EVALUATING THE EFFECTIVENESS OF PERSONALIZED WEB SEARCH

A PROJECT REPORT

Submitted by

ARTHI R	71206205005
VANITHA L	71206205054
HANISA BEHAM T	71206205301

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY:: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**EVALUATING THE EFFECTIVENESS OF PERSONALIZED WEB SEARCH**" is the bonafide work of **ARTHI R, VANITHA L AND HANISA BEHAM T** who carried out the project work under my supervision

SIGNATURE

Mr.E.A.Vimal M.E,
SUPERVISOR

Department of
Information Technology
Kumaraguru college of technology,
Coimbatore – 641 006.

SIGNATURE

Dr.L.S.Jayashree M.E, PhD.,
HEAD OF THE DEPARTMENT

Department of
Information Technology
Kumaraguru college of Technolog
Coimbatore – 641 006.

The candidates with University Register No **71206205005, 71206205054 & 71206205301 examined** by us in the project viva-vo ce examination held on 19.04.2010

19/4/10

INTERNAL EXAMINER

19.4.10

EXTERNAL EXAMINER

DECLARATION

We,

ARTHI R	**Reg.No: 71206205005**
VANITHA L	**Reg.No: 71206205054**
HANISA BEHAM T	**Reg.No: 71206205301**

Hereby declare that the project entitled " **EVALUATING THE EFFECTIVENESS OF PERSONALIZED WEB SEARCH** " , submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date:

R. A Vanitha L. Hanisa beham

[Arthi R] [Vanitha L] [Hanisa Beham T]

Project Guided by

[Dr.L.S.Jayashree M.E, Ph.D] **[Mr.E.A.Vimal** M.E]

ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushaan Arutselvar Dr.N.Mahalingam B.Sc., F.I.E.,** and Correspondent **Shri.M.Balasubramaniam** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal, **Dr.S.Ramachandran,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy**, BE (HONS), Ph.D., Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-coordinator **Dr.L.S.Jayashree** M.E., Ph.D, Prof, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide **Mr.E.A.Vimal** M.E., Ph.D, Sr.Lecturer, Department of Information Technology for his helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staff of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

ABSTRACT

Although personalized search has been under way for many years and many personalization algorithms have been investigated, it is still unclear whether personalization is consistently effective on different queries for different users and under different search contexts . In our project, we present a large-scale evaluation framework for personalized search based on query logs and then evaluate five personalized search algorithms (including two click-based ones and three topical-interest-based ones). We further propose several features to automatically predict when a query will benefit from a specific personalization algorithm. Experimental results show that using a personalization algorithm for queries selected by our prediction model is better than using it simply for all queries.

LIST OF TABLES

LIST OF TABLES

LIST OF ABBREVIATIONS

LIST OF ABBREVIATIONS

1.	L-Topic	Long-Term User Topical Interest
2.	S-Topic	Short-Term User Topical Interest
3.	LS-Topic	Topical Interest Based Methods
4.	CF	Collaborative Filtering
5.	URL	Uniform Resource Locater

CONTENTS

TABLES OF CONTENTS

1. INTRODUCTION:

One criticism of search engines is that when queries are issued, most return the same results to users. In fact, the vast majority of queries to search engines are short and ambiguous. Different users may have completely different information needs and goals when using precisely the same query. For example, a biologist may query “mouse” to get information about rodents, while programmers may use the same query to find information about computer peripherals.

When such a query is issued, search engines will return a list of documents that mix different topics. It takes time for a user to choose which information he/she wants. On another query of “free mp3 download,” although most users find websites to download free mp3s, their selections can diverge: one may choose the website www.yourmp3.net, while another may prefer the website www.seekasong.com. Personalized search is considered a solution to address these problems, since it can provide different search results based upon the preferences of users. Various personalization strategies, which include, have been proposed. However, they are far from optimal. One problem of current personalized search is that most proposed algorithms are uniformly applied to all users and queries.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 OBJECTIVE:

The main intent of the project is to develop a large scale personalized search evaluation framework based on query logs. In this framework different personalized re-ranking algorithms are simulated and search accuracy is evaluated by real user clicks.

The framework enables us to evaluate personalization approaches on a large scale data set. We implement two click-based personalized search method and three topical-interest based search method

2.2 CURRENT STATUS OF THE PROBLEM

When a query is issued, search engines will return a list of documents that mix different topics. Personalized search is considered a solution to address these problems, since it can provide different search results based upon the preferences of users. However, they are far from optimal.

One problem of current personalized search is that most proposed algorithms are uniformly applied to all users and queries. In a web browser *(Curious Browser)* is developed to record a user's explicit relevance ratings of web pages (relevance feedback) and browsing behavior when viewing a page, such as dwelling time, mouse click, mouse movement and scrolling (implicit feedback). It is shown that the dwelling time on a page, amount of scrolling on a page and the combination of time and scrolling have a strong correlation with explicit relevance ratings, which suggests that implicit feedback may be helpful for inferring user information need. In user click through data is collected as training data to learn a retrieval function, which is used to produce a customized ranking of search results that suits a group of users' preferences.

We argue that queries should not be handled in the same general manner:

1) Personalization may lack effectiveness on some queries, and thus, there is no need of it for these queries.

2) Personalization algorithms have strengths and weaknesses for different queries.

3) The effectiveness of personalization algorithms may vary due to various search contexts.

2.3 PROPOSED SYSTEM AND ITS ADVANTAGES:

In this framework, different personalized reranking algorithms are simulated, and search accuracy is evaluated by real user clicks. The framework enables us to evaluate personalization approaches on a large-scale data set. We implement two click-based personalized search methods and three topical-interest-based methods, evaluate the five approaches in the proposed framework using query logs.

We reveal that personalized Web search has different levels of effectiveness for different queries, users, and search contexts. We experiment with predicting when Web search results can be improved using a personalization method based on a user's long-term interests.

ADVANTAGES:

1) The relevance of documents can be explicitly judged by the participants.

2) The user clicks are highly biased toward documents that are re-ranked at the top of the rank list.

3) Click-through data can be collected at low cost .

4) Using click-through data is closer to real cases in evaluating personalized search than user surveys.

2.4 SYSTEM REQUIREMENTS

2.4.1 HARDWARE REQUIREMENTS

PROCESSOR	:	Pentium IV
HARD DISK	:	40 GB
RAM	:	256 MB
MONITOR	:	15" Color Monitor
KEYBOARD	:	104 keys Standard Keyboard
MOUSE	:	Standard 3 Button Mouse.

2.4.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	:	Windows 9X/NT/XP
LANGUAGE USED	:	JAVA
BACK END	:	SQL Server 2000

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Personalized Search Based on Content Analysis:

User behaviour is collected from an input web log data based on his/her query posted. User is tracked using the IP address from the input web log data.Browsing behaviours of users like time,date,URL is being obtained under web log data.

