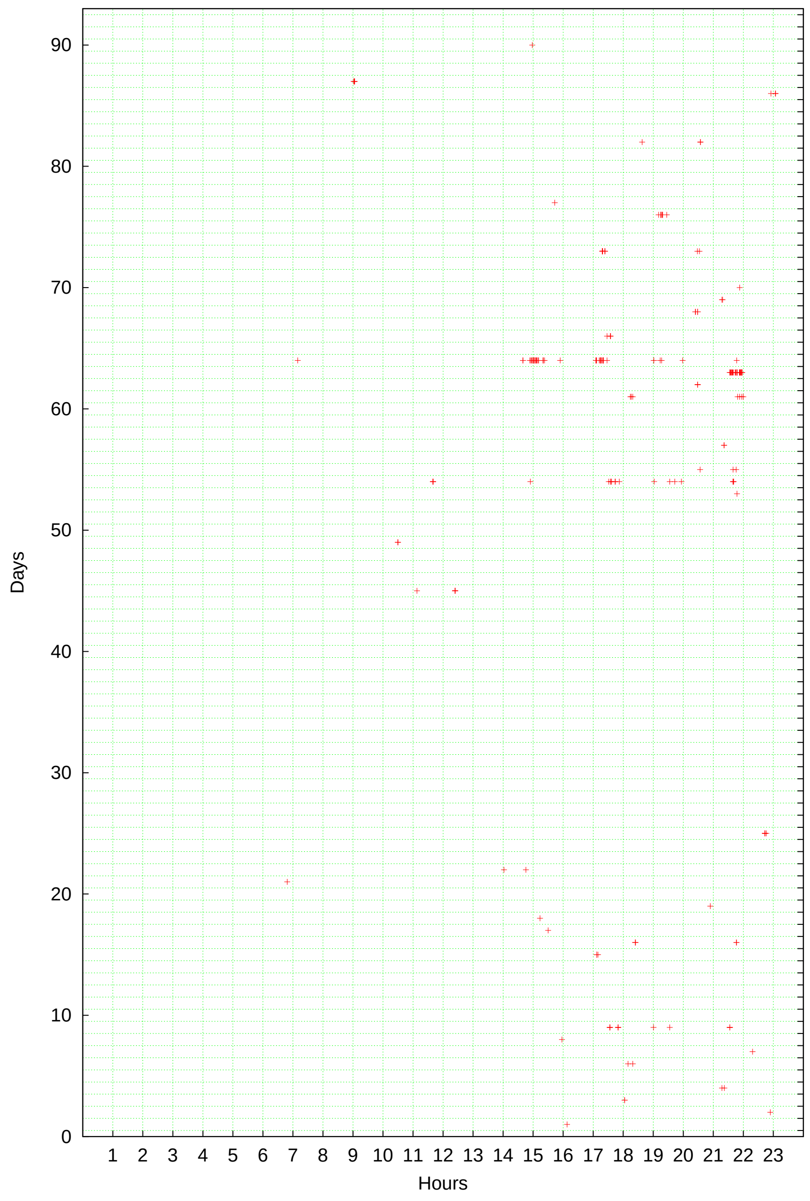

Figure 5.19: User activity (Id: 15380) at each day during three months log

5.4 Comparing Human Evaluation and Algorithm Suggestion

To try to evaluate the threshold determining algorithm, we compared its output with human decisions. We consider a group of 50 users that were chosen randomly. For each user, we created a graph that shows the user's activity during each day of the three months log. Each point on the graph represents one query of the user; some example users activity graphs are shown in Figures 5.19 and 5.20.

The inspector went over all of these graphs and tried to mark the intervals that seems to be session breaks. Two grades of 1 and 2 were used. 1 means a positive session break and 2 means

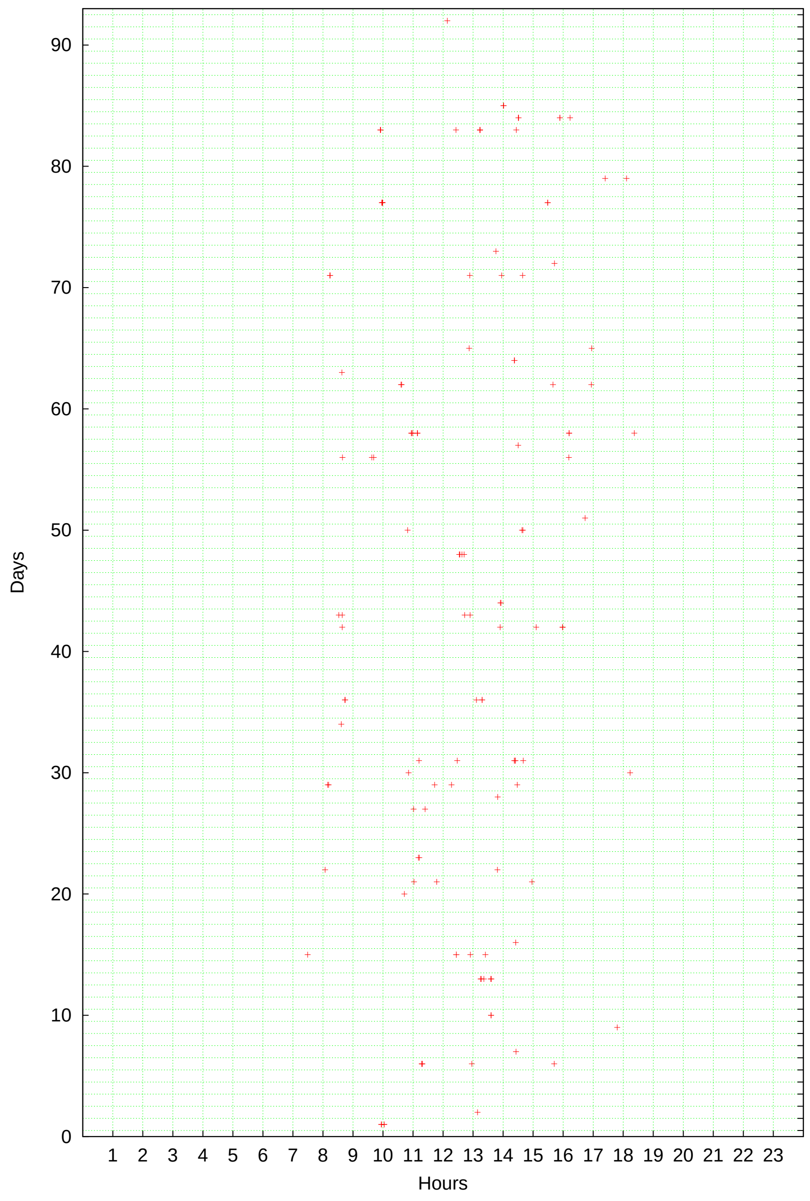

Figure 5.20: User activity (Id: 11909) at each day during three months log

that maybe it breaks, to verify the intervals. We used these markings to compare with session thresholds that we received from the algorithm. We found that out of 4992 intervals of all the 50 users we observed, we agreed with the algorithm in 1334 cases that they are session breaks (1307 of them were graded 1 and the rest were graded 2), we also agreed on 3384 cases as intervals within sessions. There were only two intervals that the inspector thought to be session breaks and the algorithm did not agree with him. On the other hand there were 270 intervals that algorithm considered as session break while our inspector did not agreed with them. Thus we got an 83% precision rate and almost 100% recall.

Chapter 6

Quests

Another idea used to determine the session boundaries is exploiting the content of the queries, that is based on lexical comparisons between the queries. With this method, a change in the topic of queries points to a session boundary (new session). In order to show the relationship between two consecutive queries, He et al. [11] classified the search patterns into eight different categories, where the applicable patterns are generalization, specialization, reformulation, repetition and new. The other patterns are browsing, relevance feedback, and others. Jansen et al. [18] used mostly these suggested search patterns. Lau and Horvitz [20] and Spink et al. [4] also proposed similar classifications.

To distinguish this from the previous way of detecting sessions using the time criteria, we propose the *quest* concept. We define a quest to be all actions taken by the user to satisfy a particular information need. This process could expand to a couple of days or even more, and in some cases can be periodic.

The way to find quests is to traverse the log entries and find the context switches. This is done using the similarity of adjacent queries, where the similarity rank could vary between 0 to 1 representing two different queries to two identical ones. These similarity ranks can then be used to create heat-maps which can show the search process of a user, as we do in Section 6.3.

6.1 Finding Quests from Queries

To be able to recognize quests in user activity, we need some criterion that can be applied to all the queries of a user. This criterion needs to quantify the similarity between queries. A lack of similarity can point to different quests. In order to be able to examine all the user activity, we found the similarity rank of a query with all other queries. This also give the opportunity to view the query development during the search process.

Before starting with the similarity evaluation process, we first removed all the stop words from the queries according to a stop words list from: http://tools.seobook.com/general/keyword-density/stop_words.txt. We also removed the following punctuation marks: [?;:!,.'"()-]. This gives us to the possibility to focus on the main portion of the query and get rid of nonsense phrases. Moreover, as the stop words tend to appear frequently in the queries, they can affect the similarity checking method, resulting in a similarity rate which does not reflect the real relation between queries.

We also removed all the repeated queries of each user, including cases that the user clicks more than one time for the same query search, or when the user checks the next result page for the same query.

The similarity rate can be determined using different kinds of evaluations. We present two different methods, one based on a simple containment rule and the other using common n-grams.

6.2 Containment Similarity Method

With this method, we used a simple process to recognize similarity between queries. Given a pair of queries, we split them on spaces into several terms (in a case that the query was more than one term) and checked for each term of the first query if it appears in the second one. We summed these appearances and divided them by the sum of terms of both queries, subtracting the number of shared terms (Jaccard coefficient). The calculated fraction is then the similarity grade for the pair of queries. Thus given query Q which is a collection of terms, after cleaning, the similarity between Q_A and Q_B is defined as follows:

$$J = \frac{|Q_A \cap Q_B|}{|Q_A \cup Q_B|} \tag{6.1}$$

Using these grades we made a heat-map graph for each user that shows his similarity grade for each pair of queries during the log activity. This makes it possible to observe the user's search process. We have done this for 200 random users out of the AOL log.

6.3 Quest Patterns

We tried to divide the heat-maps into different groups according to their visual shape and the level of the gray color. Each group points to an specific process of searching or kind of quest.

The basic differences between patterns reflect different patterns of repetitions and modifications between the queries. The heat maps show any repeats in the user queries or not.

6.3.1 Graphs with No Repeats

The most common kind of heat-maps is when the user has almost no repetitions in his queries. The user always submits a new query, so the typical figure has just a series of black squares on diagonal which are the result of comparing the query to itself. See Figure 6.1.

6.3.2 Graphs with Repetitions

Most of the users have some kind of repetitions in their queries during the log activity. These query repetitions can be categorized in different groups.

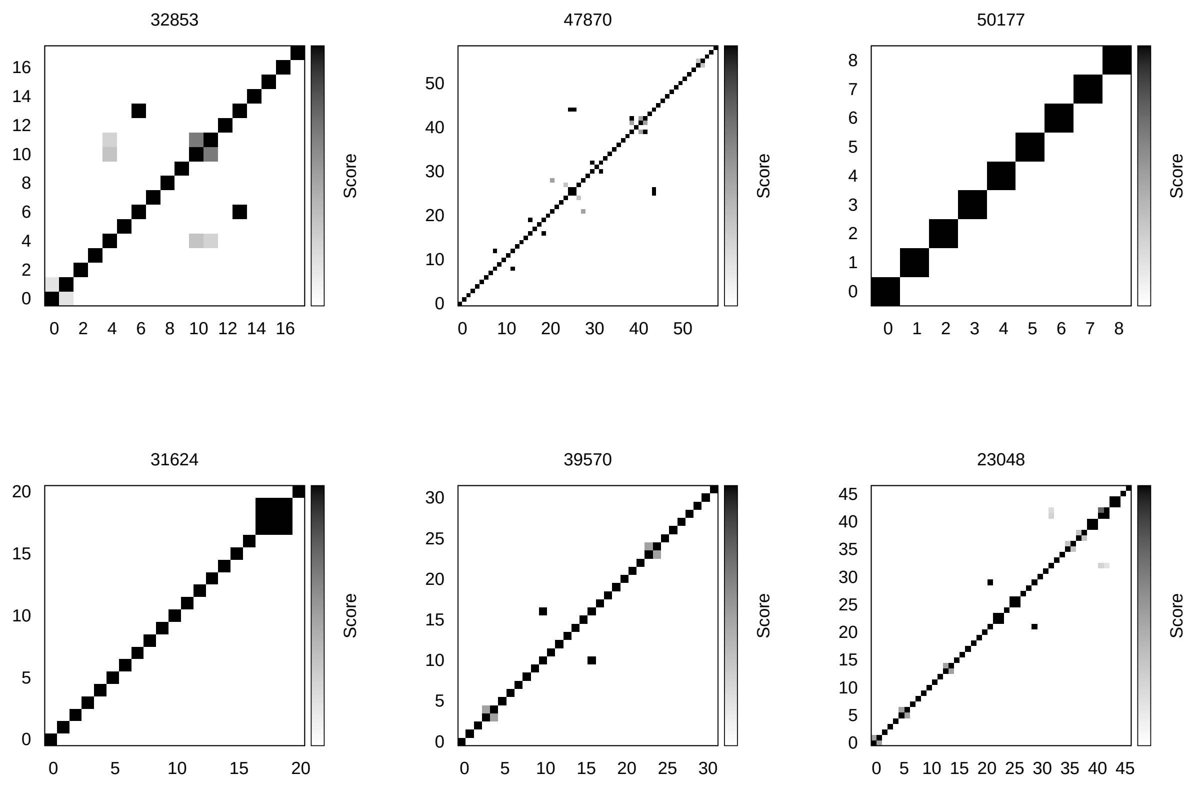

Figure 6.1: Heat-maps with few or no repetitions

- **Sword Shape of Squares**
  Heat maps that contain a shape similar to a sword (X-shape), with two symmetric lines connected to the diagonal, reflects situations where the user goes back to his previous queries and checks them again. Of course these graphs can also contain other shapes that we talk about later. See Figure 6.2.

- **Black Series of Vertical/Horizontal Squares**
  Another kind of repetitions can appear as a vertical/horizontal series of black squares (they are the same since the graph is symmetric). In this case the user returns to his previous queries at different intervals. If the pattern is regular, this is interpreted as a periodic quest process. The actual intervals can vary from hours to days. Thus periodic queries can arise from checking the news pages every few hours or checking the weather each morning or checking the bank account daily or weekly. Note that the heatmaps only show the serial numbers of the queries. See Figure 6.3.

  The vertical/horizontal series can be combined together and create big squares along the diagonal or at any other place of the map. These combined squares can point to a navigational or interest field quest.

