SDSS Log Viewer: Visual Exploratory Analysis of Large-Volume SQL Log Data

Jian Zhang[1] Chaomei Chen[1] Michael S. Vogeley[1] Danny Pan[1] Ani Thakar[2] and Jordan Raddic[2]

[1] Drexel University, 3141 Chestnut Street, Philadelphia, USA
[2]Johns Hopkins University, 3701 San Martin Dr., Baltimore, USA

ABSTRACT

User-generated Structured Query Language (SQL) queries are a rich source of information for database analysts, information scientists, and the end users of databases. In this study a group of scientists in astronomy and computer and information scientists work together to analyze a large volume of SQL log data generated by users of the Sloan Digital Sky Survey (SDSS) data archive in order to better understand users' data seeking behavior. While statistical analysis of such logs is useful at aggregated levels, efficiently exploring specific patterns of queries is often a challenging task due to the typically large volume of the data, multivariate features, and data requirements specified in SQL queries. To enable and facilitate effective and efficient exploration of the SDSS log data, we designed an interactive visualization tool, called the *SDSS Log Viewer*, which integrates time series visualization, text visualization, and dynamic query techniques. We describe two analysis scenarios of visual exploration of SDSS log data, including understanding unusually high daily query traffic and modeling the types of data seeking behaviors of massive query generators. The two scenarios demonstrate that the *SDSS Log Viewer* provides a novel and potentially valuable approach to support these targeted tasks.

Keywords: SQL log analysis, visual exploratory analysis, multivariate data, text visualization, multiple views

1. INTRODUCTION

User-generated Structured Query Language (SQL) query logs are common in science, business, engineering, and many other domains. These logs contain information about users' data seeking behavior and system performance, and thus are rich information sources for analysts such as database administrators, system designers, and user behavior researchers. Like normal transaction logs, SQL logs can be characterized as multivariate, temporal, and categorical event sequences. On the other hand, SQL logs have their own distinct feature: semi-structured and often complex queries generated by users. While statistical methods such as analytical functions offered by database systems can reveal query patterns at aggregated levels, quickly exploring SQL logs in detail is often challenging because of the typically large volume of log data, multivariate features, and text content of queries. In this paper we, a group of information scientists collaborated with a group of data scientists of the Sloan Digital Sky Survey (SDSS), work together to design and develop an interactive visualization tool, called the *SDSS Log Viewer*, which enables and facilitates visual exploration of the SQL log data generated by users of SDSS data archive. By integrating time series visualization, text visualization, and dynamic query techniques, the tool helps SDSS data analysts to quickly identify massive query generators and reveal their data seeking models. Although this is a domain-specific case study, the method and experience of this study can be generalized for similar datasets to discover patterns and relationships between log features and user-generated textual contents.

2. BACKGROUND

2.1 The Slong Digital Sky Survey

The SDSS is the largest astronomical data collection project to date. It covers the half of the Northern sky in five wavelength bands. SDSS collects digital images of 300 million astronomical objects, and digital spectral of 1.3 million objects[1]. The project has collected 40 TB of raw data, and produced 5 TB of catalog data. All of these data are archived in the SDSS Science Archive, and are publicly accessible on the Internet. The SDSS Science Archive data products are available in two main forms: 1) raw data that can be downloaded in binary form and 2) catalog data stored in an MS SQLServer database. To access the SDSS data, one can use web-based query tools provided on the SDSS SkyServer web pages. Alternatively, users can issue SQL queries directly to the data archive through access portals. To date, the archive has served more than 160 billion queries.

2.2 SDSS SQL Query Logs

The billions of queries issued by SDSS users have been logged since 2003 and are available to the public through SDSS site[1]. The SQL log data is in the form of a table. Each row has 20 columns, or attributes. Some columns record the date and time when SQL queries were submitted; some include access information associated with users such as their IP addresses and access portals; and some indicate system performance information such as how long it took to fulfill a query and how many records were returned as a result. Most of SDSS SQL log attributes are similar to normal transaction log data, which can be characterized as multivariate event sequences with categorical and temporal characteristics. In addition to these features, the SDSS SQL log data has a unique feature: a variety of SQL queries formulated by end users in response to their own needs, which can be characterized as semi-structured text data [2]. In general, the SDSS SQL log data is an exceptionally huge volume of data with mixed data types, including numeric, categorical, and textual data in a temporal manner.

2.3 Challenges to Statistical Analysis of SDSS SQL Logs

The SDSS SQL log data contain rich information about users' data seeking behavior and system performance, and thus is a rich information source for database analysts. SDSS data scientists have applied statistical methods to the analysis of SQL log data for maintaining and optimizing the performance of their database systems. To conduct such analyses, SDSS data scientists usually issued advanced SQL codes to the log database and extract statistics about user profiles and database usage. If needed, other general analysis software such as SPSS and Excel were used to process extracted results. These methods, however, are inadequate for the analysis of the SDSS SQL log data.

First of all, without a clear idea of the types of user-generated queries in the log data, analysts have to rely on assumptions and conjectures based on their past experience. For example, it had been assumed that queries arriving at an irregular rate were made by human users, whereas queries arriving at a constant rate were sent from computer programs. But such an assumption would be false when queries generated by computer programs depend up the results of previous queries. In addition, statistic results can be misleading if data values depart from assumed distributions. For example, the mean of a group of values would not be of any use if these values follow a binomial distribution. In such cases, if the analyst can better identify the underlying distributions, subsequent statistical analysis would be more reliable. Furthermore, the semantics of a user-generated query is not taken into account in many statistical analyses. The content of a query reflects the domain knowledge of a user who formulated the query and the knowledge of SQL. If analysts are able to combine their knowledge of SDSS with query contents, they will be able to not only reveal users' data seeking behavior, but also develop a better understanding of users' data seeking intentions and fine tune their systems to serve users more efficiently.

These challenges require an exploratory data analysis of SDSS SQL logs and a better understanding of the data. Exploratory data analysis emphasizes the importance of searching for patterns and forming hypotheses before conducting confirmative analyses and reaching final conclusions [3]. Because of its capability to amplify human cognition, visualization plays an important role in exploratory data analysis [4]. In this study we applied a visual exploratory method to analyze the SDSS log data.

3. RELATED WORK

Log data is a rich source of information on users' search behavior, their interests, and how their patterns of interaction evolve over time. Visualization has been widely used to explore and analyze log files for different purposes. Existing systems and techniques normally support human-generated logs and computer-generated logs. Very few studies have focused on visualizations of SQL query logs.

3.1 Visualizing Human-generated Logs

3.1.1 Search session logs

Search session logs contain sessions, a time-stamped sequence of events, which corresponds to a user's actions, such as submitting a query to a search engine or clicking on the link of a retrieved item [5]. Given the rich contents of search queries, major search engine companies, like Google, are interested in the analysis of these log data to understand users' search behavior. Recently a few visualizations of session logs appear. An exemplar study is Session Viewer [5], which particularly supports the exploration of a large amount of session log data. Session Viewer visualizes a set of events in a session as a stack of colored rectangles. Each event is represented by a rectangle and color-coded to represent the

[1] http://skyserver.sdss.org/log/en/traffic/sql.asp

category of the event. The sequence of rectangles follows the event sequence in a session. This visual encoding method conveys session structures.