One approach of personalized search is to filter or rerank search results by checking content similarity between returned web pages and user profiles. User profiles store approximations of user interests. User profiles are either specified by users themselves or are automatically learnt from a user's historical activities. As the vast majority of users are reluctant to provide any explicit feedback on search results and their interests many works on personalized Web search focus on how to automatically learn user preferences without the user being required to directly participate. In terms of how user profiles are built, there are two groups of works: topical categories or keyword lists (bags of words).Several approaches represent user interests by using topical categories.

A user profile is usually structured as a concept/topic hierarchy.User-issued queries and user-selected snippets/documents are categorized into concept hierarchies that are accumulated to generate a user profile. When the user issues a query, each of the returned snippets/documents is also classified. The documents are reranked based upon how well the document categories match user interest profiles. Some other personalized search approaches use lists of keywords to represent user interests.

3.2 Personalized Search based on User Group

In most of the above personalized search strategies, only the information provided by a user himself/herself is exploited to create user profiles. These are also some strategies that incorporate the preferences of a group of users to accomplish personalized search.

In these approaches, search histories of users who have similar interests with a test user are used to refine the search. Collaborative filtering (CF) is a typical group-based personalization method and has been used in personalized search constructed user profiles based on a modified CF algorithm proposed a novel method, to apply personalized Web search by analyzing correlations among users, queries, and web pages in click-through data.

SOFTWARE DESCRIPTION

4. SOFTWARE DESCRIPTION

4.1 JAVA

Java first became popular by being the earliest portable dynamic client-side content for the World-Wide Web in the form of platform-independent Java "applets". In the late 1990's and into the 2000's it has also become very popular on the server side, where an entire set of APIs defines the J2EE.

- Java is an object-oriented programming language developed by Sun Microsystems
  - Java was modeled after C++
  - Java is designed to be small, simple, and portable across platforms and operating systems
  - Java is used to develop executable, distributed applications for delivery to a Java-enabled Web browser or the Java Interpreter. A Java programmer can create the following:
    - **applets:** Programs that are called through an HTML page and run on a Java-enabled browser.
    - **applications:** Standalone Java programs executed independently of a browser. The execution is done using the Java interpreter
  - Java brings interactivity into the Web

There are also packages for developing XML applications, web services, servlets and other web applications, security,date and time calculations and I/O formatting, database(JDBC), and many others.

FEATURES OF JAVA

- **Simple**
    - Java is a safer and cleaner language than C or C++.
    - Memory is managed automatically so the programmer doesn't have to worry about freeing the unused space.
- **Object Oriented**
    - Object-oriented programming is based upon modeling the world in terms of software components called objects.
    - An object consists of data and methods.
    - Methods are operations that can be performed on that data.

- **Distributed**
    - Java is specifically designed to work within a network environment
    - Java has a large library of classes to handle TCP/IP, HTTP, FTP and other networking protocols.
- **Secure**
    - Java was designed with the knowledge that the applications will be transferred through the network
    - Points of entry to protected sectors of memory used by viruses and Trojan horses are impossible to reach using Java.
- **Multithread**
    - Multithreading is the ability for one program to do more than one thing at once.
    - Threads are easy to manage in Java and they take advantage of multiprocessor systems if the operating system does so.
    - Threads in java bring better interactive responsiveness and real time behavior.

- **Portable**
- **Dynamic**
    - Java was designed to adapt to an evolving environment.
    - If you make changes to a parent class in most instances it will not affect the already existing applications. A change of this magnitude in C++ will normally involve recompiling the whole application.
    - In Java you can add new methods and instance variables to libraries without affecting the client applications.
- **Robust**
    - Java unlike C or C++(in some instances) is more careful at handling data types.
    - Java does not support pointers, it uses arrays instead.
    - Java won't allow overwriting of memory and corrupting of data through pointers.

IMPLEMENTATION DETAILS

5. IMPLEMENTATION DETAILS

5.1 MODULES :

5.1.1 Collection of User Behavior on Web Log Data:

One approach of personalized search is to filter or re-rank search results by checking content similarity between returned web pages and user profiles. User profiles store approximations of user interests. User profiles are either specified by users themselves or are automatically learnt from a user's historical activities. As the vast majority of users are reluctant to provide any explicit feedback on search results and their interests, many works on personalized Web search focus on how to automatically learn user preferences without the user being required to directly participate.

In terms of how user profiles are built, there are two groups of works: topical categories or keyword lists (bags of words). Several approaches represent user interests by using topical categories. a user profile is usually structured as a concept/topic hierarchy. User-issued queries and user-selected snippets/documents are categorized into concept hierarchies that are accumulated to generate a user profile. When the user issues a query, each of the returned snippets/documents is also classified. The documents are reranked based upon how well the document categories match user interest profiles.

5.1.2 Computing Rank Scoring Metric:

The rank scoring metric is used to evaluate the effectiveness of CF systems that return an ordered list of recommended items. In the existing system use it to evaluate the retrieval performance of personalized Web search. The expected utility of a ranked list of documents is defined as

$$R_s = \sum_j \frac{\delta(s,j)}{2(j-1)/(\alpha-1)} \quad \text{- - - - - - - - - - - (1)}$$

Where j is the rank of a document, $\delta(s,j)$ is 1 if the jth document is clicked for the test query s and 0 if not clicked. α is a parameter set as five as the authors suggest. The final rank scoring reflects the utility of all test queries:

$$R = 100\frac{\sum_s R_s}{\sum_s R_s^{Max}} \quad \text{- - - - - - - - - - - - (2)}$$

Here, R_s^{Max} is the maximum utility when all clicked documents move to the top of a ranked list. A larger rank scoring value means better retrieval performance.

5.1.3 Implementation Of Personalization Algorithms.

5.1.3.1 Person-Level Re-Ranking:

5.1.3.1.1 Historical Click-Based Algorithm:

We suppose that for a Query q submitted by a user u , the web pages frequently clicked by u in the past are more relevant to U than those seldom clicked by U. Thus ,a personalized score on page p can be computed by

$$S^{P-Click}(q,p,u)=\frac{|Clicks(q,p,u)|}{|Clicks(q,\bullet,u)|+\beta}. \qquad - - - - - - - - - -(3)$$

Here, |Clicks(q,p,u)| is the number of clicks on web page p by user U for query q in the past. |Clicks(q,.,u)| is the total number of clicks for query q by U, and β is a smoothing factor (β=0.5).Actually , |Clicks(q,.,u)| and β are used to normalize |Clicks(q,p,u)|

Disadvantages:

1) It is not applicable for new queries that the user never asked.

2) Another disadvantage of this P-Click is that it may impede the discovery of newly available results because old clicked documents will be re-ranked to the top of the result list.

A feasible solution to this problem is to providing personalized results in a separated list and preserve the original ranking.

Another solution is to randomly push newly document available results toward the top of the list every once in a while or to combine them with previously clicked document.