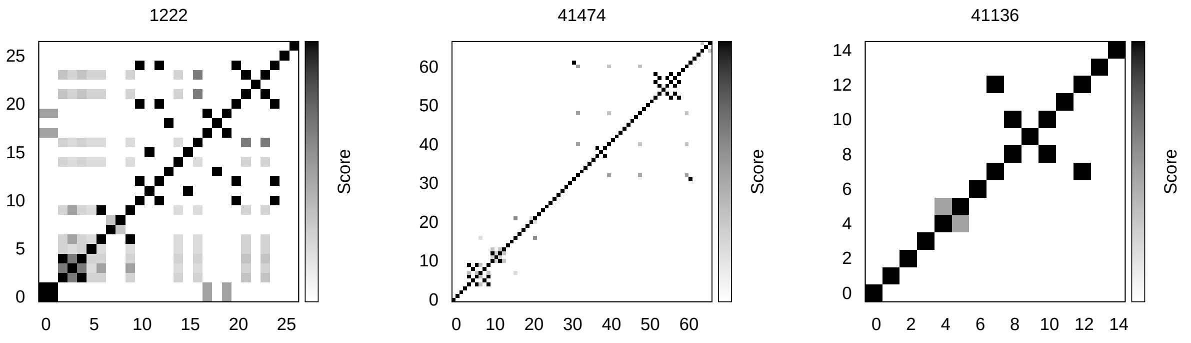

Figure 6.2: Heat-maps with sword shape of squares

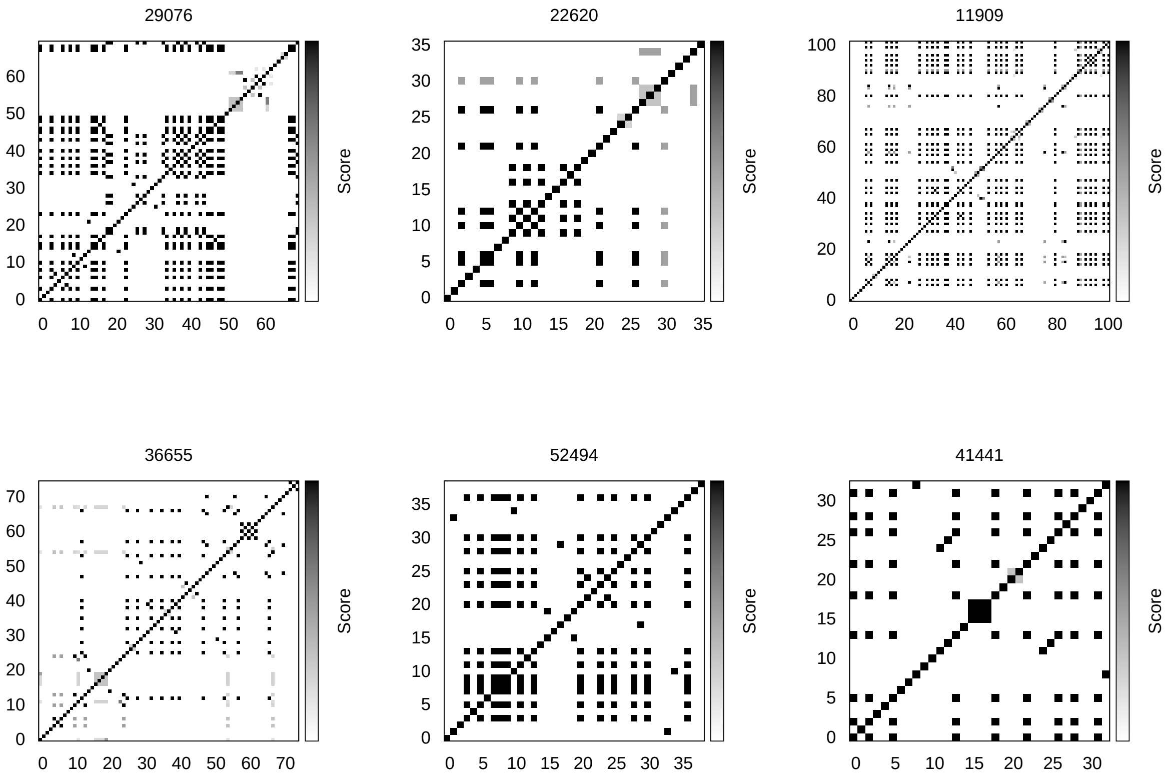

Figure 6.3: Heat-maps with black series of vertical/horizontal squares

Date	Time	Query
2006-03-29	08:11:15	yahoo
2006-03-29	11:43:20	love animals
2006-03-29	14:28:28	weddingchannnel
2006-03-30	10:51:05	yahoo
2006-03-30	18:13:42	washingtonpost
2006-03-31	11:12:17	yahoo
2006-03-31	12:27:57	petfinder
2006-03-31	14:22:15	yahoo
2006-03-31	14:24:11	dc coast
2006-03-31	14:40:13	wwwaetna
2006-04-03	08:36:31	yahoo
2006-04-05	08:44:24	yahoo

Table 6.1: User id 11909, Navigational usage of search engine

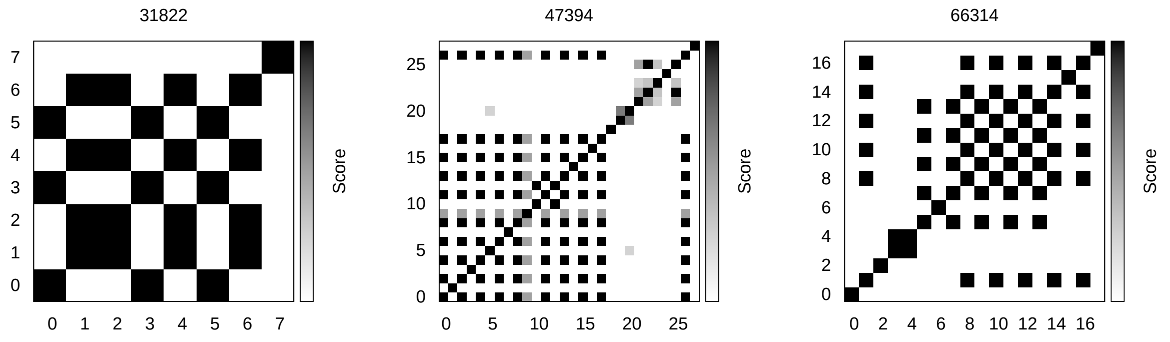

Figure 6.4: Heat-maps with chess board pattern of squares

For example, user 11909 from Figure 6.3 shows a classic use of search engine as a navigational tool, most of queries of this user is ”yahoo” which is repeated again and again. Table 6.1 shows part of the related cleaned log queries.

- **Chess-board**
  One of the interesting shapes created are figures that look like a chess board. This shape is created when the user searches alternately two different queries. See Figure 6.4.

  Table 6.2 and 6.3 provide examples of the cleaned log of users with ids 31822 and 66314 to see their queries.

- **Big Square Shape**
  Another pattern is the big squares with gray colors at different rates, which sometimes contains also black cells. These sequences show the process of updating queries, where the

Date	Time	Query
2006-03-01	14:16:25	disney channel
2006-03-14	17:24:00	missoandfriends
2006-03-16	19:34:41	missoandfriends
2006-03-17	17:26:26	disney channel
2006-03-23	09:26:04	missoandfriends
2006-03-24	16:51:09	disney channel
2006-03-24	17:09:20	missoandfriends
2006-05-28	19:53:46	dear best friend

Table 6.2: User with id 31822, Examples of alternate search

Date	Time	Query
2006-03-01	17:35:04	msncom
2006-03-05	20:01:58	bank
2006-03-05	20:04:47	checks online
2006-03-08	17:20:57	msn
2006-03-09	11:22:52	msn
2006-03-16	20:41:05	usbank
2006-03-18	17:57:48	furniture stores littleton
2006-03-20	14:33:55	usbank
2006-03-24	15:14:08	bank
2006-03-24	17:54:46	usbank
2006-03-29	16:33:43	bank
2006-03-30	15:20:17	usbank
2006-03-31	19:11:20	bank
2006-04-02	13:47:54	usbank
2006-04-03	11:20:41	bank
2006-05-04	11:51:54	msnom
2006-05-08	00:54:04	bank
2006-05-18	12:19:16	pool pak units

Table 6.3: User with id 66314, Examples of alternate search

user edits his query each time until he finds the desired page or stops searching. Thus gray squares can reflect quest development. See Figure 6.5.

For example if we take a closer look at user id 49223, he has a big gray square in the middle of his heat-map activity. This is created as a result of editing the query ’manhatten parking plaza garage’, in different order of terms with some misspelling in terms. Table 6.4 shows the related cleaned log queries.

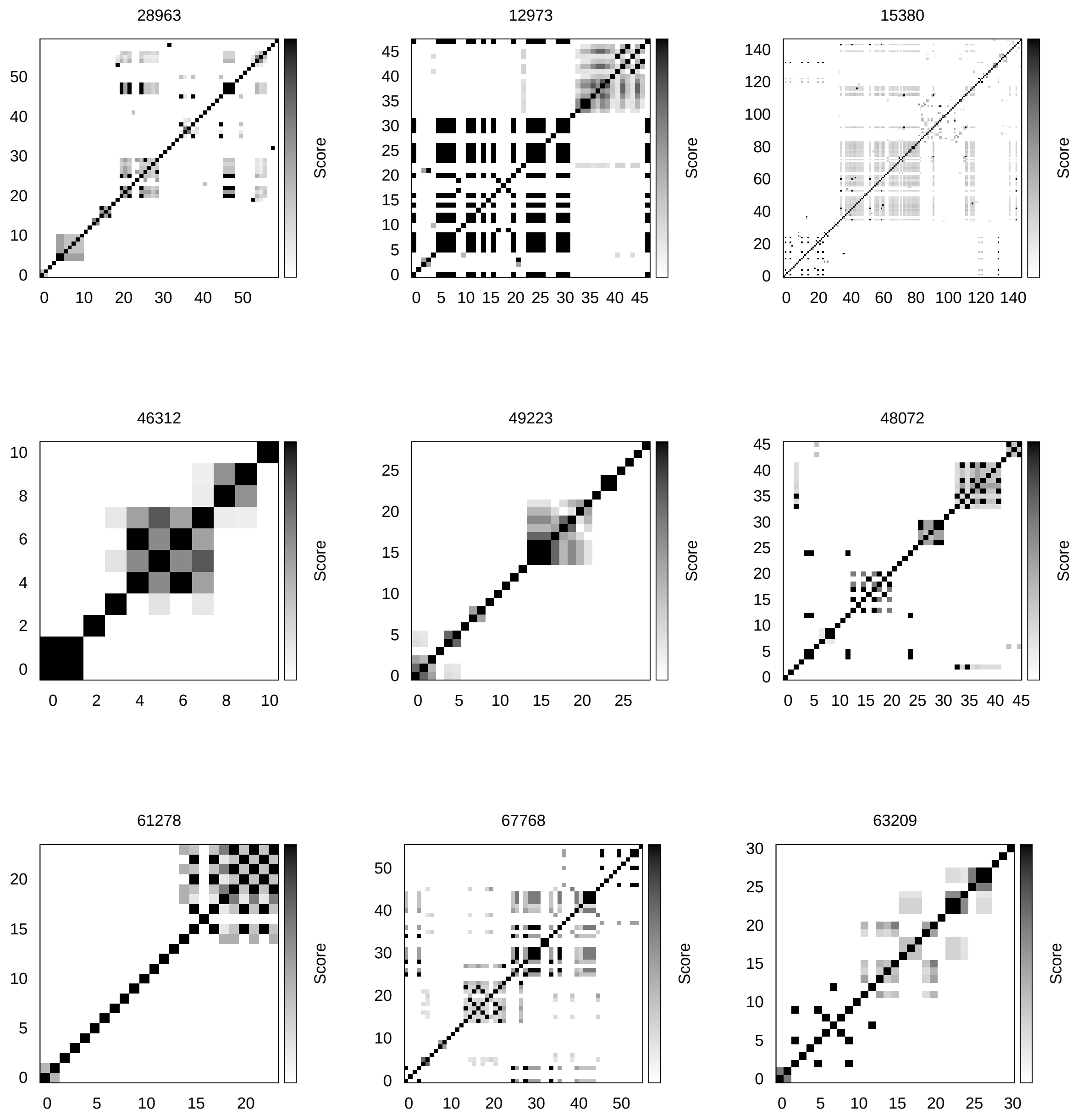

Figure 6.5: Heat-maps with big square shape

Date	Time	Query
2006-04-22	16:32:12	manhatten parking plaza garage
2006-04-22	16:34:44	manhatten plaza parking garage
2006-04-22	16:35:06	manhatten parking plaza garage
2006-04-22	16:35:37	manhatten plaza garage
2006-04-22	16:36:09	manhattan plaza garage
2006-04-22	16:46:57	manhattan plaza parking garage
2006-04-22	16:52:07	manhatten parking garages
2006-04-22	16:54:51	manhattan parking garages

Table 6.4: User id 49223, Examples of editing queries

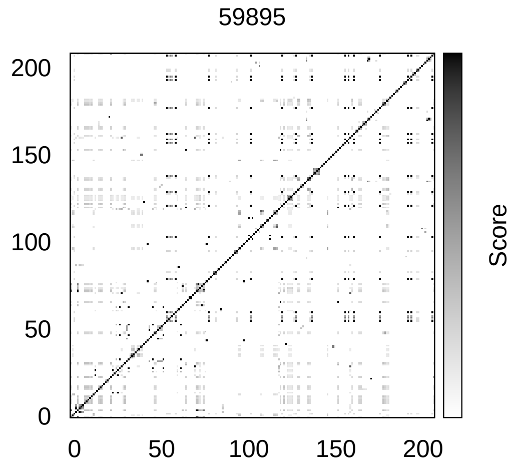

Figure 6.6: User id 59895

- **Field of Interest**
  The activity of user 59895 in Figure 6.6 can be interpreted as interest field quest, this user adds the ”golf” term to almost all his queries. This common subject in the queries causes all the gray squares at the heat-map. Table 6.5 shows some cleaned queries of this user.

  The id 61151 user search activity in Figure 6.7 also can be categorized as interest field quest, this user tends to connect the ”picture” term to his queries during the search. Table 6.6 shows some cleaned queries of this user.

Date	Time	Query
2006-03-02	07:51:07	golf shoes
2006-03-02	07:52:21	mens necklaces
2006-03-02	07:57:42	mens black necklaces
2006-03-02	08:01:14	mens black chain necklaces
2006-03-03	21:54:35	izzo golf
2006-03-03	21:56:49	proactive golf
2006-03-03	22:06:13	charter golf
2006-03-03	22:12:02	golf cart covers
2006-03-03	22:30:12	taylormade golf

Table 6.5: User id 59895, Combination of 'golf' term with queries

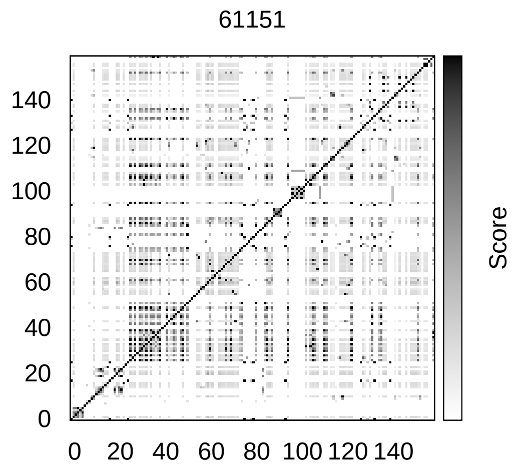

Figure 6.7: User id 61151

6.4 N-gram Similarity Method

Using the previous method for recognizing repeats is not always successful since this technique is simple and just checks if any terms that are separated with spaces appear in the second query. However there are many cases where users modify their searches by using similar words instead of just adding or deleting words. It is obvious that we can not uncover such modifications using this method, so we decided to find the similarity between the queries by using the n-grams method. With this method we create all substrings with length n of a pair of queries, and then look for shared n-grams and grade them as previously (shard n-grams divided by total n-grams). The choice of parameter n affects the results. We preferred to use 4-grams since we believe that using 2-grams or 3-grams will not really reveal the repeats ideally since their length is too short, and there would be

Date	Time	Query
2006-03-19	15:57:14	pictures shadows
2006-03-19	16:14:07	pictures synyster gates
2006-03-19	16:25:44	pictures johnny christ
2006-03-19	16:35:12	pictures reverend avenged sevenfold
2006-03-19	18:19:49	pictures zacky vengaence
2006-03-19	18:37:47	pictures avenged sevenfold
2006-03-20	14:55:59	pictures patrick stump
2006-03-20	15:23:17	pictures pete wentz naked
2006-03-20	15:40:21	pictures joe trohman

Table 6.6: User id 61151, Combination of ’pictures’ term with queries

Date	Time	Query
2006-03-11	11:43:10	y1007miami
2006-03-11	11:45:01	y1007maimi
2006-03-11	11:48:21	y1007 miami
2006-03-11	11:48:38	y1007miami
2006-03-11	11:51:54	y1007 miami
2006-03-11	11:55:05	y1007miami
2006-03-11	13:08:13	y100 7 miami
2006-03-11	13:11:17	y100 7miami

Table 6.7: User id 29941, Using connected or separated terms in queries

too much similarity between queries that are not related to each other. Using 4-grams we leave the terms with length less than or equal to four characters, and for the longer terms we create all their 4-grams. Using this method, we created the similarity graphs again and observed some changes in the graphs. Most of graphs now have new gray regions that indicates the relationship between the queries that were not revealed before. This improvement will help us to classify the users.

Figures 6.8, 6.9 show examples of changes in heat-maps because of searching the queries with separated terms or n-grams. Table 6.7 shows the related part of cleaned queries in user Id 29941 heat-map that caused the new dark square,in which the n-gram method succeed to reveal their relationship.

Using n-gram method, it is possible to observe the query editing process earlier. We also see that the dimensions of the squares get bigger by joining small gray squares together.

Comparing both grading methods, the n-grams method tends to give higher grades at non identical queries which is expressed by darker shade of gray color as it has the advantage of cutting the queries into small chunks.