Compared to search session logs, SDSS SQL query logs have some similar features like the temporal feature, the sequence feature, and text contents. However, SDSS SQL query logs have two major features that differ from search session logs. First, search queries are normally terms without structures, whereas SQL queries are semi-structured text. Second, session logs normally have a session ID to identify a session, but SDSS query log does not have the notion of a session. IP addresses are the only way to separate distinct users. On the one hand, it is impractical to apply Session Viewer to the analysis of the SDSS log data due to these differences. On the other hand, Session Viewer's design of a stack of colored rectangles is a good example of how the structure of a session can be intuitively represented. It inspired our visual encoding design of SQL text contents.

3.1.2 Web Clickstreams

Web clickstreams is one of the most common user-generated log data. Researchers focused on visualizing users' clicksteams for usability improvement [6-9], traffic monitoring [7], understanding users' information foraging behavior [10, 11], and mapping the structure of a website [12]. As websites often have a hierarchical structure, tree structures such as hierarchical trees, tree maps, and radial trees are common choices in web clickstream visualization. Exemplar systems include radial tree in [9] and hierarchical tree in WebQuilt [8]. Since web clickstreams may include a loop, network visualization is also applicable as demonstrated in WebViz and WebQuilt. In general, web clickstream logs contain paths of a fixed set of web pages, thus differ from the SDSS SQL query logs. Existing systems are inadequate for visually exploring the SQL log data.

3.1.3 User-generated Text Contents

SDSS SQL logs contain user-generated text contents. Such contents become common in the Web 2.0 era. Tremendous text contents have been created by users, and also been visualized for various research purposes. Due to its popularity and availability of data, text contents of Wikipedia attract many visualization studies [13-20]. While some of these studies focused on the metadata of Wikipedia entries, such as authors [18-20] and editing frequencies [13], a few traced the revision history of text contents [15, 16]. For example, the history flow system [16] divides the text of a Wiki document into segments, and then represents each segment as a bar and color-codes these bars based on their authors, thereby transforming a Wiki document into a line of color-coded bars whose length are proportional to the lengths of corresponding segments. Different revision versions become a set of lines. Alignment of these lines based on their revision time can help analysts to reveal cooperative and conflict patterns. Later the same visual encoding strategy was used in the chromogram visualization for transforming words into images [15].

Similar to Wiki documents, revision and editing of software codes are common tasks. Visualization methods have been used to illustrate the revision and editing history of computer software. For an overview of software visualization, readers can refer to [21]. Exemplar studies like [22, 23] tokenized codes into pieces and color-coded these pieces based on features related to analysis tasks.

These studies underline a recurring theme of transforming text contents into images so as to reduce the recognition load of reading text to looking at images. Textual contents are first divided into tokens, e.g. paragraphs, sentences, and words, and then colors are assigned to these tokens based on features related to targeted analysis tasks, such as revision times and authors. To visualize such color-code tokens, various shapes are used, such as lines [16], rectangles [15], areas [23], and ellipses [14]. These shapes can be subsequently aligned based on other features such as temporal relations. Resultant visualizations show patterns, trends, and anomalies. In this study we adopted this visual encoding method in our visualizations of SQL text contents.

User-generated text contents in social media such as Twitter and Facebook have increased dramatically in recent years. Visualizations of these short and sometimes compact texts can be used to increase the awareness of ongoing events [24] or analyze news stories [25]. A common approach used in these studies is to represent time as a linear ordered axis and plot the volume of events in stacked areas. Text contents are usually represented by a small number of topic terms, which are plotted within or outside stacked areas. Such visualization methods have been used to analyze time-oriented events as seen in ThemeRiver [26], EventRiver [27], and TIARA [28]. Unlike SQL query contents, user-generated social media contents are usually in the form of free text. Therefore, the "bag of words" method is normally used to extract topic terms to represent a collection of documents. SQL queries, in contrast, have clearly defined structures. The semantics will be lost without these structures. In our case, the "bag of words" method thus is inappropriate.

3.2 Visualizing Computer-generated Logs

Computer-generated logs are mainly associated with events that occur inside a computer system without direct involvements of users, such as network traffic and server activities. Many network management and administrative systems explore and monitor their computer-generated logs for administrative purpose, such as traffic monitoring and intrusion detection. Exemplar studies include visualizations of event log in LogView [29], visual browsing computer-log files in MieLog [30], and NFlowViz [31]. Some features in computer-generated logs are similar to SQL query logs, e.g. attributes about performance and access. For these features, the designs for computer-generated log visualization can also help this study. For example, LogView [29] clustered system events into a few groups and visualized groups and individual events in a squerified treemap. Since categorical attributes in SDSS SQL logs have a finite number of values, this visual encoding method is useful for this study.

3.3 Visualizing SQL Queries

Non-visual analyses of SQL queries for database tuning or query optimization are common research activities. But visualizations of user-generated queries are rare. QueryScope [32] visualized the nested structure of a query and its sub-queries (queries within a query). A query and its sub-queries are shown as circles. Tables of the underlying database in queries are shown as filled circles within query circles and color-coded based on their features. Based on this visualization, patterns of nested structures can be easily explored. QueryScope used multiple circles to represent one query, thus it needs a lot of display areas. It is hard to scale it up to display tens of thousands of queries simultaneously. Therefore, this visual encoding method is not particularly suitable for analyzing the large volume of SDSS queries.

4. THE DESIGN OF THE *SDSS LOG VIEWER*

We have outlined the state of the art of visual exploratory analysis of a variety of log data. Few existing tools, however, are readily applicable to the analysis of the SDSS SQL logs. Therefore we designed an interactive visualization tool, called the *SDSS Log Viewer,* to enable and facilitate the analysis of the SDSS SQL logs. Our design adopted a user-centered design methodology. The overall process includes a requirement analysis and several rounds of rapid prototyping. This section first introduces the requirement analysis, which builds the foundation for design rationales, and then describes the design and implementation of the tool in detail. Due to the limit of the length of the article, we omit the process of prototyping and implementation.

4.1 Requirement Analysis

In a user-centered design, the first step is to identify the primary target users. One group of target users of this study includes the SDSS database scientists at the Johns Hopkins University (JHU). This user group is responsible for the daily operation of the SDSS Archive. They have comprehensive knowledge of the SDSS project and the data archive. They also have advanced knowledge of SQL and are capable of quickly identifying users' query intentions based on query contents. The second target user group constitutes researchers from information science (the first and second authors). Information scientists are interested in users' information seeking behavior. User-generated SQL queries for the SDSS data are a valuable source for the study of users' interests in the SDSS data and how they made use of the publically available resources. On the other hand, they need to collaborate with users in the first group users so as to interpret and validate patterns and trends identified in the visual exploratory analysis of the SDSS SQL queries.