5.1.3.1.2 **User-Topical-Interest Based Algorithm:**

In this we implemented a personalization method based on long-term user topical interests(we denote this method with L-Topic). When a user submits a query ,each returned pages are mapped to a category vector. Then ,the similarity between the user profile and the page category vector is computed by

$$sim(\mathbf{c}_l(u), \mathbf{c}(p)) = \frac{\mathbf{c}_l(u) \cdot \mathbf{c}(p)}{\|\mathbf{c}_l(u)\| \|\mathbf{c}(p)\|}. \qquad \text{- - - - - - - - - - - (4)}$$

Here,c(p) is the category vector of web page p.User-profile is computed based on his/her past clicked web pages by

$$\mathbf{c}_l(u) = \sum_{p \in P(u)} P(p|u) w(p) \mathbf{c}(p) \qquad \text{- - - - - - - - - - - - - (5)}$$

Here,P(u) is the collection of web pages that were visited by user U in the past. P(p|u) can be thought of as the probability that user U clicks web page p ,

$$P(p|u) = \frac{Clicks(\bullet, p, u)}{Clicks(\bullet, \bullet, u)}. \qquad \text{- - - - - - - - - - - - (6)}$$

Here, Click(.,p,u) is the total number of times that U clicked, and Click(.,.,u) is the number of times that U clicked on web page p, w(p) is an impact weight for page p when generate user profiles.

$$w(p) = \log\frac{|U|}{|U(p)|}. \qquad - - - - - - - - (7)$$

|U| is the number of total user's who have ever visited web pages p.The similarity between user interests and a web page issued to re-rank search results. To reduce the instability of personalization, only the web pages that are similar enough with user interests are reranked. The personalization score of document p is defined as

$$S^{L-Topic}(q,p,u) = \begin{cases} sim(\mathbf{c}_l(u),\mathbf{c}(p)) & \text{if } sim(\mathbf{c}_l(u),\mathbf{c}(p)) \geq t \\ 0 & \text{if } sim(\mathbf{c}_l(u),\mathbf{c}(p)) < t \end{cases} \quad t \in [0,1]. - - - - (8)$$

The short-term user profile is more useful for improving search in an ongoing session. We use clicks on previous queries in an ongoing session to build a short-term user profile and then exploit short-term interests to personalize search. Such an approach is denoted with S-Topic. A short-term user profile cs(u) is computed as

$$\mathbf{c}_s(u) = \frac{1}{|P_s(q)|}\sum_{p\in P_s(q)} \mathbf{c}(p). \qquad - - - - - - - - - - -(9)$$

Ps(q) is the collection of visited pages on previous queries in this session. The similarity between short-term user interests and a web page is defined as

$$sim(\mathbf{c}_s(u),\mathbf{c}(p)) = \frac{\mathbf{c}_s(u)\cdot\mathbf{c}(p)}{\|\mathbf{c}_s(u)\|\,\|\mathbf{c}(p)\|}. - - - - - - - - - -(10)$$

A personalized score of page p by using a short-term profile is computed as

$$S^{S-Topic}(q,p,u) = \begin{cases} sim(\mathbf{c}_s(u),\mathbf{c}(p)) & \text{if } sim(\mathbf{c}_s(u),\mathbf{c}(p)) \geq t \\ 0 & \text{if } sim(\mathbf{c}_s(u),\mathbf{c}(p)) < t \end{cases} \quad t \in [0,1]. - - - (11)$$

We can also fuse the long-term personalized score and the short-term personalized score by a simple linear combination:

$$sim(\mathbf{c}_{ls}(u), \mathbf{c}(p)) = \theta sim(\mathbf{c}_s(u), \mathbf{c}(p)) + (1 - \theta) sim(\mathbf{c}_l(u), \mathbf{c}(p)). \text{------} (12)$$

Thus,

$$S^{LS-Topic}(q, p, u) = \begin{cases} sim(\mathbf{c}_{ls}(u), \mathbf{c}(p)) & \text{if } sim(\mathbf{c}_{ls}(u), \mathbf{c}(p)) \geq t \\ 0 & \text{if } sim(\mathbf{c}_{ls}(u), \mathbf{c}(p)) < t \end{cases} \quad t \in [0, 1]. \text{------} (13)$$

We denote this hybrid approach with LS-Topic. Methods L-Topic, S-Topic, and LS-Topic are generally called **topical-interest-based** methods.

5.1.3.2 **Group Level Re-Ranking:**

We implement a K-Nearest Neighbor CF algorithm as a representative of group-based personalization. Due to sparse data, we find that applying traditional CF methods on Web search is inadequate. Instead, we compute user similarity based on long-term user profiles. The K-Nearest neighbors are obtained based on the user similarity:

$$s_u(u_a) = \{u_s \mid rank(sim(u_a, u_s)) \leq K\} \text{------------} (14)$$

Then, we use the historical clicks made by all similar users in a group to rerank the search results:

$$S^{G-Click}(q,p,u)=\frac{\sum_{u_s \in S_u(u)} sim(u_s,u)\,|clicks(q,p,u_s)|}{\beta+\sum_{u_s \in S_u(u)} |clicks(q,\bullet,u_s)|} \quad ------- (15)$$

We denote this group-level approach with G-Click.

5.1.4 **Performance Comparison :**

The above graph shows that the improvement of rank scoring metrics for the algorithms such as P-Click, L-Topic, S-Topic, LS-Topic and the group level reranking.

The group level reranking algorithm shows the better rank scoring value.