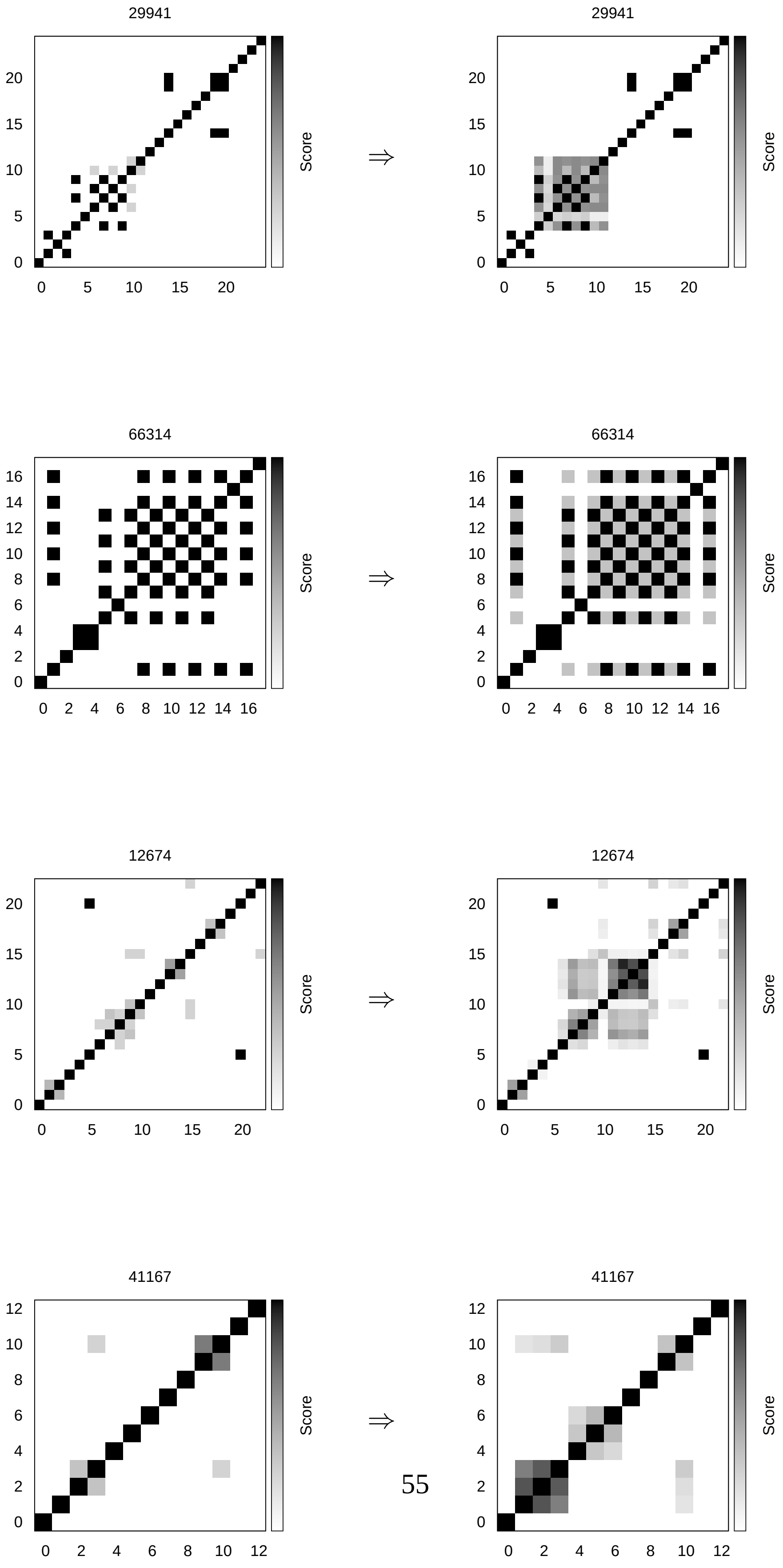

Figure 6.8: Comparison between containment and n-grams methods heat-maps

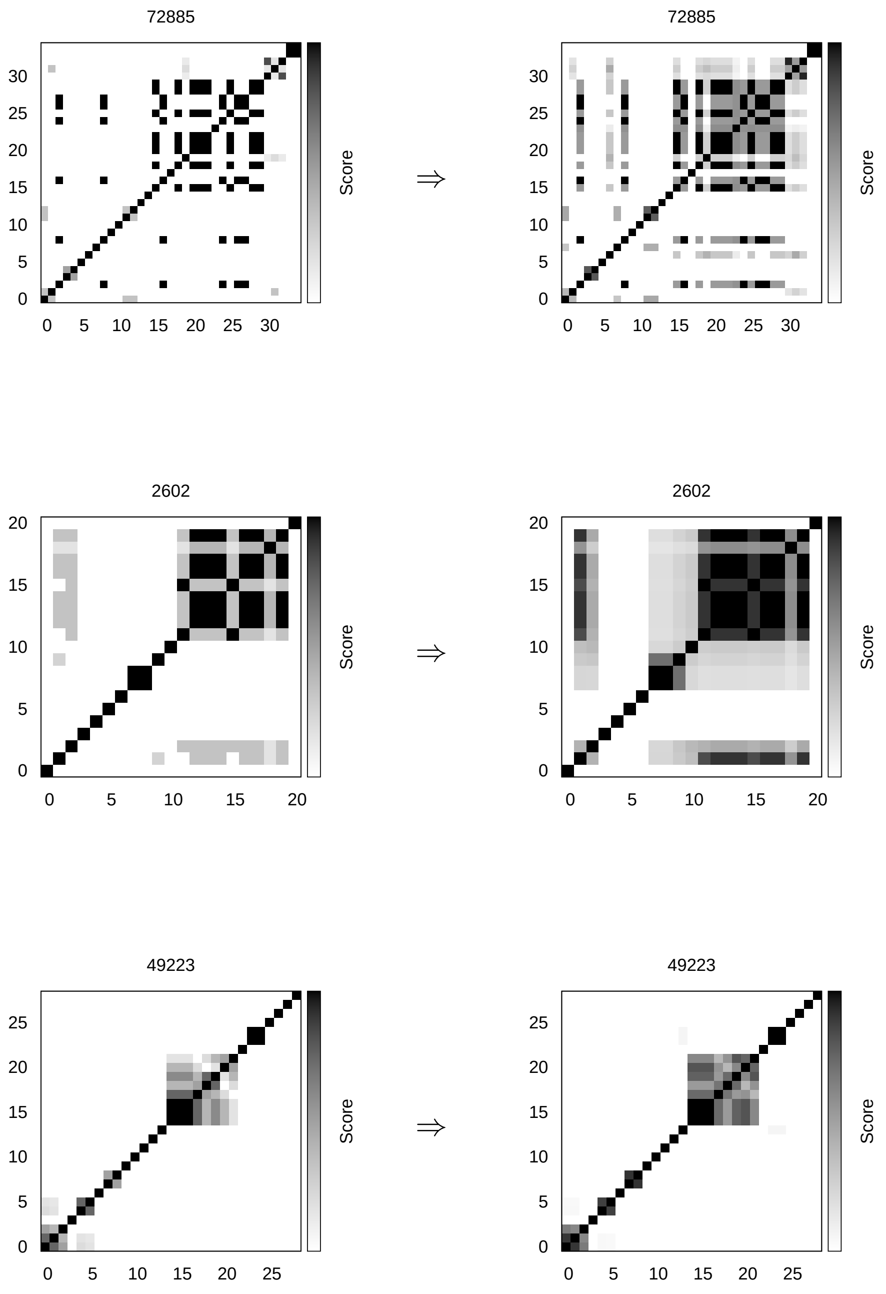

Figure 6.9: Comparison between containment and n-grams methods heat-maps

6.5 Robots and Quests

As we mentioned at the beginning, our research is based on users that are supposed to be humans. This was assured by removing from the AOL log file the users that were supposed to be robots and also those users that were not surely classified to be humans according to Duskin's work [9]. It may be interesting to see the search patterns created by robots. Thus we repeated the process of finding quests using the n-gram method. This enables us to compare humans and robots in terms of query modification. In other words we can reveal if robots issue related queries with changes between them, or only repetitions of the same queries.

It is worthwhile to remember that before starting with the similarity evaluation process all the stop words and punctuation marks are removed from the queries. Moreover, we also removed all the repeated queries of each user, in case that the user clicks more than one time for the same query search or when the user checks the next result page for the same query. So each square in the heat-maps that represents one query, can actually represent a sequence of identical queries.

We preferred to use the n-gram method to evaluate the similarity between the queries, since it reveals more precisely the relationship between the queries. As before we set the n parameter to four. The similarity method grades the queries after they are cleaned, so it may happen that a pair of queries are not exactly the same and thus considered as different queries but, after removing the stop words and punctuation marks they became exactly the same.

After conducting the similarity evaluation and creating the heat-maps we found some unique robot patterns. We expected to have similar patterns for all the robot users, but we noticed different patterns that some of them were also more appropriate to human users. Here we present some examples of them. These patterns are classified according to their shape.

One pattern that occurred for some robot users was the appearance of black squares with big dimensions in the heat-maps. This can be explained by the fact that distinct queries are identified before the cleaning process, but the grades are based on the similarity between the pairs of cleaned queries. So sequences of queries with small differences that are cleaned away are graded maximally and thus create these big squares in the heat-maps. See Figure 6.10. Table 6.8 shows part of the log of user Id: 8591867 that caused the big black square.

Another pattern is a very dense map, like those shown in Figure 6.11. Most of these point to two or more processes of searching terms that are interleaved together. In some cases this may actually be strange human behaviors. We had a closer look on the query logs of some of these users.

For example user Id 986909 in Figure 6.11 which is considered a robot mostly searches for ”text twist”, but it also searches for other queries in between and even checks different suggested URLs for them. Moreover its time intervals are obviously longer than a few seconds.

User Id 2004199 in Figure 6.11 mostly searches for the site ”stickyhole”, and has some other queries where he went through some of the suggested URLs. We think it is more likely a human than a robot, although he mostly searches the same query.

User Id 3245567 that is represented in Figure 6.11 mostly searches ”indiana state university” but his log contains other queries with some sporadic clicking. Although this user has a heavy use of mostly one query, it can hardly be marked as a robot.

Finally, we also observed some patterns that we think may not surly represent a robot user as

Date	Time	Query	Cleaned New Query
2006-04-08	03:04:15	www.yahoo.profiles	yahoo profiles
2006-04-08	03:04:36	yahoo.profiles	yahoo profiles
2006-04-08	03:04:39	www.yahoo.profiles	yahoo profiles
2006-04-08	03:04:42	yahoo.profiles	yahoo profiles
2006-04-08	03:04:46	www.yahoo.profiles	yahoo profiles
2006-04-08	03:04:47	www.yahoo.profiles	(repetition - ignored)
2006-04-08	03:04:47	yahoo.profiles	yahoo profiles
2006-04-08	03:04:48	yahoo.profiles	(repetition - ignored)
2006-04-08	03:04:49	www.yahoo.profiles	yahoo profiles
2006-04-08	03:04:54	www.yahoo.profiles	(repetition - ignored)
2006-04-08	03:04:54	yahoo.profiles	yahoo profiles
2006-04-08	03:04:55	yahoo.profiles	(repetition - ignored)
2006-04-08	03:04:56	www.yahoo.profiles	yahoo profiles
2006-04-08	03:04:58	yahoo.profiles	yahoo profiles

Table 6.8: Some part of User Id 8591867 that cause the big black square

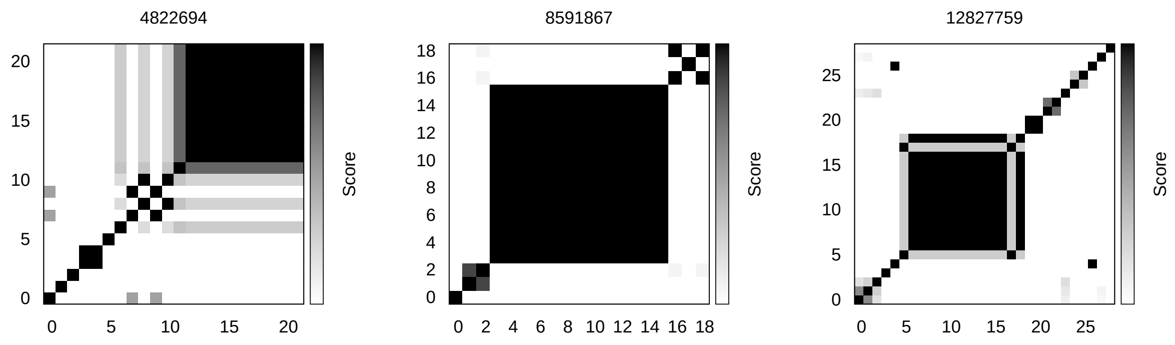

Figure 6.10: Robot users with big black squares

we had seen these patterns before while conducting the same analysis for human users. See Figure 6.12 for examples.

However it is worthwhile to mention that the users with similar heat-map patterns to human users mostly had very short time intervals, and moreover they have long sequences of queries that were repeated without any editing. This massive repetition of previous queries raises the suspicion that these users are indeed not human but rather robots.

We observed some sporadic clicking on the search result and some editing of queries for the robot tagged users, and on the other hand we saw that some time intervals were too short for the human tagged users. This raises the question if these users are correctly classified as robots or not. This may indicate that the division of users into three groups of humans, robots, and unclassified

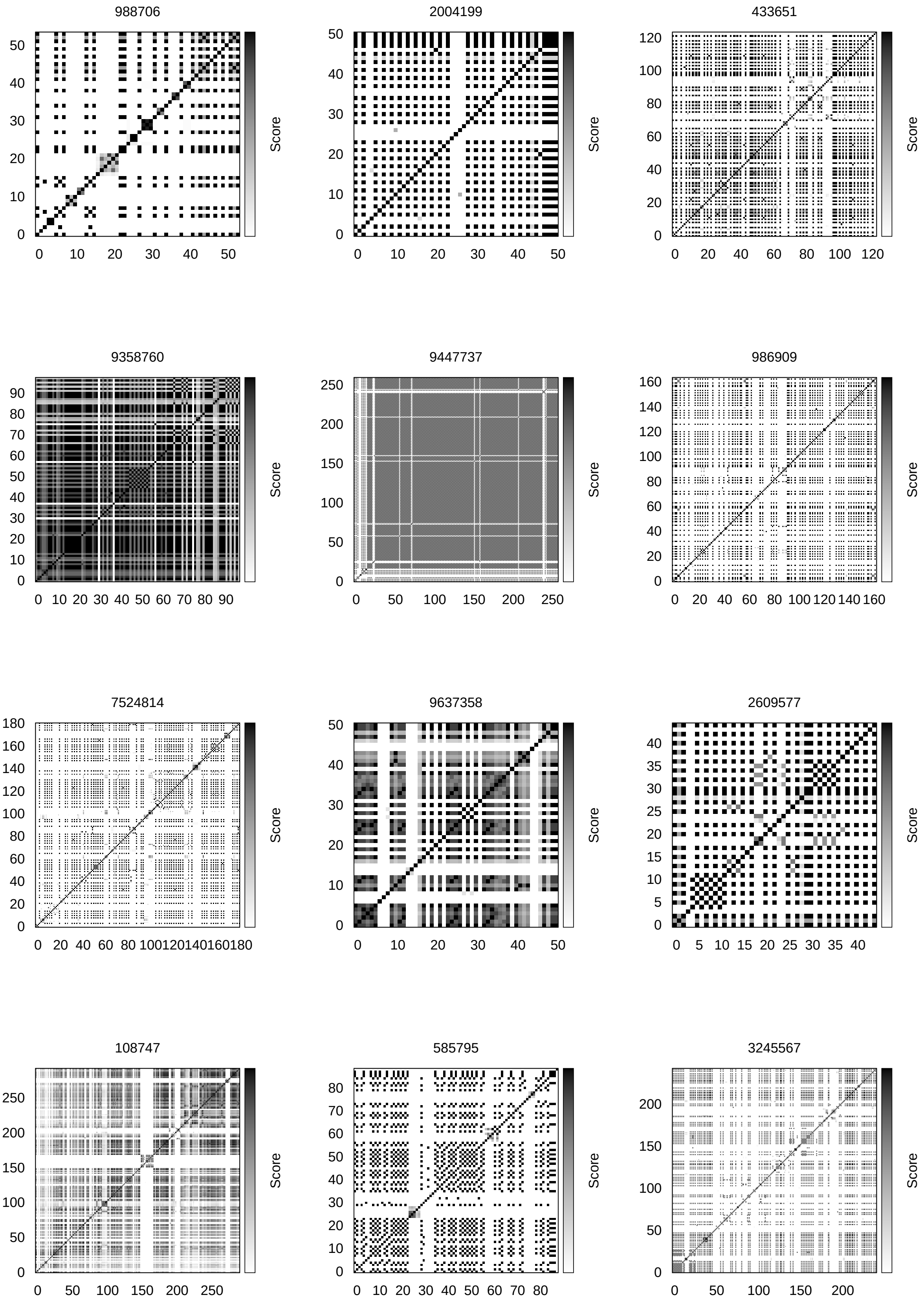

Figure 6.11: Heat-maps of robot users

users may not be precise. It can also indicate that the criteria that were used to classify the users focused on details and did not consider the user activity in a wider view that takes into account the whole behavior of the user.

We concluded that the work done regarding to classification the users is not final, and more precise work is needed to better distinguish the robot users from the human ones.

6.6 Quests Characteristics

Given the identification of quests, we can investigate the quests size distribution. This provide information about the length of the quests, and shows the number of change steps the user needed to perform in order to accomplish the search mission and receive the desired web page, or alternatively to quit the search.