In this study, requirements were collected from both groups (called analysts hereafter). To ground the design, the two groups of users met regularly to discuss and update each other's analysis goals, tasks, workflows, and analysis cases. Based on the discussions, the primary goals of the *SDSS Log Viewer* are set as follows:

1) to facilitate the identification of unusual query traffic, and

2) to rapidly explore user-generated SQL contents to answer questions concerning who submitted a particular sequence of queries, what seem to be their intention, and whether these activities connected to other attributes in the log.

More specific design requirements were derived subsequently in order to achieve these goals. Here we highlight four major requirements identified:

1. Support the exploration of time series of query traffic

The first goal is to identify unusual query traffic. Analysts are interested in overall traffic trends, such as, how query traffic changes over time, how traffic changes in a certain time interval from hours to days, and how traffic changes on a larger scale of time intervals such as months and years. Visual exploration could help analysts identify unusual patterns and enable analysts to narrow down the scope of their inspection quickly and focus on specific queries. A tool for visual exploration thus needs to provide an overview of query traffic in meaningful temporal granularities such as days or months. The system should enable analysts to perform comparisons easily.

2. *Support quick exploration of massive data*

Once analysts pick the log data for detailed exploration, they face a huge volume of heterogeneous data, including multivariate categorical and numeric values and semi-structured SQL query contents. They need to quickly identify major sources of queries of interest, browse the contents of SQL queries to understand their meanings, infer intentions of queries, and reveal potential patterns by associating attributes with their semantics. To support these analytic activities, the system needs to provide intuitive visual representations and display a large number of SQL queries in a compact view. In addition, the system should be flexible and highly interactive such that the analyst can freely examine patterns across the high dimensional array of attributes.

3. *Support basic statistics of log attributes*

One of the important goals of exploratory data analysis is to understand basic distributions of data. Analysts need such basic statistics to form further confirmative analyses. So the system needs to offer some basic statistics results of log attributes.

4. *Support dynamic queries*

Once analysts discover interesting cases, they will need to isolate these cases for further examination. Analysts would need to filter out records based on values of a log attribute or a combination of several attributes. Visual information seeking activities should be supported by the system in terms of a dynamic query filter [33]. The dynamic query filter needs to collect values of log attributes from data under analysis and enable analysts to select values of attributes that are of interest to them.

4.2 Visual Interfaces of the *SDSS Log Viewer*

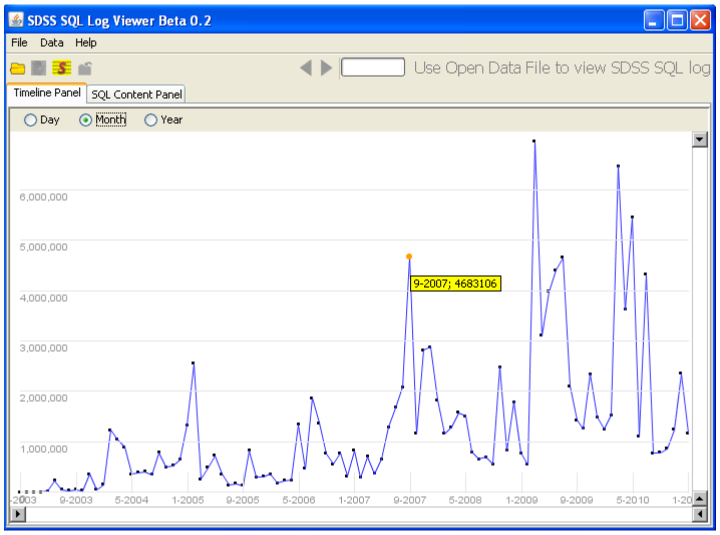


Figure 1. Timeline View.

To fulfill the design requirements, the *SDSS Log Viewer*'s user interface provides multiple views for visual exploratory analysis of SDSS SQL logs. The Timeline View (Figure 1) serves as the starting point of visual exploration. This design is motivated by a requirement identified in the initial task analysis interview. Our target users are particularly interested in temporal traffic patterns. Analysts can view traffic trends across different temporal units, such as a day, a month, or a year. Once analysts spot unusual query patterns, they need to load the related SQL log data into the system to launch a three-view SQL Content Panel (Figure 2) for visual exploration of queries at the content level.

The SQL Content View presents user-generated SQL queries as lines of color-coded bars. This design enables analysts quickly spot patterns, trends, and anomalies without reading actual texts. Meanwhile the SkyMap View plots the spatial information specified by users in their queries on a two-dimensional map of the Universe. This is a unique property of

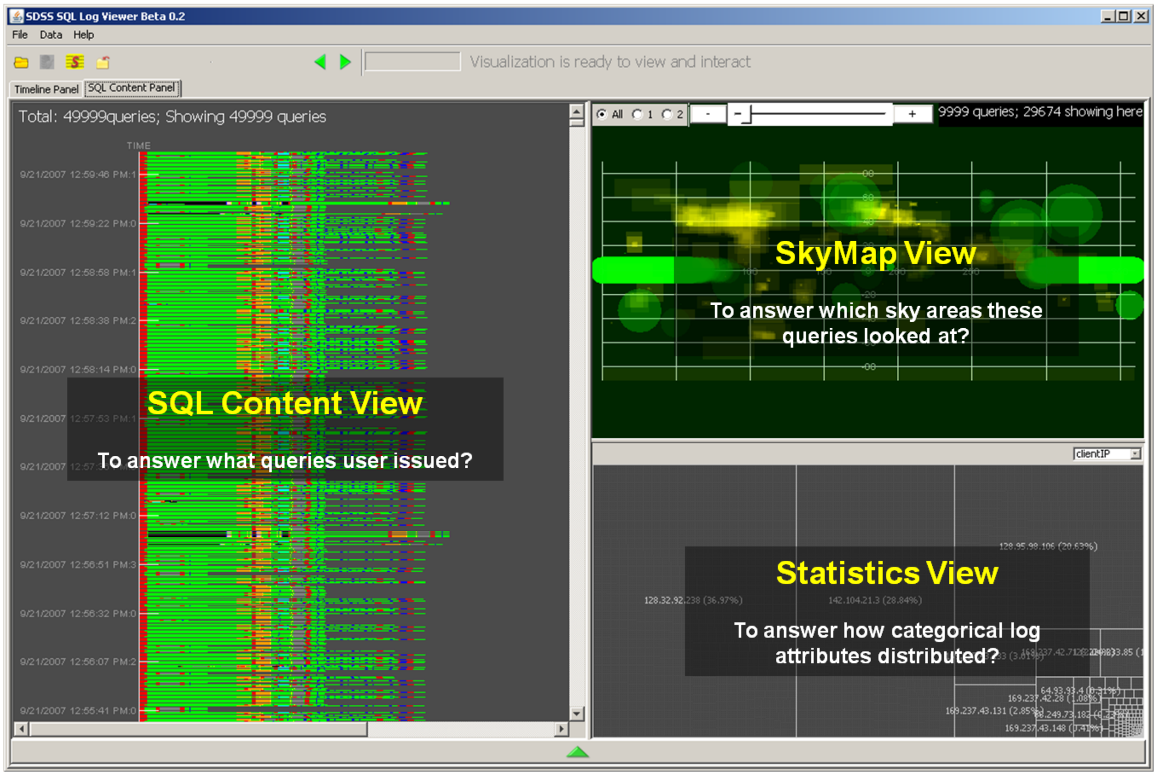
SDSS queries because of the astronomical nature of the data. In this view analysts can pinpoint the precise areas of the sky that users intended to search for SDSS data. The Statistics View displays basic statistic distributions of categorical attributes using a treemap visualization design. When analysts need to filter queries further, they can activate a Dynamic Query Menu (Figure 4) by clicking on the dynamic query button at the bottom of the Panel. A group of filters are provided so that one can control the display of various attributes in the three views in the SQL Content Panel. The next four subsections describe each view of the *SDSS Log Viewer*.