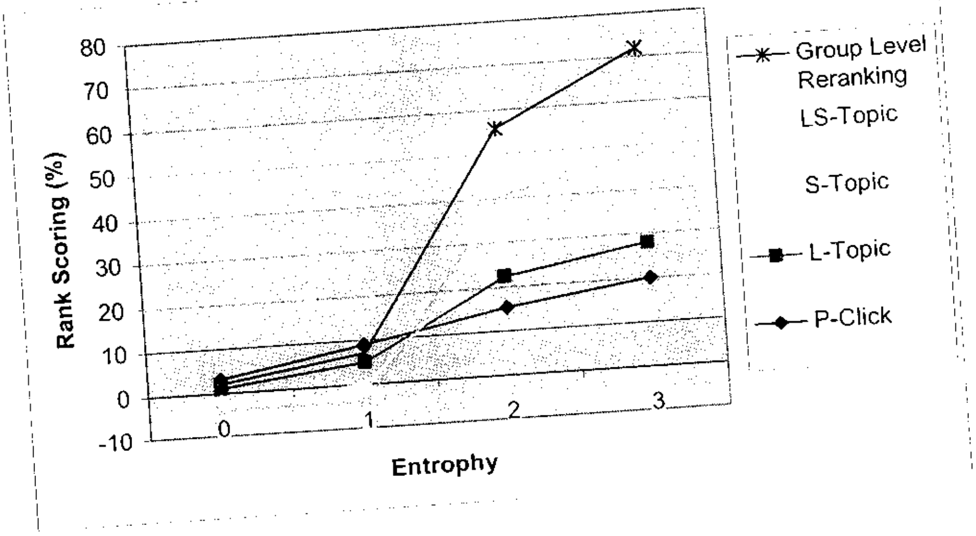


FIG : 1.1 PERFORMANCE COMPARISON

5.3 SYSTEM FLOW DIAGRAM :

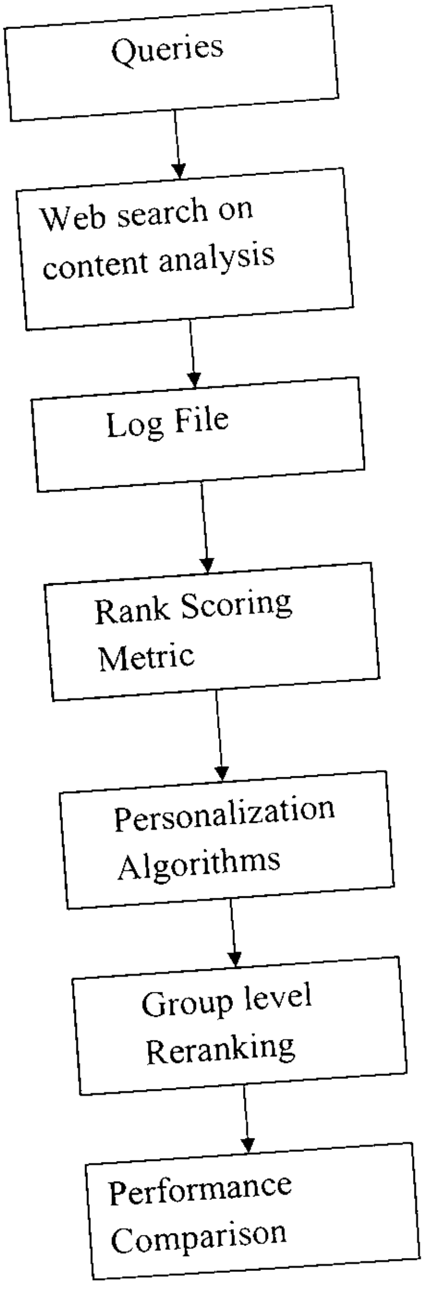


FIG : 1·2

DATABASE DESIGN

6.Database Design

1.Table Name:Log File:

Field Name	Field Type	Comments
TransID	number	Users transaction no
ClientIP	Varchar	Users IP name
Status	Number	Port Number
Date And Time	DateTime	Timespan of the User
Method	Varchar	Get and post methods
WebAddress	Varchar	URL's visited by a user
Query	Varchar	Query posted by the user
Clicks	Int	No of clicks clicked by a user on a web page
URL	Varchar	Specifies the url

TABLE NO:1

2.Table Name : Score

FIELD NAME	FIELD TYPE	COMMENTS
Query	Varchar	Query posted by a user
ClientIP	Varchar	User's IP name
WebAddress	Varchar	Links visited by the user
Score	Float	Contains the values for each query

TABLE NO : 2

3.Table Name:Group Click:

FIELD NAME	FIELD TYPE	COMMENTS
Query	Varchar	Query posted by a user
ClientIP	Varchar	User's IP name
WebAddress	Varchar	Links visited by the user
GroupClickValue	Numeric	Contains the reranked values

TABLE NO : 3

PERFORMANCE EVALUATION

7. PERFORMANCE EVALUATION

Testing is a critical element of software quality and assurance and represents the ultimate review of specification design and coding. It is a vital activity that has to be enforced in the development of the system. This could be done in parallel during all the phases of system development . The feedback received from these texts can be used for further enhancement of the system under consideration. The testing phase conducts test using the Software Requirements Specifications as a reference and with the goal to see whether system satisfies the specified requirements.

Standard procedures have been followed in testing our system. Test cases are generated for each screen. These test cases will cover every possibility which could result in both positive and negative results. These test plans are maintained for any further testing done on the system.

The main types of test carried out are:

- Unit testing
- Integration testing
- System testing
- Validation testing

7.1 UNIT TESTING

Each and every module is tested separately to check if its intended functionality is met. Some unit testings are performed are:

1.Validating the user

2. Loading database and its applications

7.2 INTEGRATION TESTING

It is the testing performed to detect errors on inter connection between modules. The application should connect to respective databases. The application events should be backed up in the log file for future recovery.

7.3 SYSTEM TESTING

The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

7.4 VALIDATION TESTING

This test is done to check for the validity of the entered input. The user inputs to the corresponding application input fields are verified before updating in the database.

CONCLUSION

8. CONCLUSION

We developed an evaluation framework based on real query logs to enable large-scale evaluation of personalized search. We use query logs to evaluate five personalized search algorithms. In the experiments, click-based personalization algorithms worked well. Although the algorithms work only for repeated queries, they are simple and stable. We suggest that search engines take advantage of user histories in search if privacy issues do not prohibit it. The topical-interest- based personalized search algorithms implemented were not as stable as the click-based ones under our framework. They could improve search accuracy for some queries, but they harmed performance for more queries. As these methods were not optimal, we will continue our evaluation work on improved versions in the future.

FUTURE ENHANCEMENTS

9. FUTURE ENHANCEMENTS

Experimental results showed that using the L-Topic algorithm for queries selected by our prediction model would achieve better overall performance than using it simply for all queries. We will try to build prediction models for other algorithms in future work. We found that no personalization algorithms can outperform others for all queries. Different methods have different strengths and weaknesses. A promising direction we will explore in the future is to automatically predict which algorithm should be used for given a query and/or to combine the strengths of different personalization methods

APPENDICES

10. APPENDICES

10.1 SOURCE CODE:

Main Program:

Personalized Web Search:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class PersonalizedWebSearch extends JFrame implements ActionListener
{
        JMenuBar mbMenu;
        JMenu meProcess,meexit,meAlg,mePLC,mips1;
        JMenuItem milfu,mips,miExit,LTopic,STopic,LSTopic,GLR;
        JDesktopPane desktop;
        public Object object[]=new Object[20];

        public PersonalizedWebSearch()
        {
                super("Personalized WebSearch");
                mbMenu = new JMenuBar();
                meProcess = new JMenu("Preprocessing");
                meAlg = new JMenu("Personalization Algorithms");
                mePLC =new JMenu("Person-Level Reranking");
                meexit = new JMenu("Exit");
                miExit = new JMenuItem("Exit");
                milfu = new JMenuItem("Log File Updation");
                mips = new JMenuItem("Historical Click-Based Algorithm");
                mips1 = new JMenu("User-Topical-Interest-Based Algorithms");
                LTopic = new JMenuItem("LTopic");
                STopic = new JMenuItem("STopic");
                LSTopic = new JMenuItem("LSTopic");
                GLR = new JMenuItem("Group Level ReRanking");
                mbMenu.add(meProcess);
                mbMenu.add(meexit);
                meProcess.add(milfu);
                mePLC.add(mips);
                mips1.add(LTopic);
                mips1.add(STopic);
                mips1.add(LSTopic);
                mePLC.add(mips1);
                mePLC.add(GLR);
```