6.6.1 Finding Quests Size

The quest size can be determined with the heat-maps. Using the n-gram method, we already have the similarity grade between each pair of user queries. We represented this data in the heat-map graphs. A typical heat-map contains a sequence of black squares along the diagonal, which is caused due to comparing the query to itself, and other squares with different shade of grey, including white and black, showing the resemblance between each pair of queries.

We look for large squares of adjacent related queries along the diagonal of heat-maps. The sizes of the quests are actually the lengths of these squares, as the squares shows the relationship between the queries.

As the heat-maps are symmetric it is enough to look at one of the up or down triangles. Choosing the down triangle, we start from the bottom left corner of the heat-map, the first black square on the diagonal, looking for a horizontal sequence of non-white squares. Of course this horizontal sequence of non-white squares can be of size one, in this case the query is not related to the next one, although it may be related to some later queries (we take care of them later). Reaching a white square means the end of the square sequence. At this time we record the size of the square sequence and its start position. Then we jump up to the next appropriate black square on the diagonal and repeat the above explained process again at the proper height of the new black square till we reach the last black square on the diagonal or to one of the squares in the final column of the heat-map. Figure 6.13 shows the pseudo code. The gradeHash used in the pseudo code is actually a hash of hashes that contains, for each user, the grades for each pair of his queries.

We observed that in some cases it happens that if we choose the upper triangle and thus start the process from the opposite direction with the last black square on the diagonal, we receive horizontal sequences of non-white squares with different lengths.

This may happen when in one direction we recognize a long sequence, while in the other direction we recognize a couple of shorter sequences that together have the same size. See example in Figure 6.14. The two small green squares are recognized using the algorithm that starts from the bottom left corner, and the big blue square is recognized using the algorithm that proceeds from top right corner.

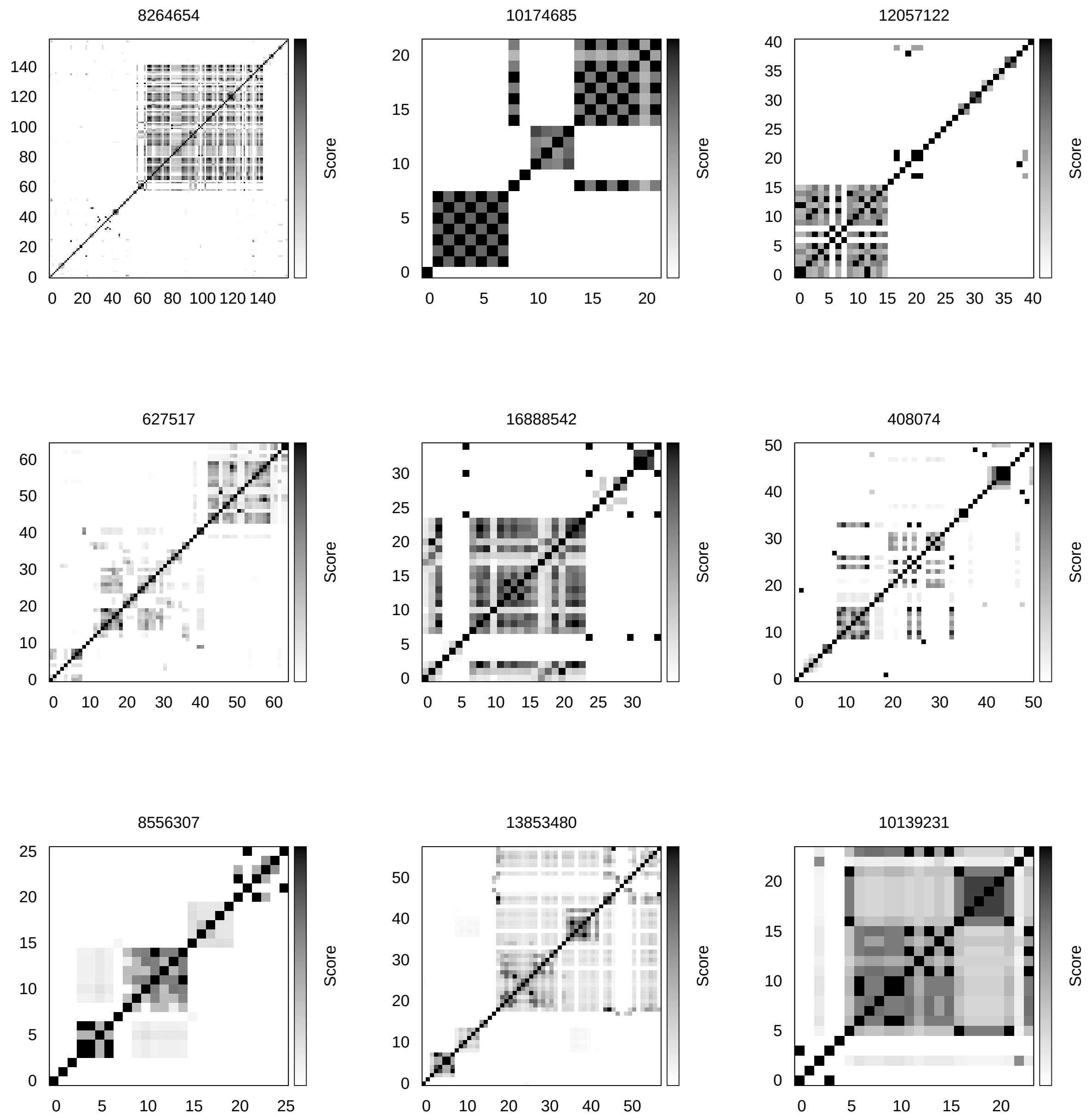

Figure 6.12: Suspicious heat-maps of robot users

```
Input: gradeHash contains the calculated grade for each pair of queries
k = 0;
size = 0;
//number of cleaned queries
arraySize = gradeHash{id};
i = 0;
while (i < arraySize)
    j = 0;
    while (i+j <= arraySize && gradeHash{id}[i][i+j] != 0)
        size++;
        j++;
    //Saves the size of square and its start position
    squaresLeft[k] = size;
    xLeft[k] = i;
    k++;
    i+= j;
    size= 0;
```

Figure 6.13: *Pseudo code of algorithm for finding squares from down left corner.*

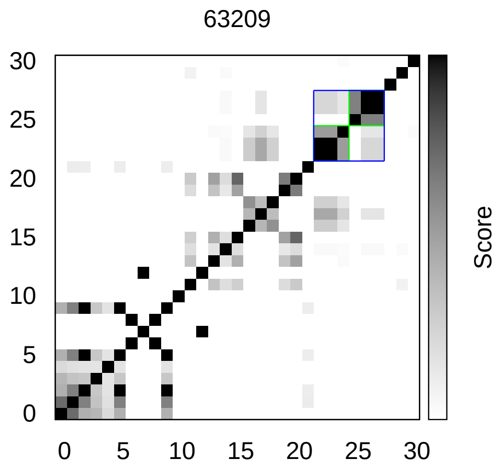

Figure 6.14: Example of different square recognition starting from opposite directions (small squares contained in a bigger one)

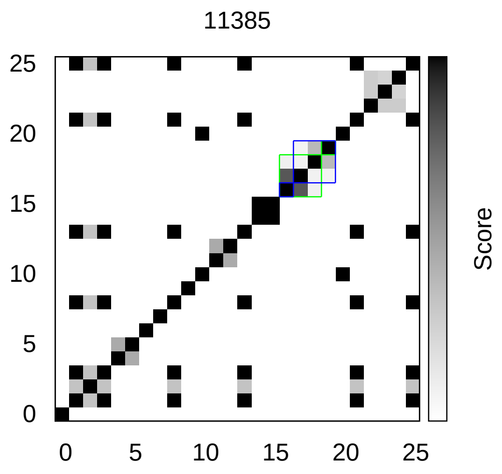

Figure 6.15: Example of different square recognition starting from opposite directions (squares have common cells)

Another case happens when in both directions we recognize sequences that have some shared squares. This leads to two series of sequences with different sizes while their total length is the same. See example in Figure 6.15. The two small green squares are recognized using the algorithm that starts from the bottom left corner, and the blue squares are recognized using the algorithm that proceeds from the top right corner. As we see both squares have common cells and finally define a bigger square.

To prevent these problems and to determine the length of sequences correctly, we choose to perform the finding sequence process in both directions. Figure 6.16 shows the pseudo code for the upper triangle. Since we save the length of sequences in both directions, we can compare these lists of sequence lengths and determine the actual size.

If the sequences in both directions agree on the same length, we accept the sequence length and move to the next sequences in both lists. Otherwise, there can be two options as explained before. In these cases we try to proceed in the list with shorter total length to achieve the length of the longer sequence on the other list. It may happen that while adding the next sequence length to achieve the longer length, we pass the desired length value, so we need to move to the other list and do the same process till we reach the same value in both lists. There could be more than one pass between lists. Figure 6.17 shows the pseudo code.

We run the above described algorithms on one arbitrary part of the AOL log. Figure 6.18 shows the histogram of quests length (the non-white square sequences length). We can see that most of the sequences are singletons with length one. There is a dramatic fall between singletons and quests with size two. The singleton quests appear almost five times more than the quests with size two. After that, as we proceed in the quests size the respective counts almost decreased in half. Figure 6.19 represents clearly that almost 76% of quests are singletons, about 14% of them are with size

```
Input: gradeHash contains the calculated grade for each pair of queries
k = 0;
size = 0;
//number of cleaned queries
arraySize = gradeHash{id};
i = arraySize - 1;
while(i >= 0)
    j = 0;
    while (i-j >= 0 && gradeHash{id}[i][i-j] != 0)
        size++;
        j++;
    //Saves the size of square and its start position
    squaresRight[k] = size;
    xRight[k] = i;
    k++;
    i-= j;
    size = 0;
```

Figure 6.16: *Pseudo code of algorithm for finding squares from top right corner.*

two and the rest which is just about 10% are longer than two. This shows that even longer quests are possible but they appear rarely, and mostly users leave the search process satisfied or not after few if any reformulations and/or repetitions.

6.6.2 Quests Completeness Relationship

Every square recognized with the above procedure represents a quest of related queries. These squares are symmetric by their diagonal so it is enough to concentrate on the upper or lower triangle. As usual we choose the down triangle. The non-white squares in the triangle shows that there is some relationship between the queries. It is interesting to know to what extent all the queries in this triangle are related to each other. White squares in the triangle show that although there is a path between these queries, there are big changes between them. So we decided to find the number of white squares in each triangle and checked which portion of the triangle area is covered with white squares. Using the agreed sequence length list we can easily find this sparseness measurement, as we also saved the start position of sequences. Figure 6.20 shows the pseudo code.

We have found that about 97% of triangle quests had no white squares. This rate contains also the triangles with size one or two that by definition can not contain white squares. It matches our previous findings about the sequence lengths (the triangle base). It also means that about 75% of triangles with base greater than two are fully covered with grey or black squares and do not contain

```
//number of squares found starting from left down corner
sizeLeft = squaresLeft.length;
//number of squares found starting from top right corner
sizeRight = squaresRight.length;
k = 0;
leftX = 1;
xPosition = -1;
a = 0;
b = sizeRight-1;
left = 1;
right = 1;
sumLeft = 0;
sumRight = 0;
while (a < sizeLeft && b >= 0)
    if (left)
        sumLeft += squaresLeft[a];
        if (leftX)
            xPosition = xLeft[a];
            leftX = 0;
        a++;
    if (right)
        sumRight += squaresRight[b];
        b- -;
    if (sumLeft < sumRight)
        while ((sumLeft < sumRight) && (a <= sizeLeft))
            sumLeft += squaresLeft[a];
            a++;
        if (a == sizeLeft+1)
            a = sizeLeft;
        left = 0;
        right = 1;
    if (sumLeft > sumRight)
        while ((sumLeft > sumRight) && (b >= 0))
            sumRight+= squaresRight[b];
            b- -;
        if (b == -1)
            b = 0;
        left = 1;
        right = 0;
    if (sumLeft == sumRight)
        squaresHash{sumLeft}++;
        squaresAgreed{k} = sumLeft;
        xAgreed{k} = xPosition;
        k++;
        leftX = 1;
        left = 1;
        right = 1;
        sumLeft = 0;
        sumRight = 0;
```

Figure 6.17: *Pseudo code of algorithm for finding the agreed size of squares.*

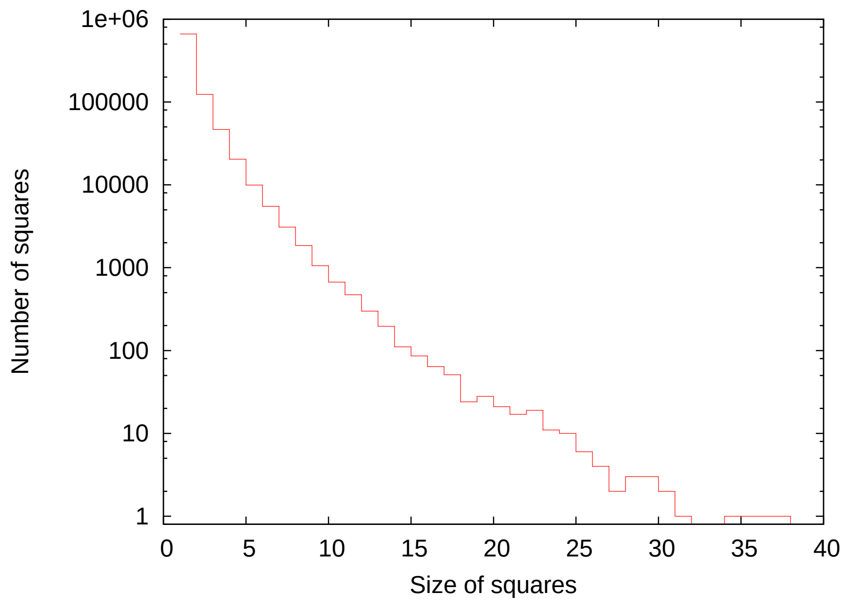

Figure 6.18: Histogram of square quests length

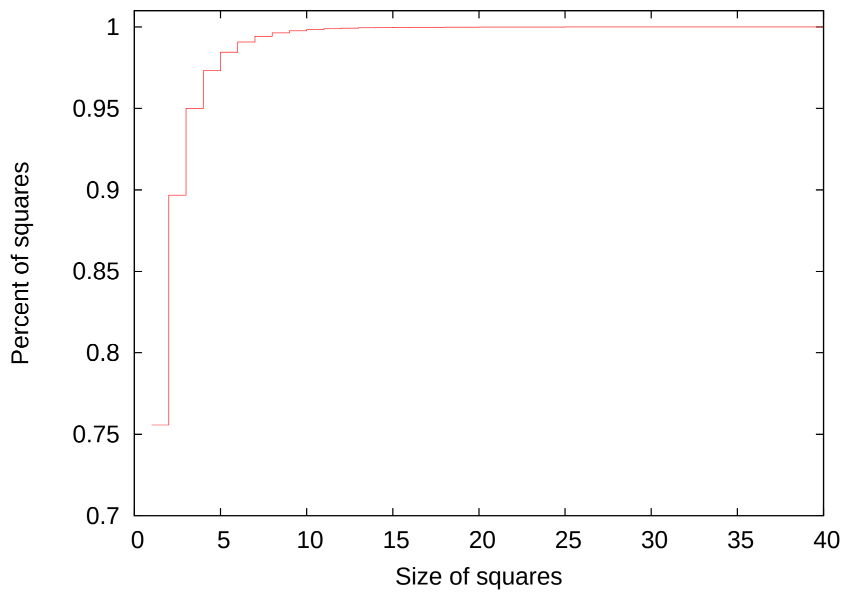

Figure 6.19: Percentage of square quests length

```
zero = 0;
//Checks if the square size is bigger than two
if(size > 2)
    n = 0;
    while (n < size)
        m = 0;
        while (n+m < size)
            if (gradeHash{id}[xPosition+n][xPosition+n+m] == 0)
                zero++;
            m++;
        n++;
```

Figure 6.20: *Pseudo code of algorithm for finding the white cells in the triangles.*

any white square.

As the white squares can only appear in triangles with bases longer or equal to three, if we remove these tiny triangles we receive that about 75% of triangles have no white squares. Figure 6.21 shows the histogram and Figure 6.22 shows the proper percentage. As we see in these figures, the portion of empty cells is reduced when the triangle size gets bigger, and for the high values it mostly stands on one occurrence. Checking the related data, we have found that the maximum value for empty cells was 428, which is bigger by 251 than the previous value. Figure 6.23 shows its heat-map. The big red emphasized square is the one that contained this amount of zero cells. Actually this big square is created due to having both cases of not equally diagnosed small squares, one is contained in the other and they have common cells, that are connected together by our algorithm. The green squares are recognized using the algorithm that proceed from left down corner while the blue ones are recognized using the algorithm that starts from the opposite side, right top corner.