4.2.1 Timeline View: Exploring Query Traffic

In visual exploratory analysis, providing a context for data interpretation is crucial. Therefore an overview of SQL query traffic is provided in the Timeline View to represent the context of query frequencies. Analysts can choose one of three temporal units, a day, a month, and a year, to view traffic trends. If analysts only want to view a certain time range, dragging the two sliders at the bottom of Timeline View will adjust the time range and display the traffic within the chosen range only. The volume of the SDSS SQL query traffic varies dramatically over time. Therefore analysts need to have the ability to adjust the range of frequency at different scales. Such adjustments can be done by controling the two sliders on the right-hand edge of Timeline View.

4.2.2 SQL Content View: Compact Display of Queries

The major challenge of exploring the contents of user-generated queries is how to help analysts to discover patterns, trends, and anomalies with a minimum amount of reading of the actual text in the large amount of queries. Reading query contents is a huge overload to analysts. It is unnecessary and unrealistic for an analyst to read every word in queries. Besides, much of the SDSS queries are complex due to the complexity of the underlying SDSS data. Showing analysts the original query text may cause two major problems. One is the scalability of display and another is the perceptual overload. Displaying a large amount of text legible enough on screen not only consumes many computer memories, but also creates perceptual barriers for analysts. Analysts need to move among different screens and carry all information in their head. Therefore, we choose to encode the original query text visually and highlight structural properties of queries.



Visualization is ready to view and interact

Total: 49999queries; Showing 49999 queries

9999 queries; 29674 showing here

To answer what queries user issued?

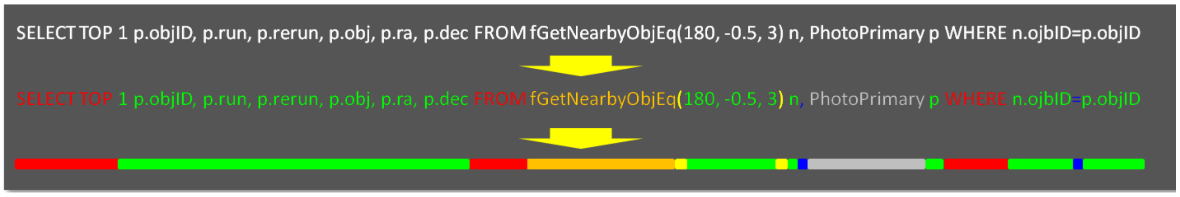
To answer which sky areas these queries looked at?

To answer how categorical log attributes distributed?

Figure 2. SQL Content Panel: SQL Content View (Left), SkyMap View (Upper Right), Statistics View (Lower Right), and the Dynamic Query Button (the bar at the bottom).

SDSS data scientists are advanced SQL programmers. They could identify the basic logic of a query by only reading their structure because, similar to other programming languages, SQL follows a strict grammar and a set of reserved words. For example, the keyword “SELECT” normally appears at the beginning of a query. Some special chars such as “=” and “>” mainly appear as part of a condition clause. In SDSS queries, there are functions and procedures created by SDSS database managers. These functions and procedures generally only appear after certain keywords, like “FROM”.

Based on these specifications, we designed our visual interfaces based on positions of SQL terms and their types. It is called color-coded query line (Figure 3). We first tokenize a line of query into terms and chars, and categorize them into ten types as shown in Table 1. These ten types are based on MS SQL Server’s category of tokens[2] and discussions with SDSS data scientists. Then each category is assigned a distinct color. The color schema is partially based MS SQL Server’s tradition. The ten colors are chosen from primary colors suggested by [34] (page 125). We keep the total number of types lower than one dozen for distinguishable color coding[34].



```
SELECT TOP 1 p.objID, p.run, p.rerun, p.obj, p.ra, p.dec FROM fGetNearbyObjEq(180, -0.5, 3) n, PhotoPrimary p WHERE n.ojbID=p.objID
SELECT TOP 1 p.objID, p.run, p.rerun, p.obj, p.ra, p.dec FROM fGetNearbyObjEq(180, -0.5, 3) n, PhotoPrimary p WHERE n.ojbID=p.objID
```

Figure 3. Visual encoding of a SQL query to a color-coded query line.

After tokenization and assignment of colors, a line of horizontal bars is created to represent the original query text. Each bar represents a token. The length of a bar is proportional to the number of characters in the corresponding token. The length could help to distinguish tokens of the same type, and the color of a bar is same as the color assigned to the token. This visual encoding method turns a text reading task to a visual scan task. It reduces the cognitive load of exploring a large amount of queries, because analysts are processing patterns of visual attributes instead of reading texts[30].

In the SQL Content View each line of color-coded bars represents one line of user-generated query. These lines are vertically aligned based on their issued time. Recent queries are shown as lines with higher positions. The background color of this view is set to dark gray so that all ten colors can be visible. To reduce visualization dimension, we use one pixel for the height of a line. Analyst can use scroll bars to explore the entire view.

Table 1. SDSS SQL token types

Type	Color		Examples
SQL keyword	Red	🟥	select, from, where
SQL function	Pink	🟪	count, sum, desc
SQL operators	Blue	🟦	+, -, =
Users’ input	Green	🟩	column names
SQL special character	Yellow	🟨	“(”, “)”
String in quotation	Black	⬛	html strings
SDSS defined function	Orange	🟧	fGetNearbyObjEq
Comment	White	⬜	string start with “--”
SDSS table	Cyan	🟦	Photoz, Field
SDSS view	Gray	▨	Galaxy, Star

4.2.3 SkyMap View: Spatial Visualization

SDSS query logs reflect users’ scientific knowledge of the Universe. Database designers created functions to allow users to specify spatial constraints in their queries. This spatial information is crucial to SDSS analysts because it indicates which areas of the sky were queried frequently, and how queried were distributed over the areas of the sky.