```
            meAlg.add(mePLC);
            meProcess.add(meAlg);
            meexit.add(miExit);
            setJMenuBar(mbMenu);
            desktop=new JDesktopPane();
            setContentPane(desktop);
            object[0]=milfu;
object[1]=miExit;
object[2]=mips;
            miExit.addActionListener(this);
            milfu.addActionListener(this);
            mips.addActionListener(this);
            LTopic.addActionListener(this);
            STopic.addActionListener(this);
            LSTopic.addActionListener(this);
            GLR.addActionListener(this);
            Dimension ss=Toolkit.getDefaultToolkit().getScreenSize();
            setSize(ss.width,ss.height);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
   }
   public void actionPerformed(ActionEvent ae)
   {
            Object source=ae.getSource();
            if(source.equals(milfu))
            {
                    LFU objLfu = new LFU(object);
                    display(objLfu);

            }
            if(source.equals(mips))
            {
                    PSFrame objLfu = new PSFrame(this);
                    display(objLfu);

            }
            if(source.equals(LTopic))
            {
                    LTopicFrame LTF=new LTopicFrame(this);
                    display(LTF);

            }
            if(source.equals(STopic))
            {
                    STopicFrame STF=new STopicFrame(this);
                    display(STF);

            }
            if(source.equals(LSTopic))
            {
                LSTopicFrame LSTF=new LSTopicFrame(this);
                display(LSTF);
            }
            if(source.equals(GLR))
            {
                GroupLevelRank GLRF=new GroupLevelRank(this);
                display(GLRF);
            }
            if(source.equals(miExit))
            {
                System.exit(0);
            }
    }

    void display(JInternalFrame obj)
    {
        new CenterFrame(obj);
        obj.setVisible(true);
        desktop.add(obj);
        try
        {
            obj.setSelected(true);
        }
        catch(java.beans.PropertyVetoException e2){}
    }

    public static void main(String args[])
    {
        PersonalizedWebSearch qm = new PersonalizedWebSearch();
    }
}
```

LS-TOPIC:

```
import java.awt.*;
import java.sql.*;
import java.io.*;
import java.awt.event.*;
import java.text.*;
import javax.swing.*;
import java.util.*;

public class LSTopicFrame extends JInternalFrame
{
        private JButton clac;
        private JButton view;
        private JPanel contentPane;
```

```
    database_conn db=null;
    ResultSet rs,rs1,rs2,rs3,rs4,rs5;
    PersonalizedWebSearch desktop=null;
    Random R=new Random();
    NumberFormat formatter=new DecimalFormat("#0.00000");
    public LSTopicFrame(PersonalizedWebSearch desktop)
    {
            //super("STopic");
            super("LSTopic",true,true,true,true);
            this.desktop=desktop;
            initializeComponent();
            db=new database_conn();
    }
    private void initializeComponent()
    {
            clac = new JButton();
            view = new JButton();
            contentPane = (JPanel)this.getContentPane();
            clac.setText("Calculate");
            clac.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e)
                    {
                            clac_actionPerformed(e);
                    }

            });
            view.setText("View Result");
            view.setEnabled(false);
            view.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e)
                    {
                            view_actionPerformed(e);
                    }

            });
            contentPane.setLayout(null);
            addComponent(contentPane, clac, 51,71,102,34);
            addComponent(contentPane, view, 178,70,102,34);
            this.setSize(new Dimension(332, 214));
    }
    private void addComponent(Container container,Component c,int x,int y,int width,int
height)
    {
            c.setBounds(x,y,width,height);
            container.add(c);
    }
```

```
        private void clac_actionPerformed(ActionEvent ae)
        {
                try
                {
                        db.stat.execute("if object_id('LSTopicRes') is not null drop table
LSTopicRes");
                        db.stat.execute("create table LSTopicRes(query varchar(100),clientip
varchar(20),webaddress varchar(200),value float)");
                        db.stat.execute("if object_id('LSTopic') is not null drop table LSTopic");
                        db.stat.execute("select L.query,L.clientip,L.webaddress,L.value as
Lvalue,S.value as Svalue into LSTopic from LTopic L,STopic S where L.query = S.query and
L.clientip = S.clientip and L.webaddress = S.webaddress");
                        double th=0.0;
                        while(true)
                        {
                                String t=JOptionPane.showInputDialog(null,"Enter LTopic
Values(between 0-1)","0.0");
                                th=Double.parseDouble(t);
                                if(th > 0.0 && th <= 1.0)
                                        break;
                        }
                        double theta = 0.0;
                        rs = db.stat.executeQuery("select (avg(Lvalue) + avg(Svalue)) /2 from
LSTopic");
                        if(rs.next())
                                theta=rs.getDouble(1);

                        rs = db.stat.executeQuery("select * from LSTopic");
                        if(rs.next())
                        {
                                do
                                {
                                        String s1 = rs.getString(1);
                                        String s2 = rs.getString(2);
                                        String s3 = rs.getString(3);
                                        double cl = rs.getDouble(4);
                                        double cs = rs.getDouble(5);
                                        double cls = (theta * cs) + ((1-theta) * cl);
                                        //System.out.println("cls  : "+cls);
                                        if(cls >= th)
                                        {
                                                db.stat3.execute("insert into LSTopicRes
values('"+s1+"','"+s2+"','"+s3+"',"+cls+")");
                                        }

                                }
                        while(rs.next());
                    }
                    JOptionPane.showMessageDialog(null,"STopic Values
Updated","INFORMATION",JOptionPane.INFORMATION_MESSAGE);
                    view.setEnabled(true);

            }
            catch(Exception e)
            {
                    System.out.println("Error While Calculating The Personalized Score");
                    e.printStackTrace();
            }
    }

    private void view_actionPerformed(ActionEvent ae)
    {
            try
            {
                    JInternalFrame table=new JInternalFrame("LSTopic ",true,true,true,true);
                    //JFrame table=new JFrame("STopic");
                    table.setLayout(new BorderLayout());
                    String header[]={"Query","Client IP","URL","LSTopicValue"};
                    Object data[][]=null;
                    rs=db.stat.executeQuery("select * from LSTopicRes");
                    if(rs.next())
                    {
                            ResultSetMetaData rsm=rs.getMetaData();
                            rs.last();
                            int r=rs.getRow();
                            int c=rsm.getColumnCount();
                            data=new Object[r][c];
                            rs.first();
                            int i=0;
                            do
                            {
                                    for(int j=0;j<data[i].length;j++)
                                    {
                                            data[i][j]=rs.getObject(j+1);
                                    }
                                    i++;
                            }
                            while(rs.next());

                            JTable result=new JTable(data,header);
                            JScrollPane sp=new JScrollPane(result);
                            table.add(sp,BorderLayout.CENTER);
```

```
                                //table.setVisible(true);
                                Dimension ss=new
Dimension(Toolkit.getDefaultToolkit().getScreenSize());
                                sp.setBounds(0,0,ss.width-50,ss.height-80);
                                table.setSize(ss.width-500,ss.height-500);
                                desktop.display(table);