While the two previous Figures 6.21, 6.22 showed absolute number of the empty cells in the triangles, Figures 6.24, 6.25 show that the portion of the empty cells out of the whole size of the triangles. As mentioned before about 75% of triangles are totally non-white. We can see that about in 99% of the triangles, half or less of their squares are empty cells, while just about 24% of triangles have some non-white portion. The maximal rate of empty cells stands on about 67% of its field.

6.6.3 Relations between Quests

We have already talked about the non-white squares sequences, the triangles they create, and the white squares they contain. Another phenomenon viewed in the heat-maps are the gray shadow squares or even black ones which are not part of these triangles but appear in the heat-maps. This

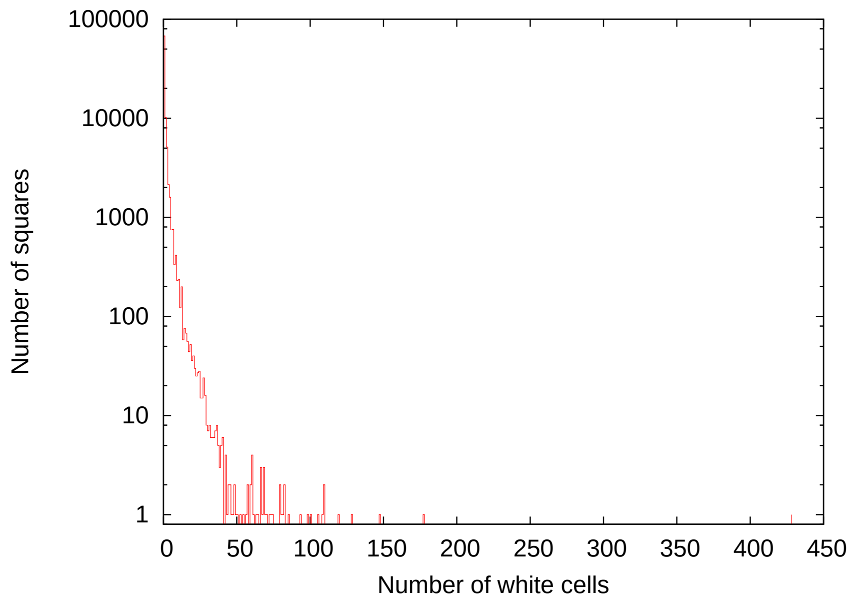

Figure 6.21: Histogram of number of zero cells in triangles

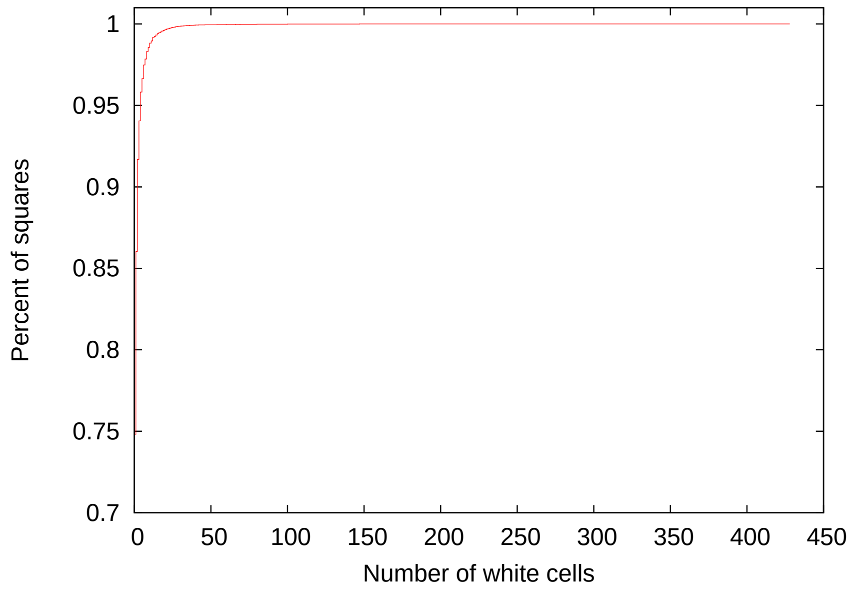

Figure 6.22: Percentage of number of zero cells in triangles

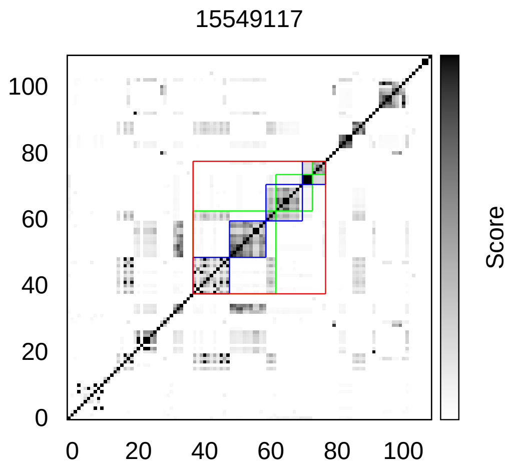

Figure 6.23: The heat-map with highest number of empty cells in a recognized square

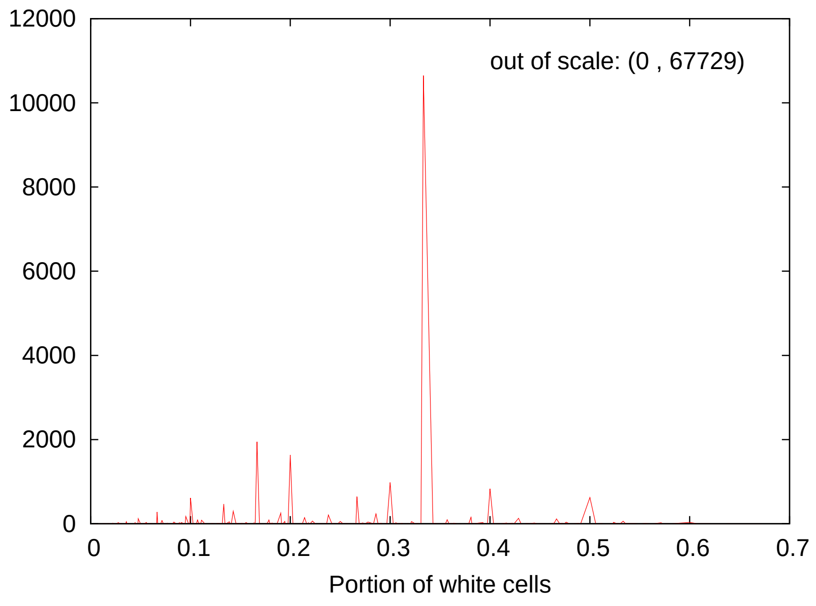

Figure 6.24: Histogram of portion of zero cells in triangles

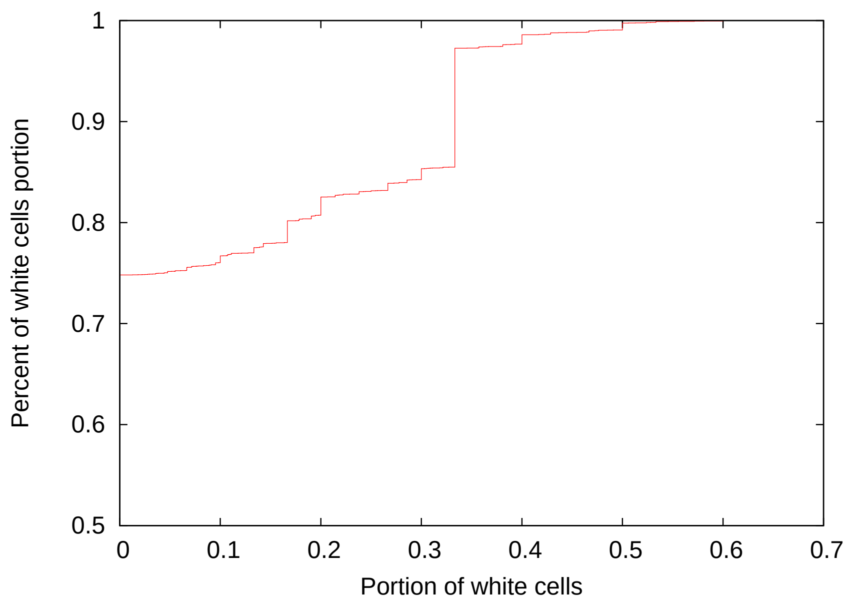

Figure 6.25: Percentage of portion of zero cells in triangles

shows a non-continuous relationship between one query and later ones. These separated non-white squares represent a special kind of quest which exhibits repetition or reformulation of queries again and again during the user activity. These repetitions occur after longer times than usual, mostly in subsequent days or weeks. We thought to quantify this phenomenon.

For each quest along the diagonal, we check if it is related to any other quest. If the quest is of size one, it is actually one singleton square on the diagonal. We already have the square sizes (the sequences), which were found by traversing in both directions on the heat-map graph, and the start position of each of them. We also have the similarity grades for all the queries, so we can easily check if there is any non-white squares in the rows of each of these quests. We may remember that each square in the heat-map belongs to one quest. If we found such a non-white square, we remove the related quest and continue till the last quest. We repeat this process for all the quests. Figure 6.26 shows the related pseudo code.

We decided to count the total size of these separated non-white square sequences. The minimal size is two that points to the case when two singleton quests are related. Figure 6.27 shows their size histogram. It does not show the occurrence of square sequences that are not related. Using this method of counting we reach higher values for the quests size, that shows the continuousness in the quest through longer times, where this relationship can spread over days, weeks and even months. Figure 6.28 shows the percentage of related quests size. We can see that about 32% of the related quests are of size two, in other words two singleton quests that are related. As the size of the related quests increases their portion is decreased, but in total we see a strong rise of quests by size ten and after that it arises slowly till it reach 99% at size twenty five. We may see that although longer sizes are available, they are not common at all. The longest size reported is 222 and there is a jump of 32 from the previous value.

Another interesting parameter to evaluate is the number of parts these related quests are assem-

```
for each square k from squaresAgreed
    relatedParts = 0;
    rSquareSize = 0;
    size = squaresAgreed{k};
    xPosition = xAgreed{k};
    related = 0;
    for each squares p from squaresAgreed
        newSq:
        if (k != p)
            size_2 = squaresAgreed{p};
            xPosition_2 = xAgreed{p};
            i = xPosition_2;
            while (i < xPosition_2 + size_2)
                j = xPosition;
                while (j < xPosition + size)
                    if(gradeHash{id}[i][j] != 0)
                        relatedParts++;
                        rSquareSize+= size_2;
                        delete squaresAgreed{p};
                        related = 1;
                        goto newSq;
                j++;
            i++;
    delete squaresAgreed{k};
    if(related != 0)
        relatedParts++;
        rSquareSize += size;
        partsHash{relatedParts}++;
        rSquareSizeHash{rSquareSize}++;
```

Figure 6.26: *Pseudo code of algorithm for related quests in heat-map.*

bled from. Figure 6.29 shows the popularity of each number of parts. It is obvious that the minimal number of parts is two. As we see, it is also the most popular value, while the next value, related quests with three parts, has almost a third of the popularity of this. The popularity of the related quest is reduced as the number of parts they are combined from is increased. Figure 6.30 shows the portion of each multi part quest. As we said before, two-parted quests are most popular and constitute about 58% of all combined quests. With quests combined of thirteen parts, we achieve almost 99% of all combined quests. We see a fast rise till we reach the quests combined from thirteen parts and after that we have a more slow and smooth rise. Large combined quests are also found but their portion is very tiny.

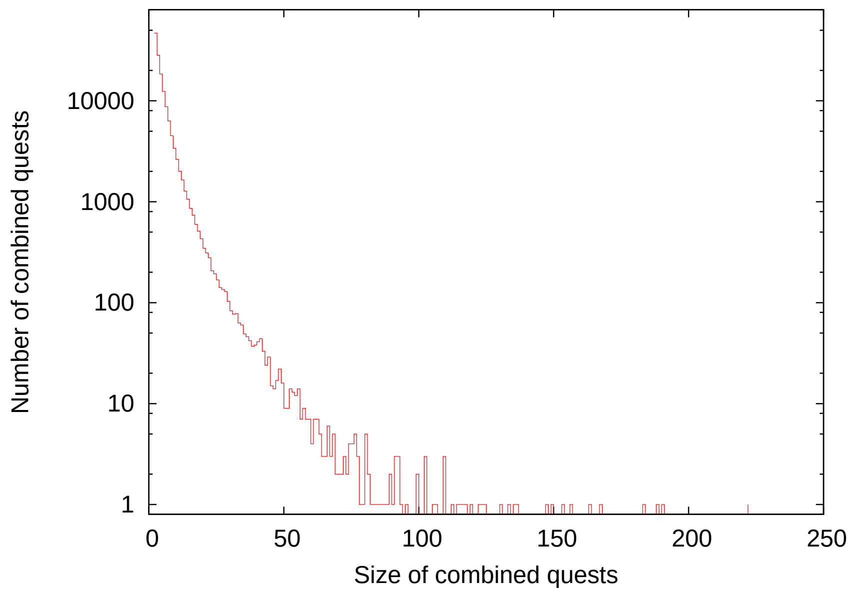

Figure 6.27: Histogram of combined quest sizes that are composed of multiple parts

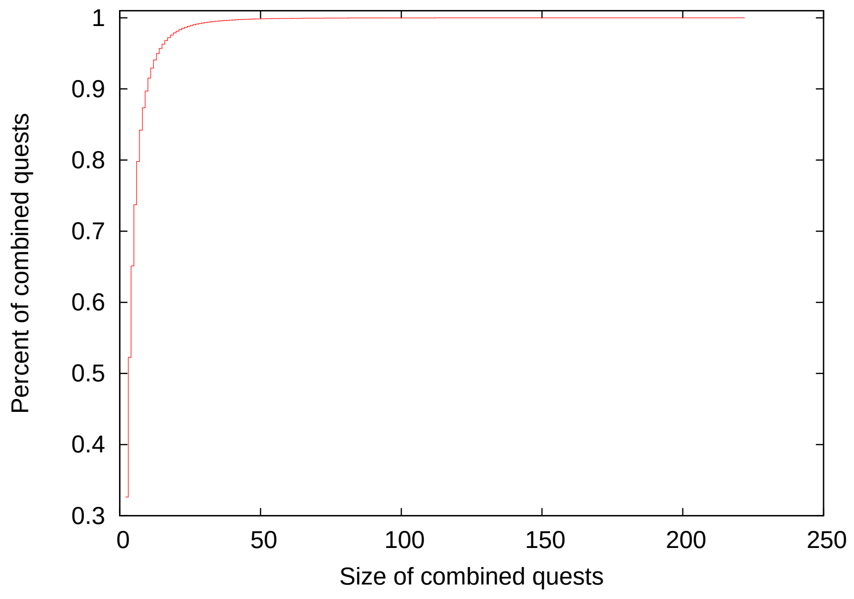

Figure 6.28: Percentage of combined quest sizes that are composed of multiple parts

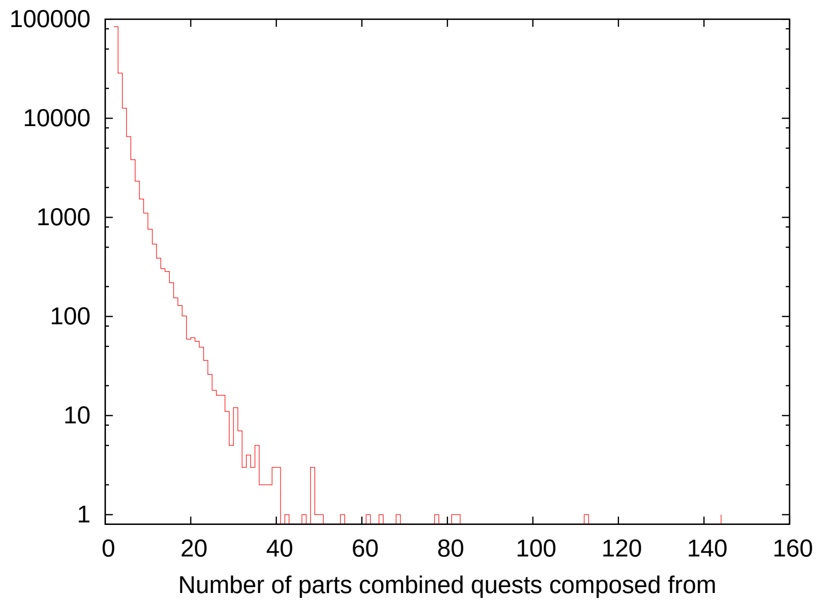

Figure 6.29: Histogram of combined quests that are composed of multiple parts

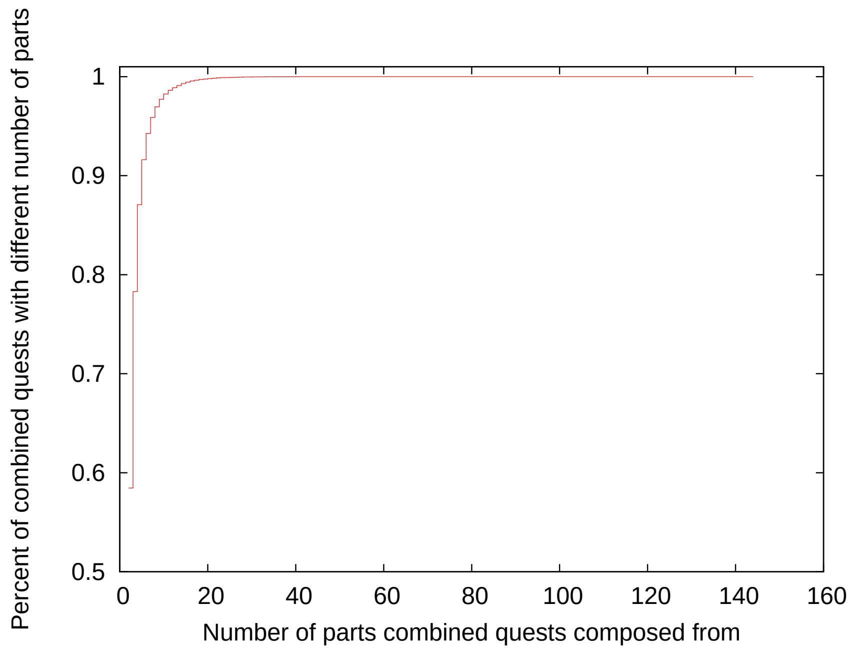

Figure 6.30: Percentage of combined quests that are composed of multiple parts

Chapter 7

Sessions and Quests

7.1 Parallel Usage of Sessions and Quests

We have seen till now two different methods for dividing and grouping the sequence of queries of users to sessions, using either the time intervals or lexical comparisons between queries to accomplish the mission. Now we consider exploiting these methods simultaneously. Having the individual session threshold using our algorithm and the heat-maps created using the n-gram method, we can easily represent these informations together.