Two types of spatial information of sky areas can be extracted from SDSS query logs. One is a circular sky area, and another is a rectangular sky area. Users can specify a circular area with a center point, in terms of a right ascension (RA) and a declination (DEC), and a radius in arcmins. RA and DEC are the two coordinates of a point on the celestial sphere in the equatorial coordinate system. RA is celestially equivalent of terrestrial longitude and DEC is equivalent to latitude.

[2] http://msdn.microsoft.com/en-us/library/aa299742(v=SQL.80).aspx

They are normally measured in degrees. For the “rectangular search”, users need to specify two points, one for top-left corner and one for bottom-right corner, with RAs and DECs.

In the current version, we used a rectangular 2D Universe map in our visual display (seen in Figure 2). This method plots circular and rectangular areas in an X-Y coordinate system where the X-axis represents RA and the Y-axis represents DEC. A black background is used as it is conventional for visualizations of the Universe. Circles (in green) and rectangles (in yellow) are directly rendered on the 2D map without projections in order to speed up rendering process. As many shapes are overlapped, we use transparency to differentiate overlapped regions. The brighter a region, the more overlapped circles and rectangles in that region; therefore, the more queries were targeting that region. Analysts can adjust the level of transparency with a slider bar on the top of the SkyMap View. Lower transparency levels can show tiny areas specified, whereas higher transparency levels can show highly overlapped regions. Analysts can interact with the map by zooming and panning for checking specific regions, and tooltips for details-on-demand.

4.2.4 Statistics View: Distributions of Categorical Attributes

To provide analysts with distributions of categorical attributes, the Statistic View applies a squarified treemap layout [35]. Compared to other techniques for displaying distributions of categorical data such as histograms, the treemap layout has two advantages for our tasks. First, this space-filling technique fully exploits the available display space without empty spaces which are normally seen in histograms. This feature is crucial to our application since only a quarter of the SQL Content Panel is available. Second, this technique can display individual visual items, thus giving further improvement opportunities, e.g. directly operations on these items such as color-coding item based on other features.

In the Statistics View, each query is represented as a rectangle. Rectangles with the same log attribute value are grouped into a large rectangle, which is labeled with the attribute value and its percentage in this log attribute. All rectangles are organized automatically to fill the entire view.

4.2.5 Dynamic Query

In visual exploratory analysis, the ability to isolate interesting cases out for in-depth analyses is important. We support this requirement by a Dynamic Query Menu (Figure 4).

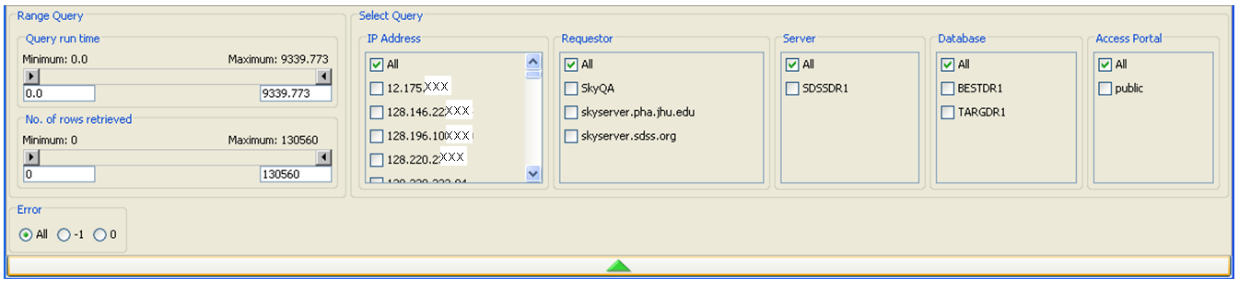


Figure 4. Dynamic Query menu, including a group of range query filters, a group of select query filters, and a radio select filters.

We group SDSS log attributes into three groups: attributes with numeric values, attributes with categorical values, and one special attribute, and support with different filters. For numeric attributes, e.g. the number of rows returned to a query, range sliders are used to help analyst set a value range (minimum and maximum). For categorical attributes, e.g. IP addresses, a group of checkboxes are used. Since “error” attribute only has three fixed values, we used radio buttons to set its values. All values in the Dynamic Query Menu are created dynamically based on data loaded in the system. When analysts set a value in a filter, only queries that satisfy selected values will be displayed in the three views at the SQL Content Panel and are interactive.

5. ANALYSIS SCENARIOS

In this section, we describe two scenarios that we analyzed SDSS users’ data seeking behavior by using the *SDSS Log Viewer*. The two analysis scenarios are of interests to the SDSS data scientists at the JHU because they want to profile their major query generators.

5.1 Days with Million Queries

The overview of daily SDSS query traffic shows that there are a few days with more than one million queries, including December 6, 2007, October 22, 2008 and a few days around March 8, 2009 (Figure 5).

We loaded data of each of these days to check who were responsible for these million-per-day queries, what they were doing, and their possible intension. From their query contents (Figure 6), we found out that these million queries were sent from two IP addresses only. One is a university in Canada[3], who dominated the first and the second million query days. Another user, also in Canada, dominated the days around March 8, 2009. SDSS analysts recognized that both institutes might retrieve SDSS data for the Canadian Astronomy Data Centre.

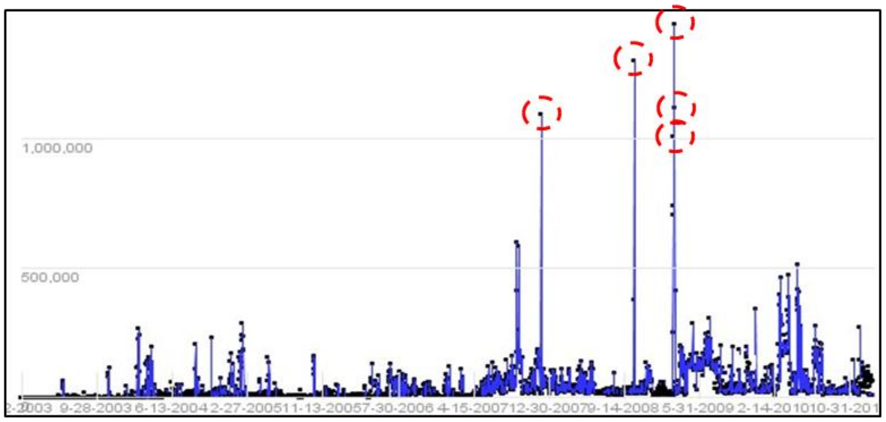


Figure 5. Million query days marked with red circles in the year of 2007, 2008, and 2009.