                        }
                        else
                                JOptionPane.showMessageDialog(null,"Similar Pages Are Not
Found","ERROR",JOptionPane.ERROR_MESSAGE);
                }
                catch(Exception e)
                {
                        System.out.println("Can't View The Table");
                        e.printStackTrace();
                }

        }
```

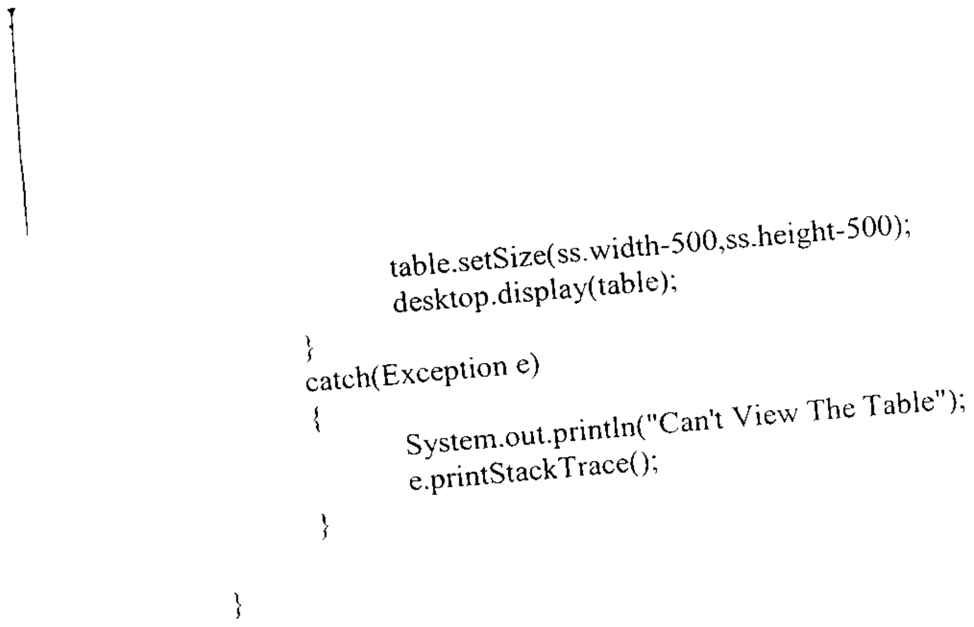
Personalization Score Calculation:

```
import java.awt.*;
import java.sql.*;
import java.io.*;
import java.awt.event.*;
import java.text.*;
import javax.swing.*;

public class PSFrame extends JInternalFrame
{
        private JButton clac;
        private JButton view;
        private JPanel contentPane;
        database_conn db=null;
        ResultSet rs,rs1,rs2,rs3,rs4,rs5;
        PersonalizedWebSearch desktop=null;
        NumberFormat formatter=new DecimalFormat("#0.00000");
        public PSFrame(PersonalizedWebSearch desktop)
        {
                super("PSFrame",true,true,true,true);
                this.desktop=desktop;
                initializeComponent();
                db=new database_conn();
        }
        private void initializeComponent()
```

```
    {
        clac = new JButton();
        view = new JButton();
        contentPane = (JPanel)this.getContentPane();
        clac.setText("Calculate");
        clac.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                clac_actionPerformed(e);
            }

        });
        view.setText("View Result");
        view.setEnabled(false);
        view.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                view_actionPerformed(e);
            }

        });
        contentPane.setLayout(null);
        addComponent(contentPane, clac, 51,71,102,34);
        addComponent(contentPane, view, 178,70,102,34);
        this.setSize(new Dimension(332, 214));
    }
    private void addComponent(Container container,Component c,int x,int y,int width,int
height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }
    private void clac_actionPerformed(ActionEvent ae)
    {
        try
        {
            db.stat.execute("if object_id('score') is not null drop table score");
            db.stat.execute("create table score(query varchar(100),clientip
varchar(20),webaddress varchar(200),score float)");
            rs=db.stat.executeQuery("select distinct query from tbl_ImportedData");
            if(rs.next())
            {
                do
                {
                    String str=rs.getString(1);
```

```
                              rs1=db.stat1.executeQuery("select distinct clientip from
tbl_ImportedData");
                              if(rs1.next())
                              {
                                  do
                                  {
                                      String str1=rs1.getString(1);
                                      rs2=db.stat2.executeQuery("select distinct
webaddress from tbl_ImportedData");
                                      if(rs2.next())
                                      {
                                          do
                                          {
                                              String str2=rs2.getString(1);

    rs3=db.stat3.executeQuery("select count(click) from tbl_ImportedData where
query='"+str+"' and clientip='"+str1+"' and webaddress='"+str2+"' and click=1");
                                              int s1=0;
                                              if(rs3.next())
                                                  s1=rs3.getInt(1);


    rs3=db.stat3.executeQuery("select count(click) from tbl_ImportedData where
query='"+str+"' and clientip='"+str1+"' and click=1");
                                              int s2=0;
                                              if(rs3.next())
                                                  s2=rs3.getInt(1);

                                              double s=s1/(s2+0.5);

                                              String st=formatter.format(s);
                                              s=Double.parseDouble(st);
                                              db.stat3.execute("insert into
score values('"+str+"','"+str1+"','"+str2+"',"+s+")");