We created graphs that show the heat-maps of the queries as before and also put line borders between the queries according to the individual threshold which is appropriate for this user. Actually we expect to see that our bounds cut the graph at places that are also appropriate visually to the shapes on the heat-maps and thus to see no collisions. But we have seen that there are quests that are cut by the session thresholds to two parts, see Figure 7.1, and there are also quests that are continued over more than two sessions, see Figure 7.2. Another phenomenon is that session thresholds do not surround each quest and a session may contain more than one quest.

We would like to measure how much these two different methods agree on the session detecting

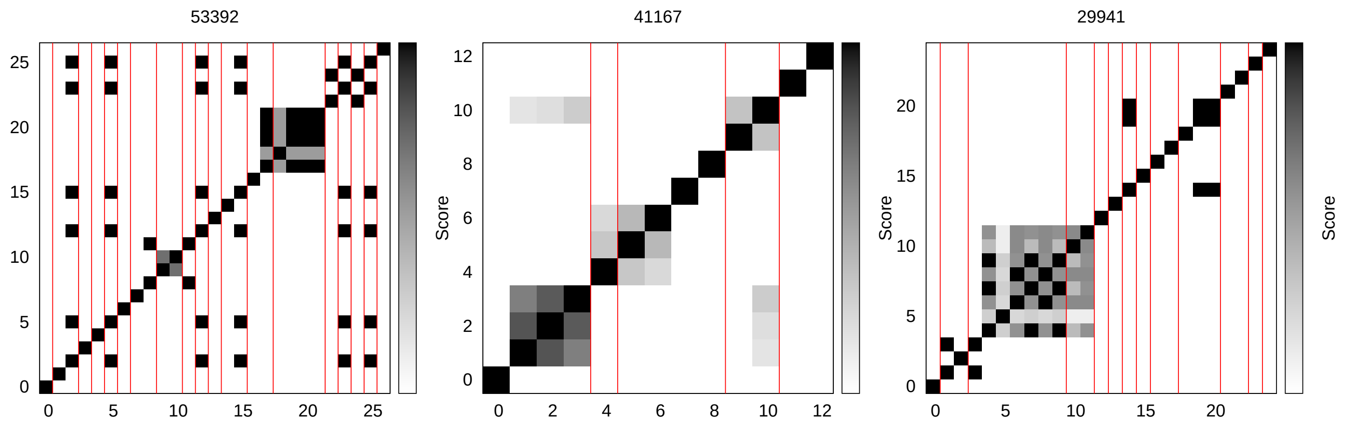

Figure 7.1: Quests that are cut by the session thresholds to two parts

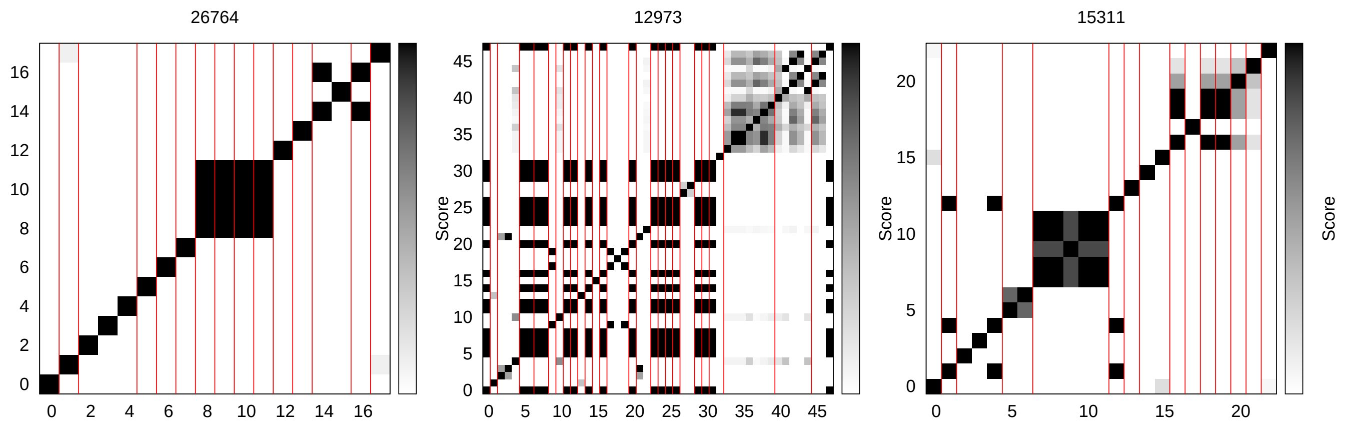

Figure 7.2: Quests that are continued over several sessions

issue. So we need to measure some parameters that show their agreement rate. This was done by finding the prevalence of each kind of collisions. First we found the number of times that the quests were cut by the session thresholds at any place during their length. This would show if the collisions are really frequent.

We found that the total number of times that quests were cut was 63116. This includes cases where the same quest was cut multiple times. As session cuts can appear only in quests with size bigger than one, we received that about 29% of quests longer than one were cut. We may remember that just 24% of quests are longer than one. This shows that a small portion of quests are cut by session thresholds, about 7% of all quests.

As we mentioned before, there are quests that are spread over more than one session. In this case the user continues his search for the same or similar topics over multiple sessions. But mostly the quests are bounded between the session thresholds. As we can see in Figure 7.3 about 94% of quests are contained in one session, roughly 5% of quests are continued over two sessions and the rest are the longer quests that contain more than two sessions during their activity. Figure 7.4 shows that the highest value recorded is one quest that spreads over twenty sessions.

Another interesting issue regarding the simultaneous usage of these two session detecting methods is when the sessions are comprised of more than one quest. This can show that topic changes in a search may not definitely point to the end of a session, and it is possible that users change their search topic several times during a session. We checked how many quests are included in a session, or in other words the number of times the user changed his topic he searched for during a session. As we look for the topic changes, in a case that a quest is divided between several sessions, it is counted in each of them separately.

We found that about 79% of sessions were composed of one quest and there were no topic changes. Figure 7.5 shows also that about 13% of sessions contained two quests. As we see the chance of having more than three quests in one session is small and is less than 4%. Figure 7.6 shows the occurrence of sessions that are composed of different numbers of quests. The maximal value of quests contained in a session is twenty eight which occurred once. We see that in most cases both detecting methods agree on the borders. Figure 7.7 shows that our individual thresh-

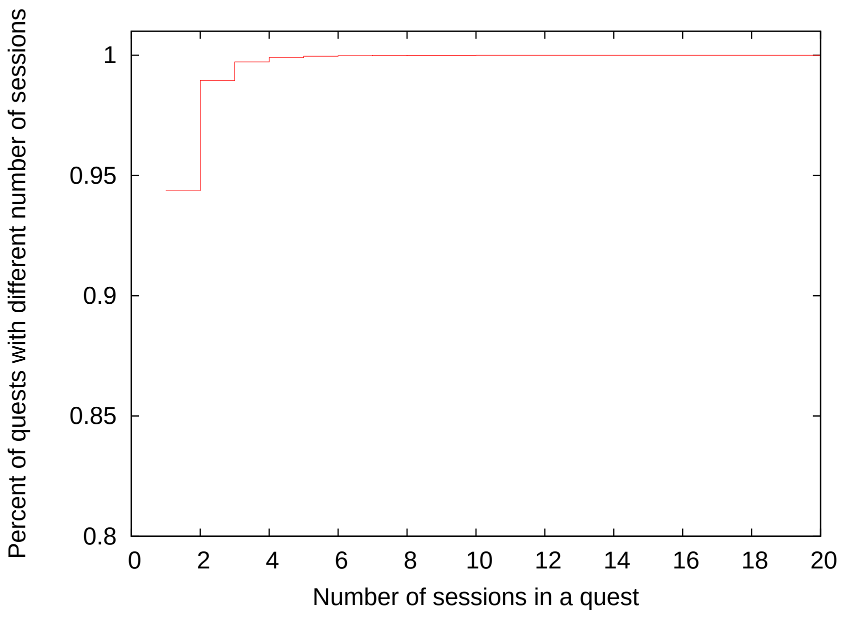

Figure 7.3: Percentage of quests that spread over different number of sessions

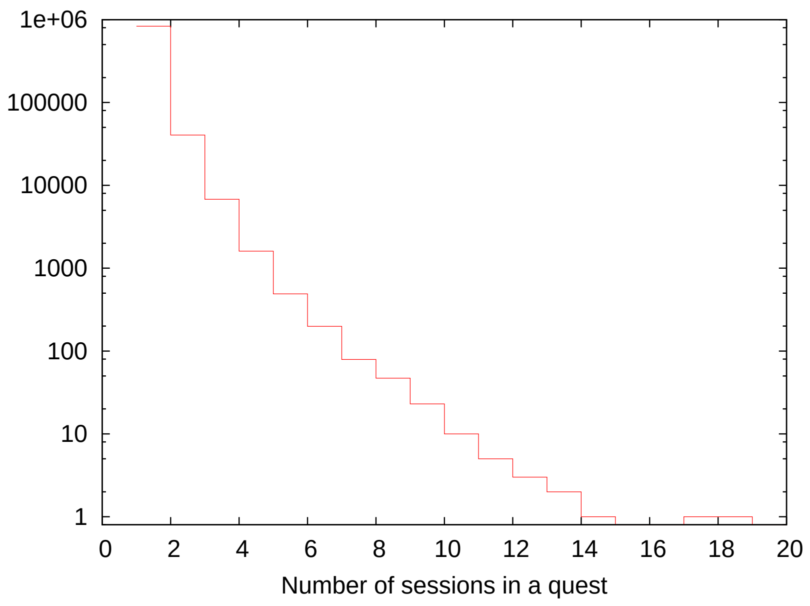

Figure 7.4: Histogram of quests that spread over different number of sessions

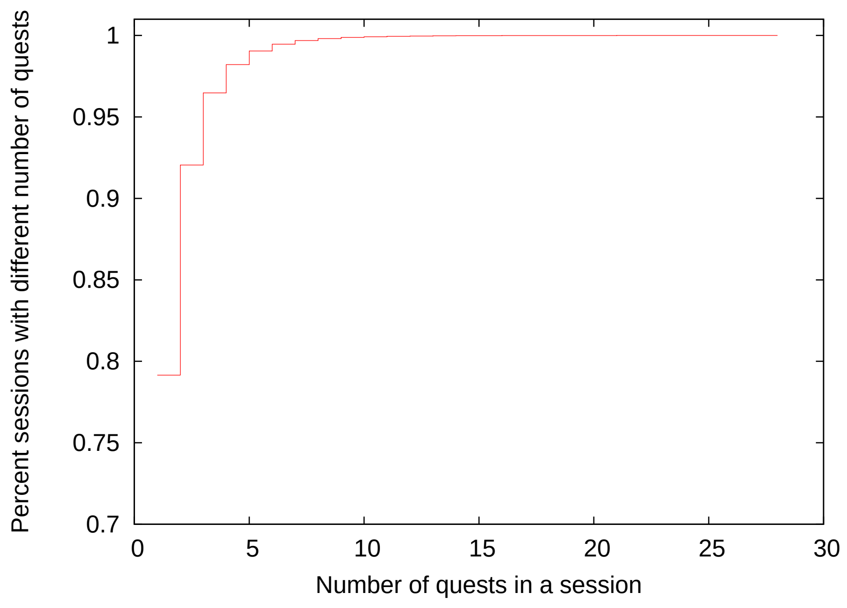

Figure 7.5: Percentage of sessions with different number of quests

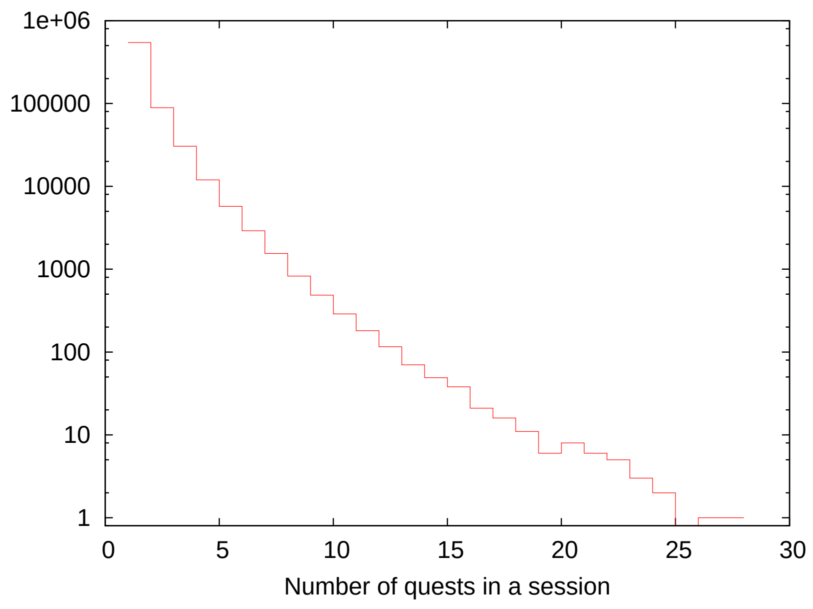

Figure 7.6: Histogram of sessions with different number of quests

Date	Time	Query	Cleaned New Query
2006-03-05	00:19:03	last kiss lyrics	kiss lyrics
2006-03-05	19:38:34	today lyrics	today lyrics
2006-03-09	16:46:03	we might as well lyrics	lyrics
2006-03-09	17:33:46	calling you lyrics	calling lyrics
2006-03-09	17:33:46	calling you lyrics	calling lyrics
2006-03-09	17:44:07	last call lyrics	call lyrics
2006-03-09	18:23:56	over my head lyrics	head lyrics
2006-03-09	18:47:03	with or without you lyrics	lyrics
2006-03-09	18:58:52	ceom as you are lyrics	ceom lyrics
2006-03-09	18:59:49	come as you are lyrics	come lyrics

Table 7.1: Some part of User Id 7783914 that cause the long quest

old lines cuts the graph properly so that in case the heat-map shows a quest (query modification process) the threshold lines bounds it form both sides.

We have checked the users that had long quests that were continued over more than ten sessions. We found that most of these quests are recognized because the user uses some term that appears in a sequence of his queries, while the user does not actually reformulate his query by adding or removing terms. Rather he just changes the whole query except the term that accompanies the queries. This dose not actually match the quest definition, but rather points to some field of interest of the user. Figure 7.8 shows the heat-map of user id 7783914 and table 7.1 shows some part of the respective log. We see that the user uses the term ”lyrics” during his search but he change the whole query except this term each time. Thus he is generally interested in song lyrics, but each song should be considered a separate quest.

This exemplifies the benefit of using a long log that enables us to observe these changes in long periods and distinguish them from quests. In short logs there is no possibility to see such a phenomenon as we do not have the user activity for a long time. It is possible that the phenomenon is referred as a quest in short logs, since there is no way to figure out otherwise, as the short length activity dose not allow it.