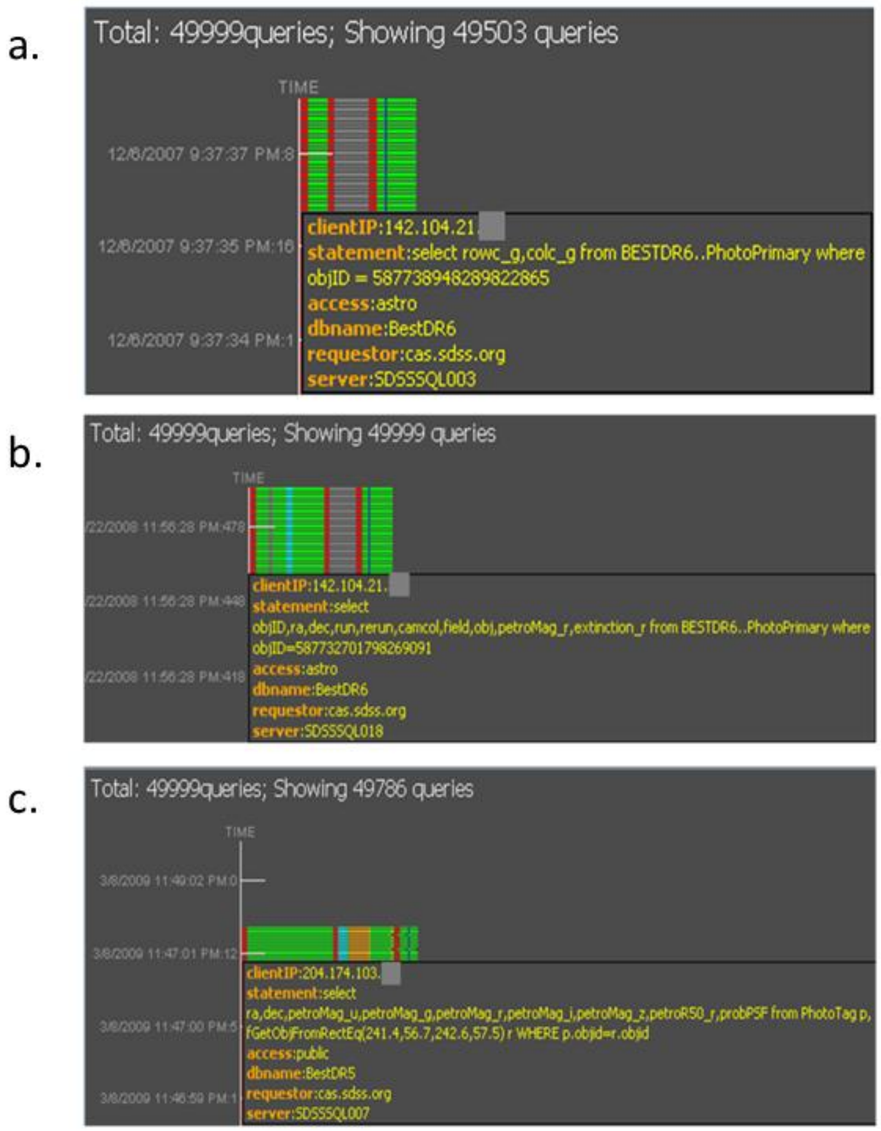
Observations of the visualizations of SQL contents quickly revealed that both of the million query generators have the same query pattern. They created computer programs to automatically generate queries with the same structure but different values of the parameter “objID”, or rectangular areas (see Figure 6 a-c). These “machine gun” queries were sent to the SDSS data archives one after another with less than 0.1 second of a gap in between.

After identifying this query pattern, SDSS analysts wondered whether all of these queries had been executed because the short time interval between queries would violate the SDSS “fair use” policy. The policy states that one IP address can only send one query per second. So we checked the error status of these queries (the function is shown at the bottom left corner in Figure 4). As it turned out, only 3.4% of the millions of queries submitted were actually executed in SDSS databases. The rest of these queries were rejected due to their violation of the SDSS query policy.

Given the observations that both million query generators (IPs) are geographically close, both have the same query pattern, and both have a similar success rate of executed queries, SDSS analysts inferred that the two million query generators might have used the same software and adopted the same query strategy. Even though the majority of queries were rejected for execution, successful ones were apparently encouraging enough for the senders to keep using their code and strategy. However, SDSS analysts pointed out that there is an existing tool on the SDSS website that can fulfill the same data retrieval purpose more efficiently and effectively. The million query senders were probably not aware of the more efficient tool. Alternatively, they were probably testing their own tool for development purposes.

The two million query generators’ behavior appears to fit what is described by the theory of “principle of least effort” [36], which states that regardless the user’s proficiency as a searcher, or their level of subject expertise, they will tend to use the most convenient methods available to them, and stop searching as soon as acceptable results are found. Results of this case study suggests that data retrieval might follow the same principles as information retrieval and designers of large scale data archive need to incorporate these principles into their design to better serve their users.

[3] The specific name of the university is omitted for privacy considerations.



Total: 49999queries; Showing 49503 queries

```
clientIP:142.104.21.[redacted]
statement:select rowc_g,colc_g from BESTDR6..PhotoPrimary where
objID = 587738948289822865
access:astro
dbname:BestDR6
requestor:cas.sdss.org
server:SDSSSQL003
```

Total: 49999queries; Showing 49999 queries

```
clientIP:142.104.21.[redacted]
statement:select
objID,ra,dec,run,rerun,camcol,field,obj,petroMag_r,extinction_r from BESTDR6..PhotoPrimary where
objID=587732701798269091
access:astro
dbname:BestDR6
requestor:cas.sdss.org
server:SDSSSQL018
```

Total: 49999queries; Showing 49786 queries

```
clientIP:204.174.103.[redacted]
statement:select
ra,dec,petroMag_u,petroMag_g,petroMag_r,petroMag_i,petroMag_z,petroR50_r,probPSF from PhotoTag p,
fGetObjFromRectEq(241.4,56.7,242.6,57.5) r WHERE p.objid=r.objid
access:public
dbname:BestDR5
requestor:cas.sdss.org
server:SDSSSQL007
```