                                          }
                                          while(rs2.next());
                                      }
                                  }
                                  while(rs1.next());
                              }
                          }
                          while(rs.next());
                          JOptionPane.showMessageDialog(null,"Personalized Score
Updated","INFORMATION",JOptionPane.INFORMATION_MESSAGE);
                      }
```

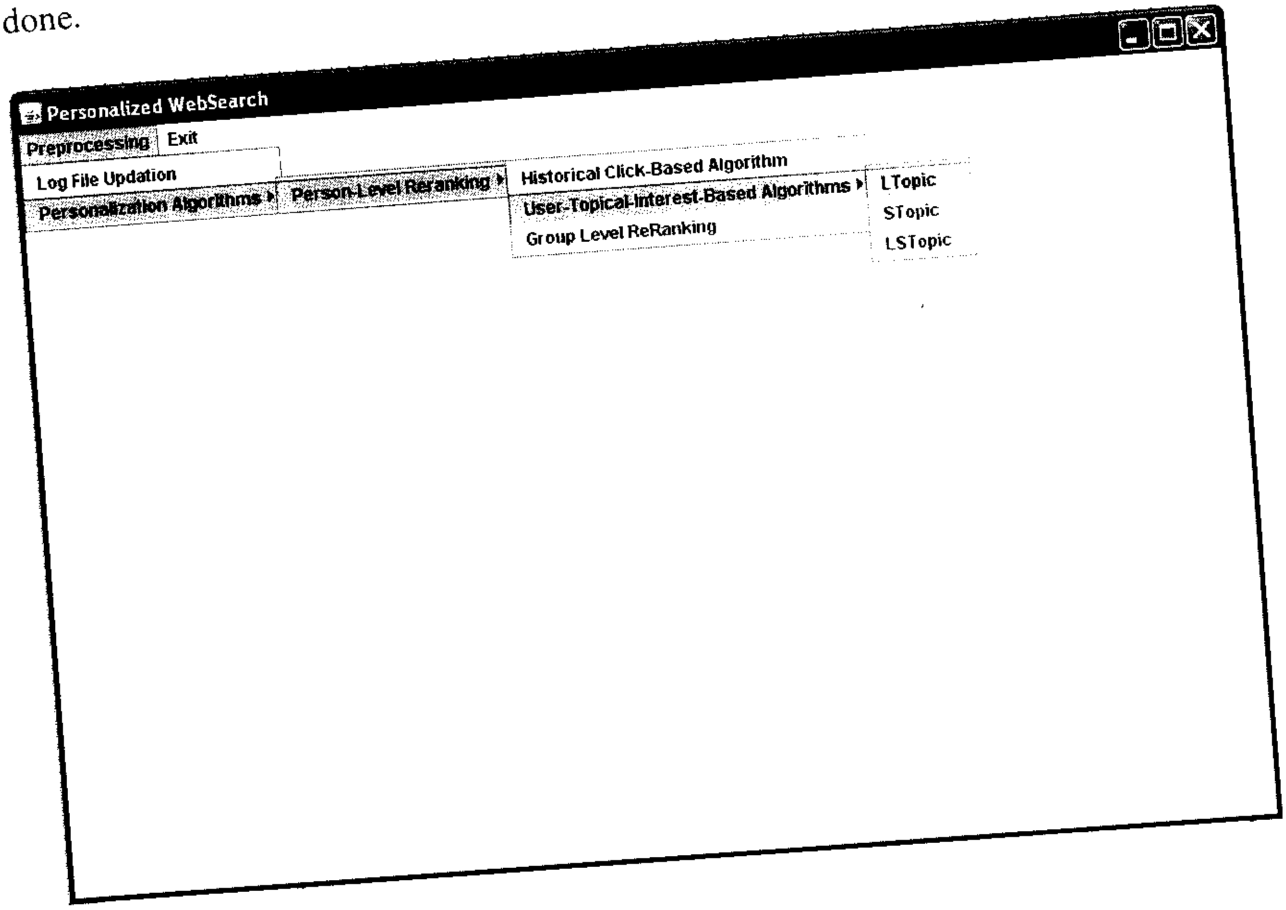
```
                        view.setEnabled(true);

                }
                catch(Exception e)
                {
                        System.out.println("Error While Calculating The Personalized Score");
                        e.printStackTrace();
                }
        }

        private void view_actionPerformed(ActionEvent ae)
        {
                try
                {
                        JInternalFrame table=new JInternalFrame("Personalized Score
",true,true,true,true);
                        table.setLayout(new BorderLayout());
                        String header[]={"Query","Client IP","URL","Score"};
                        Object data[][]=null;
                        rs=db.stat.executeQuery("select * from score ");
                        if(rs.next())
                        {
                                ResultSetMetaData rsm=rs.getMetaData();
                                rs.last();
                                int r=rs.getRow();
                                int c=rsm.getColumnCount();
                                data=new Object[r][c];
                                rs.first();
                                int i=0;
                                do
                                {
                                        for(int j=0;j<data[i].length;j++)
                                        {
                                                data[i][j]=rs.getObject(j+1);
                                        }
                                        i++;
                                }
                                while(rs.next());
                        }
                        JTable result=new JTable(data,header);
                        JScrollPane sp=new JScrollPane(result);
                        table.add(sp,BorderLayout.CENTER);
                        //table.setVisible(true);
                        Dimension ss=new
Dimension(Toolkit.getDefaultToolkit().getScreenSize());
                        sp.setBounds(0,0,ss.width-50,ss.height-80);
```



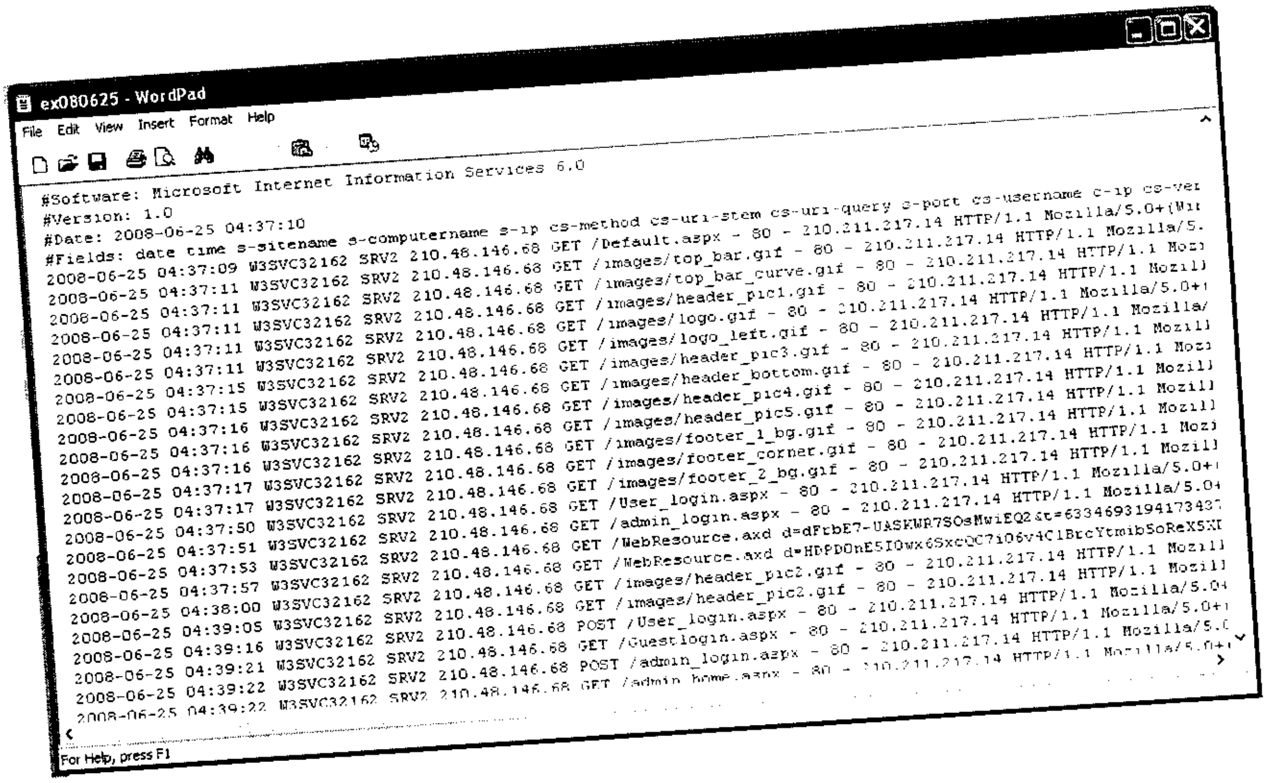
```
            table.setSize(ss.width-500,ss.height-500);
            desktop.display(table);