There were also users that had quests that contained a high number of sessions, but this was created as the user repeated the same query in long time intervals, which caused it to be in different sessions, but with misspelling so that we did not remove them since they were assumed to be different queries.

7.2 Exploiting Session Thresholds in Detecting Quests

We thought that there might also be a non-negligible portion of users that always search for one or two permanent queries and repeat them again and again. Since we just consider the different queries such repetitions are removed and do not appear at all in our quest analysis. Thus we decided to utilize the time dimension while looking for different queries. This means that we consider the

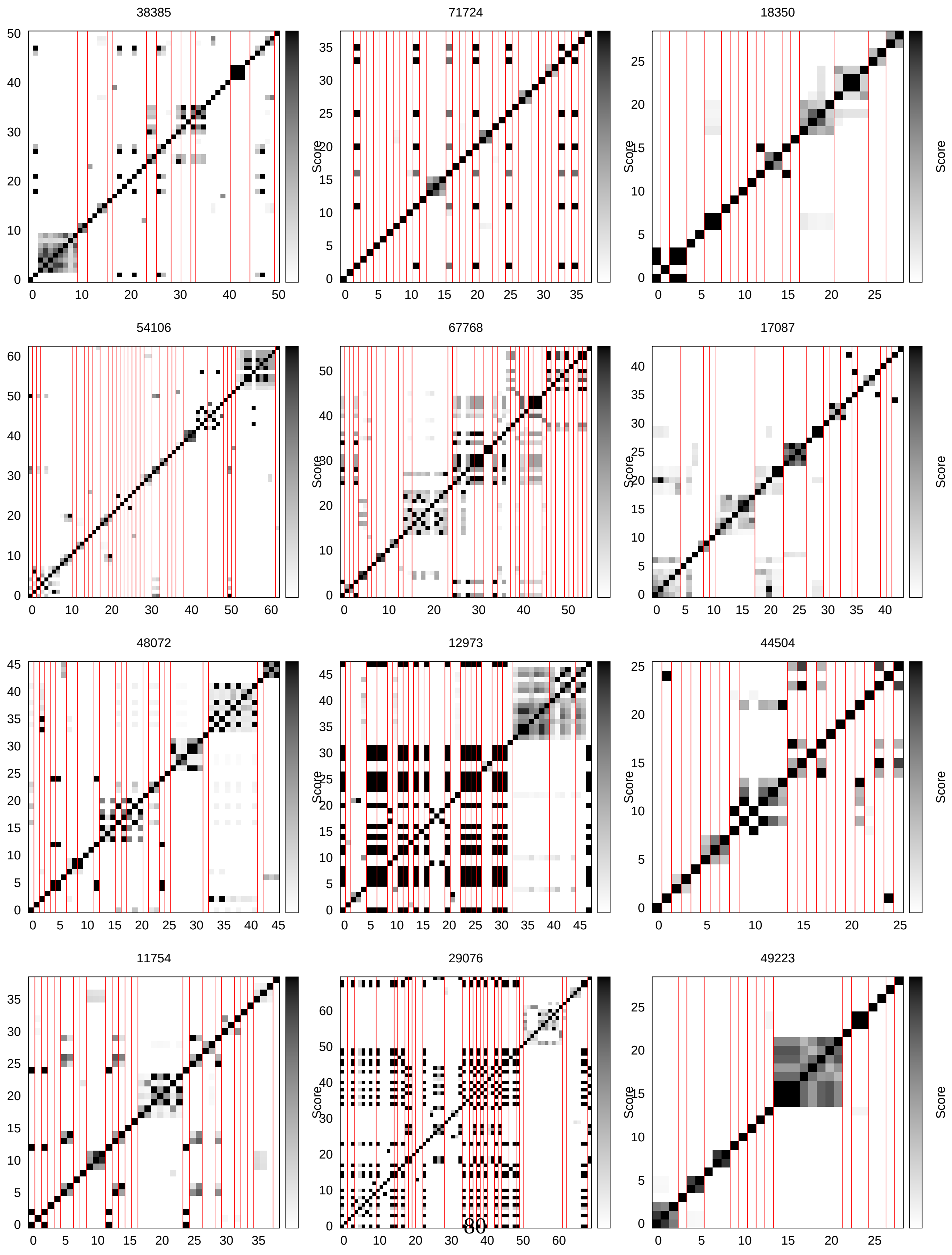

Figure 7.7: Parallel usage of session bounds and heat-maps

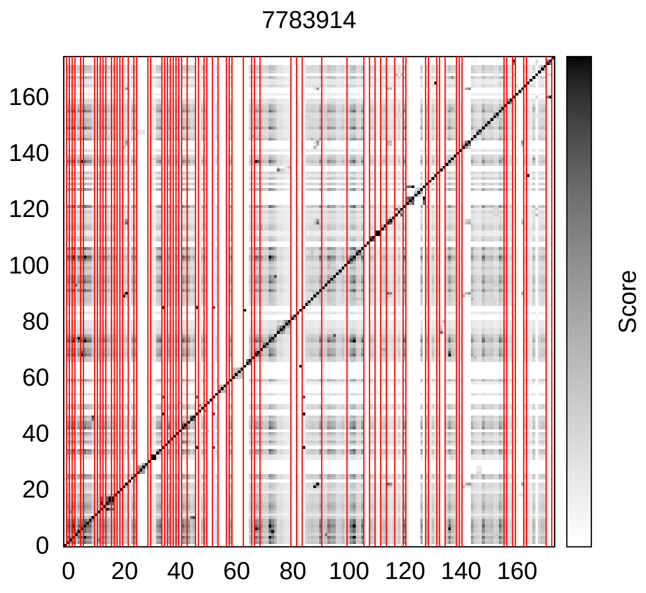

Figure 7.8

time interval that passed between the equal queries, and based on it decide to retain or remove them. As we already have the personal session threshold for each user, we decided to remove the equal queries in case that the time interval between them is shorter than the session threshold, and to keep them if the time interval passes the session threshold and announce a new session.

Using this approach we repeated our previous analysis. We checked the size distribution of quests. It is obvious that now there would be quests that are larger, since the equal queries that were not removed, would not stand alone and will be recognized as part of quests that were recognized before since they have the highest similarity grades with their previous query. Thus we expect to see that the size distribution moves to bigger values and there would be less singleton quests (quests with length one). Figure 7.9 shows the new distribution of quests size. We see in Figure 7.10 that actually only 3% of singleton quests are decreased and there is small increase in popularity of loger quests.

We think that adding these equal queries would not affect on the number of zero graded cells in the quest triangles. Thus their portion in the quests is also supposed to stay as before. As shown in Figures 7.11, 7.12 we see minor rise in the popularity of quests with no zero cell. This may be explained by the new equal queries that were added to the quests with sizes one or two and caused them to have larger sizes (meaning equal or bigger than three) and thus counted as squares with no zero cells. It might be worthwhile to remember that squares with length less than three by definition can not have zero cells and thus are not shown in the figures. We also see the same behavior in Figures 7.13, 7.14 which show the percentage of zero cells out of the whole size of the quest triangle.

Another aspect we checked was the matching of the two session detecting methods. There were three measurements that we used before and thus repeat them again. The first one was the number of cuts of session thresholds in quests. We expect to have more cuts of quests by the