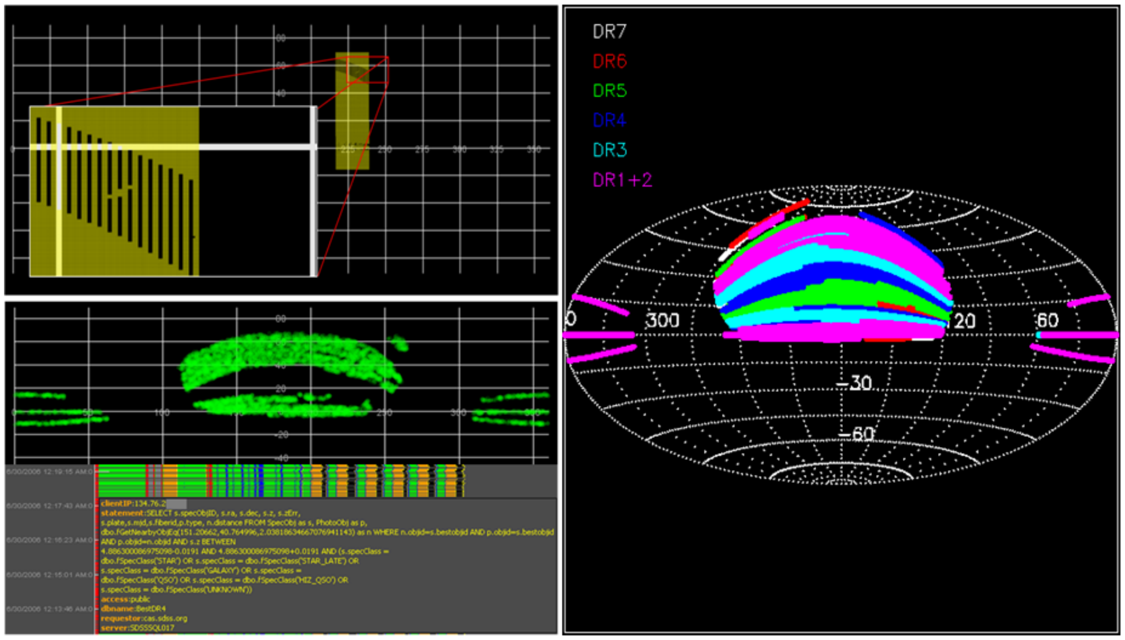
Figure 6. (a) Million queries in Dec. 6th, 2007 from one IP in the same template but different objID numbers; (b) Million queries in Oct. 22nd, 2008 from one IP in the same template but different objID numbers; (c) Million queries in Mar. 8th, 2009 from one IP in the same template but different rectangular areas

5.2 Common Models of Massive Query Generators

Besides these million query days, we also explored other heavy traffic days of queries in each year. Visual exploration of query contents in these days reveals that only a few (usually one to three) massive query generators were responsible for such unusually high traffic. They automatically generated SQL queries and used a few query templates with very few variations. In-depth analysis revealed several common query models of these massive query generators. Here we describe three common ones.

5.2.1 Scan queries

Among massive query generators, some of them systematically scan a very large region of the sky, e.g. the Northern sky or even the entire sky, step by step with fairly simple queries. This kind of queries can be easily detected in the SkyMap View. Figure 7 (Upper Left) shows a fraction of a quarter million queries that scanned the entire Northern sky with small rectangles but deliberately ignored some areas mapped by the SDSS project.



```
clientIP:134.76.2[redacted]
statement:SELECT s.specObjID, s.ra, s.dec, s.z, s.zErr,
s.plate,s.mjd,s.fiberid,p.type, n.distance FROM SpecObj as s, PhotoObj as p,
dbo.fGetNearbyObjEq(151.20662,40.764996,2.0381863466707694l143) as n WHERE n.objid=s.bestobjid AND p.objid=s.bestobjid
AND p.objid=n.objid AND s.z BETWEEN
4.886300086975098-0.0191 AND 4.886300086975098+0.0191 AND (s.specClass =
dbo.fSpecClass('STAR') OR s.specClass = dbo.fSpecClass('STAR_LATE') OR
s.specClass = dbo.fSpecClass('GALAXY') OR s.specClass =
dbo.fSpecClass('QSO') OR s.specClass = dbo.fSpecClass('HIZ_QSO') OR
s.specClass = dbo.fSpecClass('UNKNOWN'))
access:public
dbname:BestDR4
requestor:cas.sdss.org
server:SDSSSQL017
```

Figure. 7. (Upper Left) A fraction of a quarter million queries that scan the entire Northern sky, but deliberately miss some SDSS mapped area. (Lower Left) Typical search queries originated in Germany that searched for high redshift objects only in SDSS mapped areas. The SkyMap in the upper panel shows that these queries were restricted to SDSS mapped areas only. The lower panel is the visualization of these queries as query lines with a tooltip window containing detailed information of a query. (Right) Sky areas covered by SDSS 1~7 Data Releases[4]. The query trails shown in the Lower Left panel are confined to the SDSS coverage area.

5.2.2 Search queries

Unlike scan queries, search queries focused on specific sky regions, normally areas mapped by the SDSS project. Search queries contain specific and often complex condition clauses to locate astronomical objects that are of interest to the senders of queries. Visual patterns of search queries are shown in both the SkyMap View and the SQL Content View. Figure 7 (Lower Left) shows a source of queries originated in Germany searching for high redshift stars, galaxies, quasars, or unknown objects within the SDSS coverage area.

5.2.3 Retrieve queries

When data seekers know what objects they want, they would use retrieve queries to obtain data. A common way to retrieve data is to specify object IDs in their queries and select relevant columns in data tables. This type of queries do not contain spatial information explicitly, therefore no visual patterns can be extracted from such queries in the SkyMap View. On the other hand, these queries can be easily detected in the SQL Content View because of their short condition clause as shown in Figures 6a and 6b. Typical visual patterns of retrieve queries have very short query lines.

6. CONCLUSION AND FUTURE WORK

In this study we have presented the *SDSS Log Viewer*, an interactive visualization tool designed for visual exploratory analysis of SDSS SQL logs, which involves spatial, temporal, multivariate, and semi-structured text data of a huge volume. Our work provides a novel approach to address the needs of visual exploratory analysis. Time series analysis, text visualization for semi-structured text, and dynamic query are integrated to explore and analyze users' data seeking behavior with reference to the underlying astronomical nature of the data. We expect that this design will be valuable in other applications with similar data and needs, for example, geospatial and biochemical data archives.

The SDSS Log Viewer reduces users' cognitive load of reading through a large volume of queries as text by tapping into our perceptual ability to recognize visual patterns from color-coded simple images. Although it is possible to arrive at the same insights regarding the two million query generators by reading through individual queries, reading and

[4] http://www.sdss.org/DR7/coverage/

comparing patterns shared by millions of queries will be extremely time consuming. In contrast, in a matter of minutes, we were able to recognize a pattern that is shared by this magnitude of query traffic with the help of the visual exploration tool. In addition, patterns of sky scanning queries would be very hard to detect without the help of a tool such as the SkyMap View.

The SDSS data scientists at the JHU have found the *Log Viewer* to be quite useful in identifying and understanding unusual query patterns, and their feedback indicates that the tool has great potential in facilitating their future analysis needs. They commented that:

> *"The visualizations analysed (e.g. being able to spot certain patterns at a glance) are especially useful due to the exceptionally large volume of queries. It is not feasible to examine queries individually or even in groups with ad hoc techniques. The visualization offered by the* SDSS Log Viewer *will be absolutely crucial in further in-depth analysis of the log data."*

While the value of the *SDSS Log Viewer* has been demonstrated and tested, the two analysis scenarios also revealed limitations that further improvements may address. First, although the *Log Viewer* is efficient in finding regular patterns formed by a large volume of automatically created queries, it is hard for analysts to check individual queries if the number of queries is too small and they were distributed sparsely over time. Such queries are normally generated manually by users. It is known in cyber security research that similar strategies were used by intruders to keep a low profile and avoid being detected. SDSS data scientists are also interested in how users' knowledge of SQL improving over time. For the analysis of a small number of queries, directly reading queries as text is probably more efficient than our visual exploration approach. Next, analysis scenarios revealed that once analysts identified patterns, e.g. a large number of similar queries, in the SQL Content View, they wanted to group these queries into an even smaller visual representation so as to further reduce cognitive load. Such needs will require the *Log Viewer* to not only present information, but also provide analytical functions to automatically find queries having similar structures. Further improvement of the *Log Viewer* will transform it from an information visualization system to a visual analytics system. While the SDSS sky survey is by far the most extensive effort in terms of its coverage, astronomers are planning even larger-scale sky surveys. Many lessons learned from how users access the SDSS data would be valuable for future sky surveys.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. IIS-0612129.

NOTES

The software is available at http://nevac.ischool.drexel.edu/~james/SDSSLogViewer/SDSSLogViewer.html. A video is available at http://youtu.be/zJynMAByz3E.

REFERENCES

[1] D. G. York, J. Adelman, J. E. Anderson *et al.*, "The Sloan Digital Sky Survey: Technical summary," Astronomical Journal, 120, 1579-1587 (2000).

[2] J. J. Thomas, and A. K. Cook, [Illuminating the Path: The research and development agenda for visual analytics] IEEE Press, (2005).

[3] J. W. Tukey, [Exploratory Data Analysis] Addison Wesley, Reading, MA USA(1977).

[4] H. Lam, [Visual Exploratory Analysis of Large Data Sets: Evaluation and Application] The University of British Columbia, Vancouver(2008).

[5] H. Lam, D. Russell, D. Tang *et al.*, [Session Viewer: Visual Exploratory Analysis of Web Session Logs] IEEE, Sacramento, CA USA(2007).

[6] H. Hochheiser, and B. Schneiderman, "Using Interactive Visualizations of WWW Log Data to Characterize Access Patterns and Inform Site Design," Journal of the American Society for Information Science and Technology, 52(4), 331-343 (2001).

[7] H. E. Chi, "Improving Web Usability Through Visualization," IEEE Internet Computing, 6(2), 64-71 (2002).

[8] S. J. Waterson, J. I. Hong, T. Sohn *et al.*, [What did they do? Understanding clickstreams with the WebQuilt visualization System], (2002).

[9] B. Wong, and G. Marsden, [Effectively Exploiting Server Log Information for Large ScaleWeb Sites], (2001).

[10] P. Pirolli, and S. Card, "Information Foraging," Psychological Review, 106, 643-675 (1999).

[11] A. Wexelblat, and P. Maes, [Footprints: History-Rich Tools for Information Foraging] ACM, Pittsburgh, PA USA(1999).
[12] J. E. Pitkow, and K. A. Bharat, [WebViz: A tool for Worldwide web access log analysis], (1994).
[13] B. Suh, E. Chi, A. Kittur *et al.*, [Lifting the Veil: Improving Accountability and Social Transparency in Wikipedia with WikiDashboard], Florence, Italy(2008).
[14] B. Otjacques, M. Cornil, and F. Feltz, "Visualizing Cooperative Activities with Ellimaps: The Case of Wikipedia," Lecture Notes on Computer Science, 5738, 44-51 (2009).
[15] M. Wattenberg, F. Viegas, and K. Hollenbach, "Visualizing Activity on Wikipedia with Chromograms," Lecture Notes on Computer Science, 4663, 272-287 (2010).
[16] F. Viegas, M. Wattenberg, and K. Dave, [Studying Cooperation and Conflict between Authors with *History Flow* Visualizations], Vienna, Austria(2004).
[17] T. Iba, K. Nemoto, B. Peters *et al.*, "Analyzing the Creative Editing Behavior of Wikipedia Editors Through Dynamic Social Network Analysis," Procedia Social and Behavioral Sciences, 2, 6441-6456 (2010).
[18] D. Fisher, and P. Dourish, [Social and Temporla Structures in Everyday Collaboration], Vienna, Austria(2004).
[19] U. Brandes, and J. Lerner, "Visual Analysis of Controversy in User-generated Encyclopedias," Information Visualization, 7, 34-48 (2008).
[20] A. Balakrishnan, S. Kiesler, A. Kittur *et al.*, [Pitfalls of Information Access with Visualizations in Remote Collaborative Analysis], Savannah, GA(2010).
[21] S. Deihl, [Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software] Springer, Berlin, Germany(2007).
[22] S. Eick, J. Steffen, and E. Sumner, "Seesoft - A Tool for Visualizing Line Oriented Software Statistics," IEEE Transaction on Software Engineering, 18(11), 957-968 (1992).
[23] E. Gilbert, and K. Karahalios, "CodeSaw: A Social Visualization of Distributed Software Development," Lecture Notes on Computer Science, 4663, 303-316 (2007).
[24] M. Dork, D. Gruen, C. Williamson *et al.*, "A Visual Backchannel for Large-Scale Events," IEEE Transaction on Visualization and Computer Graphics, 16(6), 1129-1136 (2010).
[25] N. Diakopoulos, M. Naaman, and F. Kivran-Swaine, [Diamonds in the Rough: Social Media Visual Analytics for Journalistic Inquiry], Salt Lake City, UT, USA(2010).
[26] S. Havre, E. Hetzler, P. Whitney *et al.*, "ThemeRiver: Visualizing Thematic Changes in Large Document Collections," IEEE Transactions on Visualization and Computer Graphics, 8(1), 9-20 (2002).
[27] D. Luo, J. Yang, J. Fan *et al.*, [EventRiver: Interactive Visual Exploration of Streaming Text], Bordeaux, France(2009).
[28] S. Liu, M. Zhou, S. Pan *et al.*, [Interactive, Topic-based Visual Text Summarization and Analysis], Hong Kong, China(2009).
[29] A. Makanju, S. Brooks, N. A. Zincir-Heywood *et al.*, [LogView: Visualizing Event Log Clusters], Fredericton, Canada(2008).
[30] T. Takada, and H. Koike, [MieLog: A Highly Interactive Visual Log Browser Using Information Visualization and Statistical Analysis] SAGE, Philadelphia, PA USA(2002).
[31] F. Mansmann, F. Fischer, D. Keim *et al.*, "Interactive Analysis of NetFlows for Misuse Detection in Large IP Networks," Dfn Forum Kommunikationstechnologien - Verteilte Systeme Im Wissenschaftsbereich, (2009).
[32] L. Hu, Y.-C. Chang, C. Lang *et al.*, [QueryScope: Visualizing Queries for Repeatable Database Tuning], Auckland, New Zealand(2008).
[33] C. Ahlberg, and B. Shneiderman, [Visual information seeking: tight coupling of dynamic query filters with starfield displays] ACM, (1994).
[34] C. Ware, [Information Visualization: Perception for Design] Morgan Kaufmann, San Francisco, CA(2004).
[35] M. Bruls, K. Huizing, and J. van Wijk, [Squarified Treemaps], (2000).
[36] T. Mann, [A Guide to Library Research Methods] Oxford University Press, New York(1987).