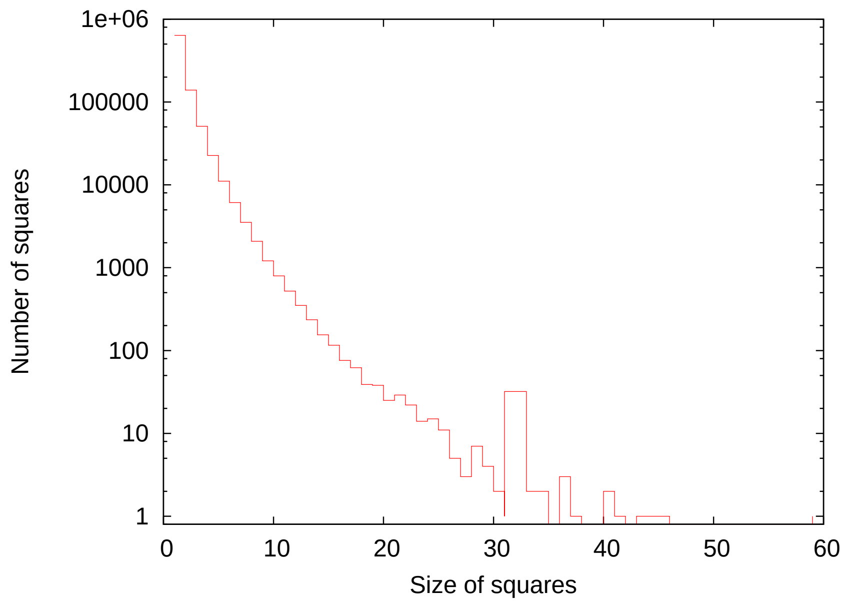

Figure 7.9: Histogram of square quests length

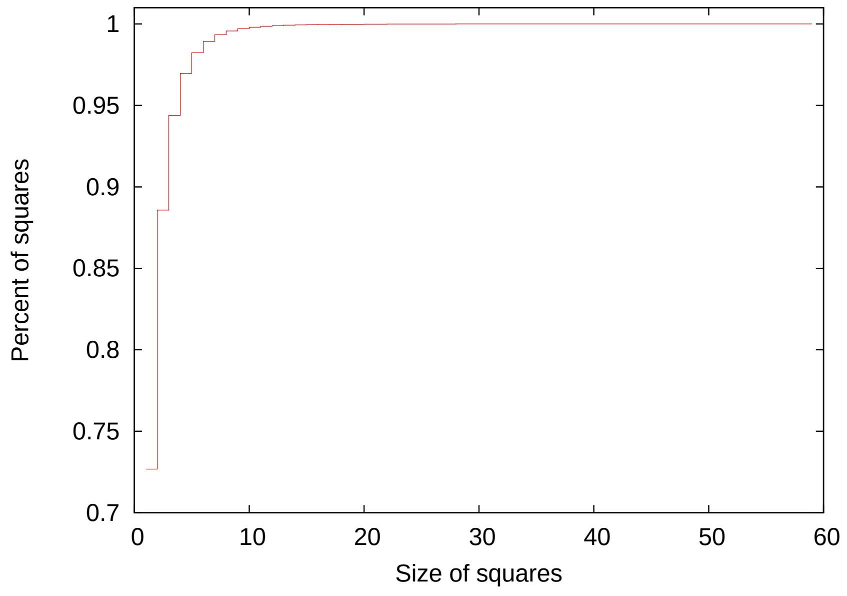

Figure 7.10: Percentage of square quests length

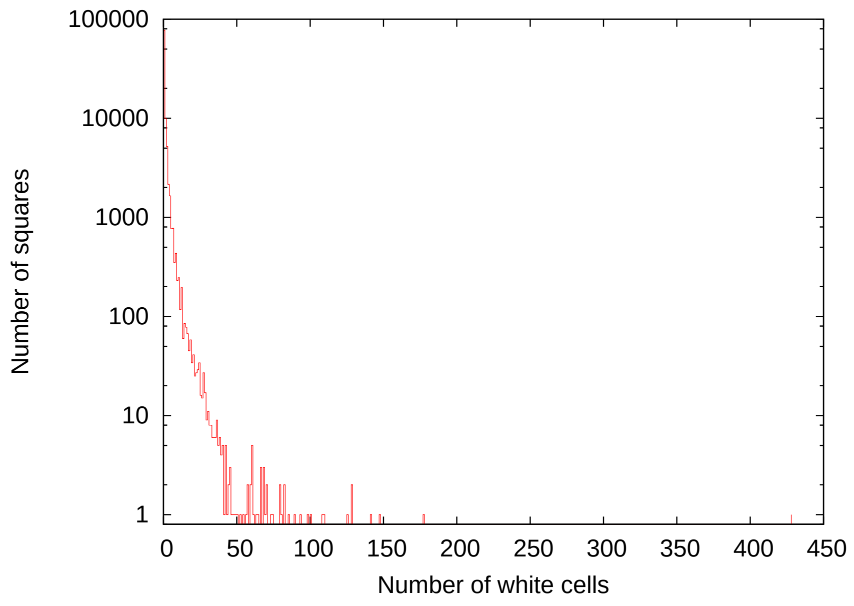

Figure 7.11: Histogram of number of zero cells in triangles

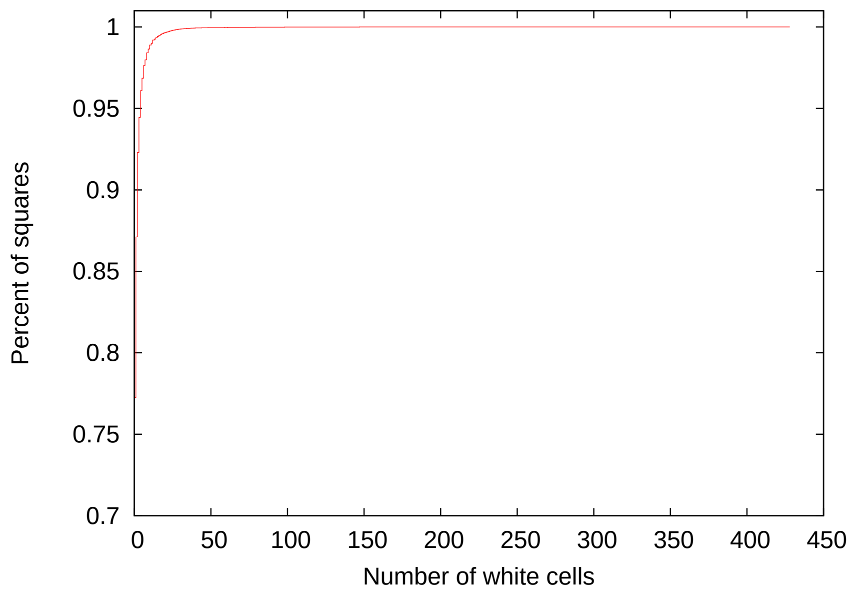

Figure 7.12: Percentage of number of zero cells in triangles

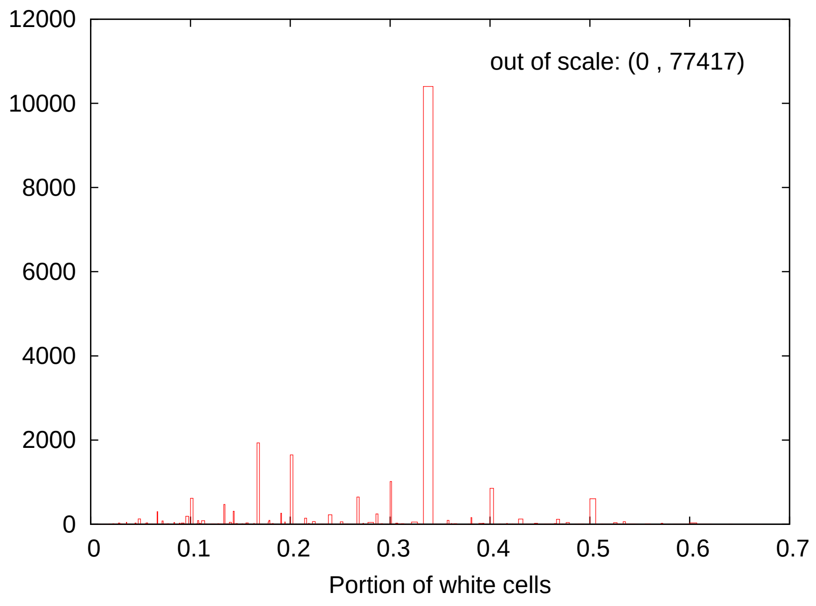

Figure 7.13: Histogram of portion of zero cells in triangles

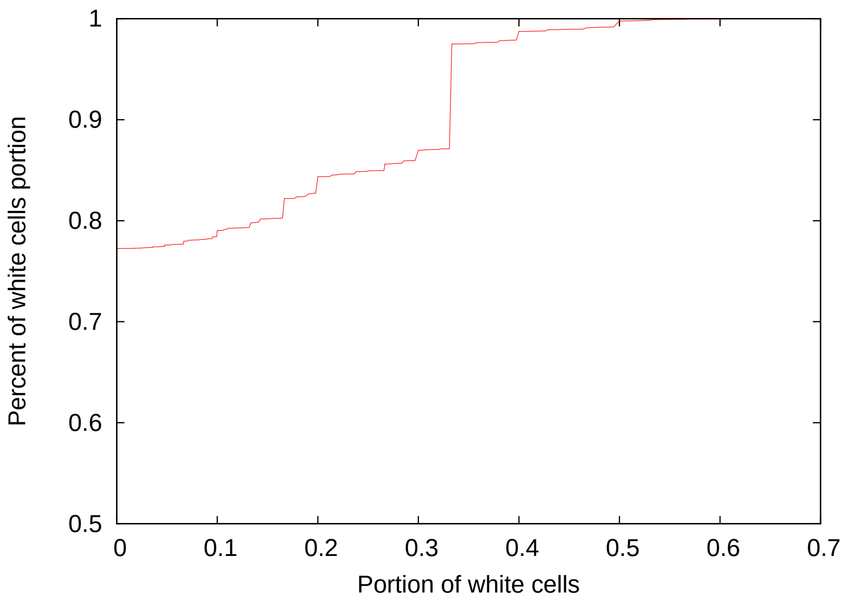

Figure 7.14: Percentage of portion of zero cells in triangles

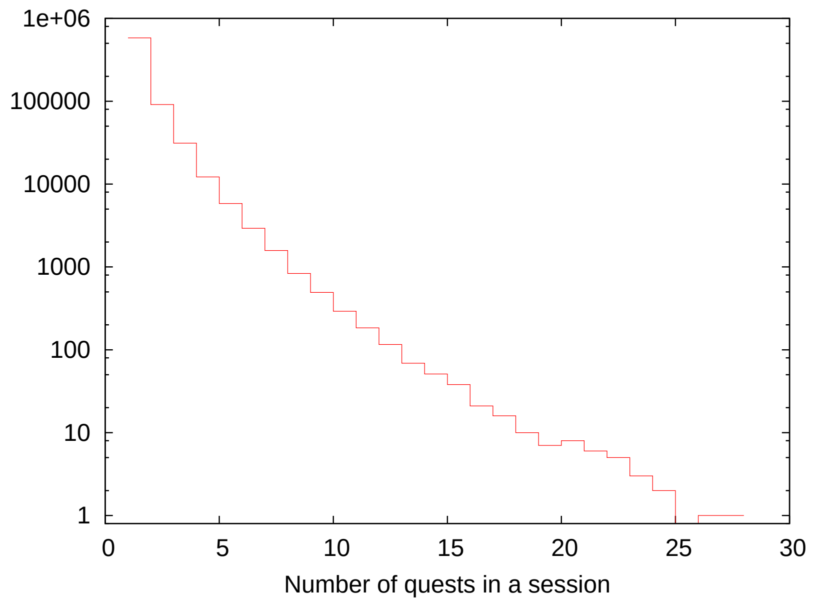

Figure 7.15: Histogram of sessions with different number of quests

session thresholds since the equal queries we kept are those that their time intervals are longer than the session thresholds and as explained before they would be added to existing quests thus causing more cuts. Each increment of the quest size is accompanied with another cut. The new value of cuts is 109796. We see that it increased by 46680, which is roughly 74% growth in the number of cuts.

The second parameter we checked is the number of quests in each session. It is supposed to have a similar distribution with small raise in the popularity of sessions that contain singleton quests, because we added the equal queries. These equal queries are connected to already existing quests, but they appear at new session borders, and as we look for the number of quests in a session, the interrupted quests would be counted in each session separately and thus we see an increase in the number of quests in sessions. Figure 7.15 shows the related histogram. We can see in Figure 7.16 that the percentage of each kind of sessions are almost the same as before.

The last interesting measurement that may change is the number of quests that are spread on more than one session. As we explained before the equal queries will be connected to the existing quests while on the other hand they belong to another session since they are added in case that the time interval passed the session threshold. This addition in quest length will cause the quest to be continued in another session. Thus as represented in Figure 7.17, we see less quests that are limited to one session and a significant increase in the population of longer quests is observed. We also recognized quests with longer lengths that were not recognized before although most of them occur only once. Figure 7.18 shows the percentage of quests that are continued over different number of sessions.

This shows that there are not too many users whose activity in the search engine is basically a couple of permanent queries that are repeated all the time. We estimate that about 3% to 5% of quests represent such usage of search engines. We see that this addition to our analysis does not change too much our results and it may also be ignored.

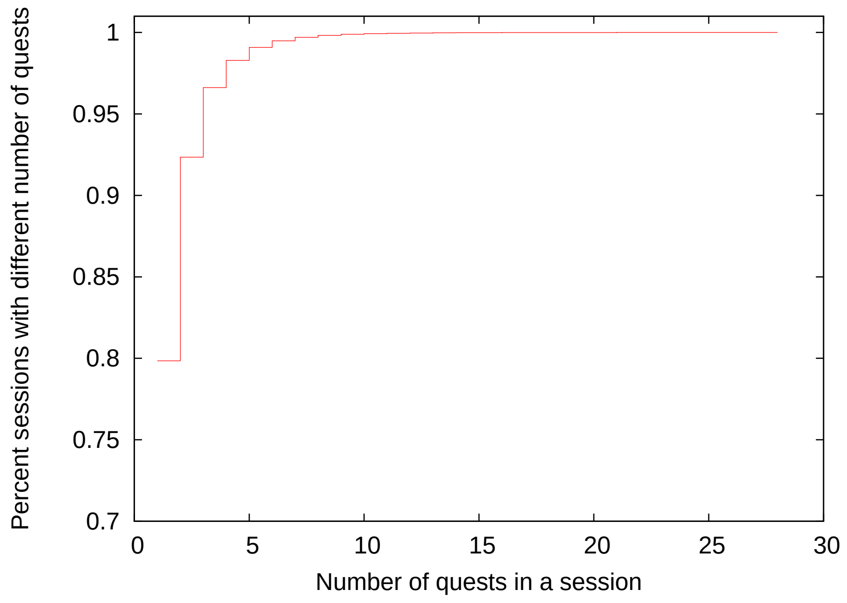

Figure 7.16: Percentage of sessions with different number of quests

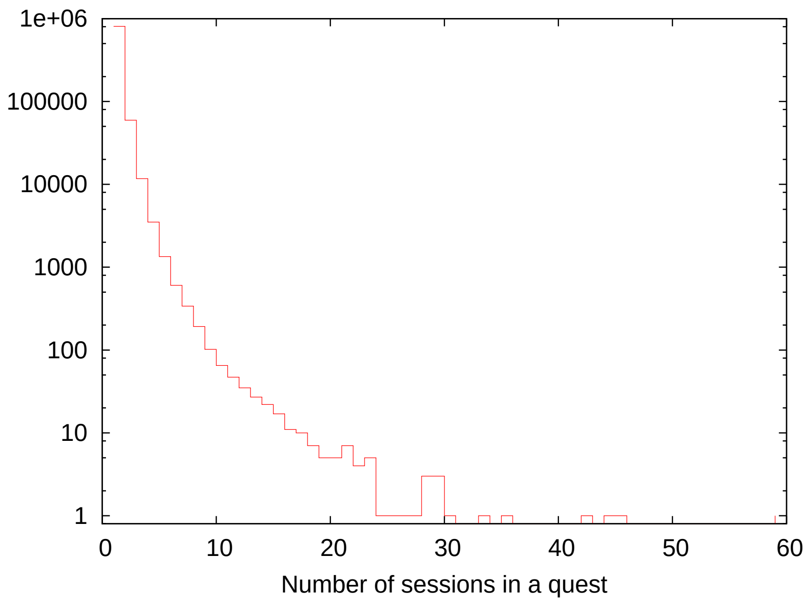

Figure 7.17: Histogram of quests that spread over different number of sessions

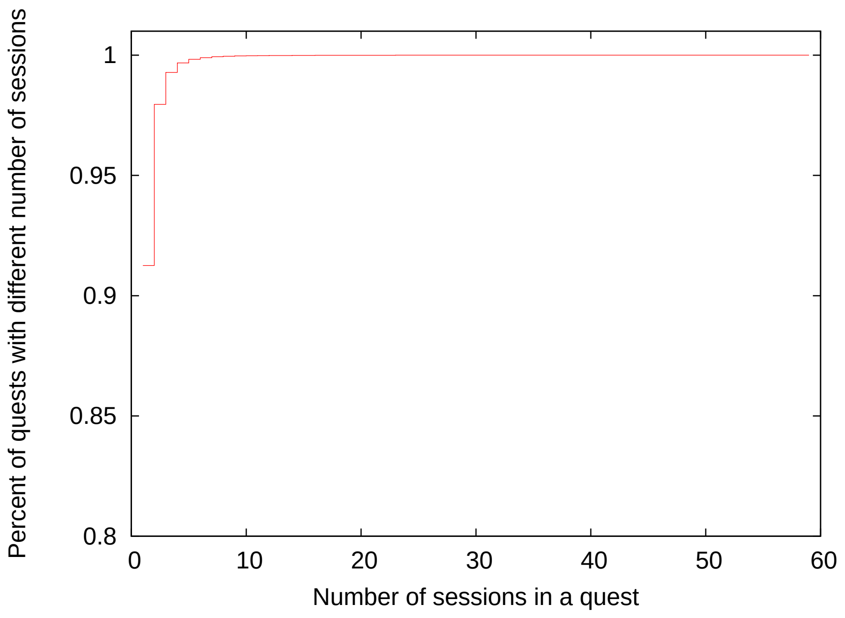

Figure 7.18: Percentage of quests that spread over different number of sessions

Chapter 8

Conclusion

In this research we tried to characterize long term user search patterns. To do this we need to use search log files, the only available means for our research. Previous work done using these logs mostly focused on short term user behavior and tried to characterize search parameters such as query length, number of viewed pages, number of clicked URLs, etc. As we meant to find long term behaviors we used the longest available log file, namely the AOL log.

An important deficiency of the log files is that these are just records of individual user queries, and there is no grouping of these activities, while the first step in characterizing user behavior is the ability to identify sequence of user search actions and the breaks between them.

The grouping of sequences of searches can be done in different ways, with several variations. We tried as a part of our research to make some order in this field. So we present two separate concepts, sessions and quests.

The notation of sessions is based on the time interval between the user queries. Silverstein et al. [27] used a global threshold for all the users and based on that categorized the users' activities. The value of the threshold varies in different studies and values such as 5, 10, 15, and 30 minutes [27, 8, 10] were used. This variability in the value of the threshold shows that there is no global value which is proper for all the users, and any value can be good just for a portion of users. We showed that using a global threshold of 10, 20 or 60 minutes affects the time interval distribution, and thus detected sessions. In particular, we received some artifacts at the global threshold regardless of its value. This happens since any global threshold causes the fusion of some short sessions into a longer one, or vise versa, dividing a long session into some shorter ones.

To obtain a better identification of sessions we suggested using a personal threshold for each user which is based on that user's activities. This personal threshold tries to give the best possible proximity to the real one. Murray et al. [22] proposed a hierarchical agglomerative clustering algorithm to find the session thresholds.

Our algorithm codifies common sense and knowledge rather than relying on more abstract parameters. Using our personal threshold algorithm we received a smooth distribution of time intervals without any artifacts. This shows that using a personal threshold is a better choice for grouping users queries into sessions. We also compared our findings with human evaluation and received high precision and recall.

The second way to group the users searches is based on the lexical similarity between the queries. The sequences of user searches combined using this method are called quests. This idea was also used before by He et al. [11] and Jansen et al. [18]. Spink et al. [4] proposed similar classifications. The query reformulation patterns help in detecting the session bounds. We used two methods to implement this idea. The first one was a simple containment method and the second was an n-gram method. In both we used the Jaccard coefficient to evaluate the similarity between the queries. Given the similarity grades between each pair of user queries we made heat-maps that were used to discover different search patterns. Using them we recognized patterns such as different kinds of repetitions or editing that have specific shapes. Our work is the first where the search patterns are recognized using such heat-maps.

We also characterized the quests found using our heat-maps. We checked their sizes, to what extent the queries in the quest are related, and their portion in the quest. We also considered the relationship between the quests, which shows repetition or reformulation of queries again and again during long term user activities.

Finally we used our personal threshold and heat-maps of quests simultaneously on the same log. This way we checked how these two different grouping methods agree with each other on the session detection issue.

We also proposed a simple method to identify navigational queries. Although there are other researchers [19, 21, 23, 2] that suggested methods to recognize them, our method is simple and does not need to collect data in advance. We found a longer time interval after the navigational queries and their fraction in web searches. We also studied non-clicked and multi-clicked navigational queries. As we worked with a long log, we identified queries that are related to news events, by noting the changes in the query popularity during different weeks.

In this research we tried to establish a common context for the two different query-grouping concepts. We provided the first steps toward long-term behavior studies, showed related characteristics and user search patterns. Our finding can be exploited in learning techniques that need to detect user sessions. It can also be used as a basis for deeper analysis of additional aspects of search such as modeling the long-term user behavior.

Bibliography

[1] M. Arlitt. Characterizing web user sessions. *Performance Evaluation Rev*, 28(2):50–56, 2000.

[2] R. BaezaYates, L. CalderonBenavides, and GonzalezCaro. The intention behind web queries. *Lecture Notes in Computer Science*, 98, 2006.

[3] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. *SIGIR'04*, pages 321–328, 2004.

[4] A. S. Bernard, B. J. Jansen, and H. C. Ozmultu. Use of query reformulation and relevance feedback by excite users. *Internet Research: Electronic Networking Applications and Policy*, 10:317–328, 2000.

[5] D. J. Brenes and D. Gayo-Avello. Stratified analysis of AOL query log. *Information Sciences*, 179(12):1844–1858, 2009.

[6] D. J. Brenes, D. Gayo-Avello, and K. Pérez-González. Survey and evaluation of query intent detection methods. *WSCD '09: Proceedings of the 2009 workshop on Web Search Click Data*, pages 1–7, 2009.

[7] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36:3–10, 2002.

[8] N. Buzikashvili and B. Jansen. Limits of the web log analysis artifacts. *Workshop on Logging Traces of Web Activity*, 2008.

[9] O. Duskin. Distinguishing humans from robots in web search logs. Master's thesis, The Hebrew University of Jerusalem,School of Computer Science and Engineering, 2009.

[10] D. He and A. Goker. Detecting session boundaries from web user logs. *Proceedings of the 22nd Annual Colloquium on Information Retrieval Research*, pages 57–66, 2000.

[11] D. He and D. J. Harper. Combining evidence for automatic web session identification. *Information Processing and Management*, 38:727–742, 2002.

[12] T. Huynh and J. Miller. Empirical observations on the session timeout threshold. *Information Processing and Management*, 45(5):513–528, 2009.

[13] B. Jansen and A. Spink. An analysis of web documents retrieved and viewed. *Proceedings of the Fourth International Conference on Internet Computing*, pages 65–69, 2003.

[14] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing and Management*, 44(3):1251–1266, 2008.

[15] B. J. Jansen and U. Pooch. A review of web searching studies and a framework for future research. *Journal of the American Society for Information Science and Technology*, 52(3):235–246, 2001.

[16] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *ACM SIGIR Forum*, 32(1):5–17, 1998.

[17] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36:207–227, 2000.

[18] B. J. Jansen1, A. Spink, C. Blakely, S. K. Murray, J. Lin, and A. Chowdhury. Defining a session on web search engines. *Journal of the American Society for Information Science and Technology*, 58:862–871, 2007.

[19] I. Kang and G. Kim. Query type classification for web document retrieval. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, 2003.

[20] T. Lau and E. Horvitz. Patterns of search: analyzing and modeling web query refinement. *Proceedings of the Seventh International Conference on User Modeling*, page 119128, 1999.

[21] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. *Proceedings of the 14th international conference on World Wide Web*, pages 391–400, 2005.

[22] G. Murray, J. Lin, and A. Chowdhury. Identification of user sessions with hierarchical agglomerative clustering. *Journal of the American Society of Information Science*, 6:3–8, 2006.

[23] D. Nettleton, L. Calderón-Benavides, and R. Baeza-Yates. Analysis of web search engine query session and clicked documents. pages 207–226, 2007.

[24] H. Pu, S. Chuang, and C. Yang. Subject categorization of query terms for exploring web users' search interests. *Journal of the American Society for Information Science and Technology*, 53(8):617–630, 2002.

[25] D. E. Rose and D. Levinson. Understanding user goals in web search. *Proceedings of the 13th International Conference on World Wide Web, ACM, New York, USA*, pages 13–19, 2004.

[26] N. Ross and D. Wolfram. End user searching on the Internet: An analysis of term pair topics submitted to the excite search engine. *Journal of the American Society for Information Science*, 51(10):949–958, 2000.

[27] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a very large Altavista query log. *ACM SIGIR Forum*, 33:6–12, 1999.

[28] A. Spink, B. Jansen, and T. Saracevic. Vox populi: The public searching of the web. *Journal of the American Society of Information Science*, 52(12):1073–1074, 2001.

[29] A. Spink, B. Jansen, D. Wolfram, and T. Saracevic. E-sex to E-commerce: Web search changes. *IEEE Computer*, 35(3):107–109, 2002.

[30] A. Spink, S. Ozmutlu, H. C. Ozmutlu, and B. J. Jansen. US versus European web searching trends. *SIGIR Forum*, 36:32–38, 2002.

[31] A. Spink, D. Wolfram, B. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American Society of Information Science*, 53(2):226–234, 2001.