        }
        catch(Exception e)
        {
            System.out.println("Can't View The Table");
            e.printStackTrace();

        }

    }
```

10.2 SCREEN SHOTS:

The main menu of the project provides information about various operations done.



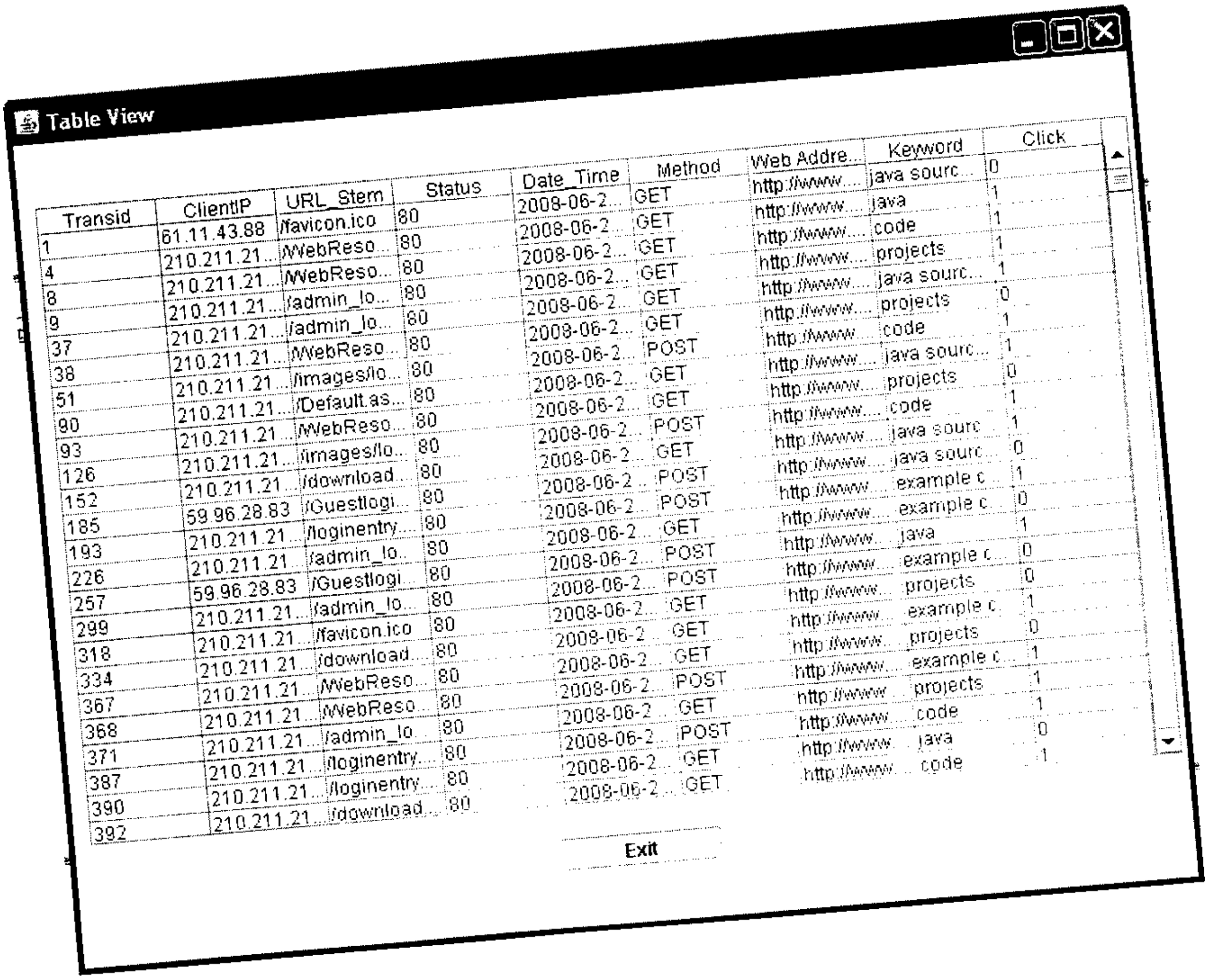
10.2.1 SAMPLE LOG FILE:



```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2008-06-25 04:37:10
#Fields: date time s-sitename s-computername s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs-ver
2008-06-25 04:37:09 W3SVC32162 SRV2 210.48.146.68 GET /Default.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+(Wi
2008-06-25 04:37:11 W3SVC32162 SRV2 210.48.146.68 GET /images/top_bar.gif - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.
2008-06-25 04:37:11 W3SVC32162 SRV2 210.48.146.68 GET /images/top_bar_curve.gif - 80 - 210.211.217.14 HTTP/1.1 Mozi
2008-06-25 04:37:11 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic1.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:11 W3SVC32162 SRV2 210.48.146.68 GET /images/logo.gif - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
2008-06-25 04:37:11 W3SVC32162 SRV2 210.48.146.68 GET /images/logo_left.gif - 80 - 210.211.217.14 HTTP/1.1 Mozilla/
2008-06-25 04:37:15 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic3.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:15 W3SVC32162 SRV2 210.48.146.68 GET /images/header_bottom.gif - 80 - 210.211.217.14 HTTP/1.1 Mozi
2008-06-25 04:37:16 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic4.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:16 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic5.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:16 W3SVC32162 SRV2 210.48.146.68 GET /images/footer_1_bg.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:17 W3SVC32162 SRV2 210.48.146.68 GET /images/footer_corner.gif - 80 - 210.211.217.14 HTTP/1.1 Mozi
2008-06-25 04:37:17 W3SVC32162 SRV2 210.48.146.68 GET /images/footer_2_bg.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:37:50 W3SVC32162 SRV2 210.48.146.68 GET /User_login.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
2008-06-25 04:37:51 W3SVC32162 SRV2 210.48.146.68 GET /admin_login.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
2008-06-25 04:37:53 W3SVC32162 SRV2 210.48.146.68 GET /WebResource.axd d=dFrbE7-UASKWR7SOsMwiEQ2&t=633469319417343
2008-06-25 04:37:57 W3SVC32162 SRV2 210.48.146.68 GET /WebResource.axd d=HDPDOnE5IOwx6SxcQC7i06v4C1BrcYtmib5oReX5XI
2008-06-25 04:38:00 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic2.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:39:05 W3SVC32162 SRV2 210.48.146.68 GET /images/header_pic2.gif - 80 - 210.211.217.14 HTTP/1.1 Mozill
2008-06-25 04:39:16 W3SVC32162 SRV2 210.48.146.68 POST /User_login.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
2008-06-25 04:39:21 W3SVC32162 SRV2 210.48.146.68 GET /Guestlogin.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
2008-06-25 04:39:22 W3SVC32162 SRV2 210.48.146.68 POST /admin_login.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0
2008-06-25 04:39:22 W3SVC32162 SRV2 210.48.146.68 GET /admin_home.aspx - 80 - 210.211.217.14 HTTP/1.1 Mozilla/5.0+
```

10.2.2 LOG FILE UPDATION:

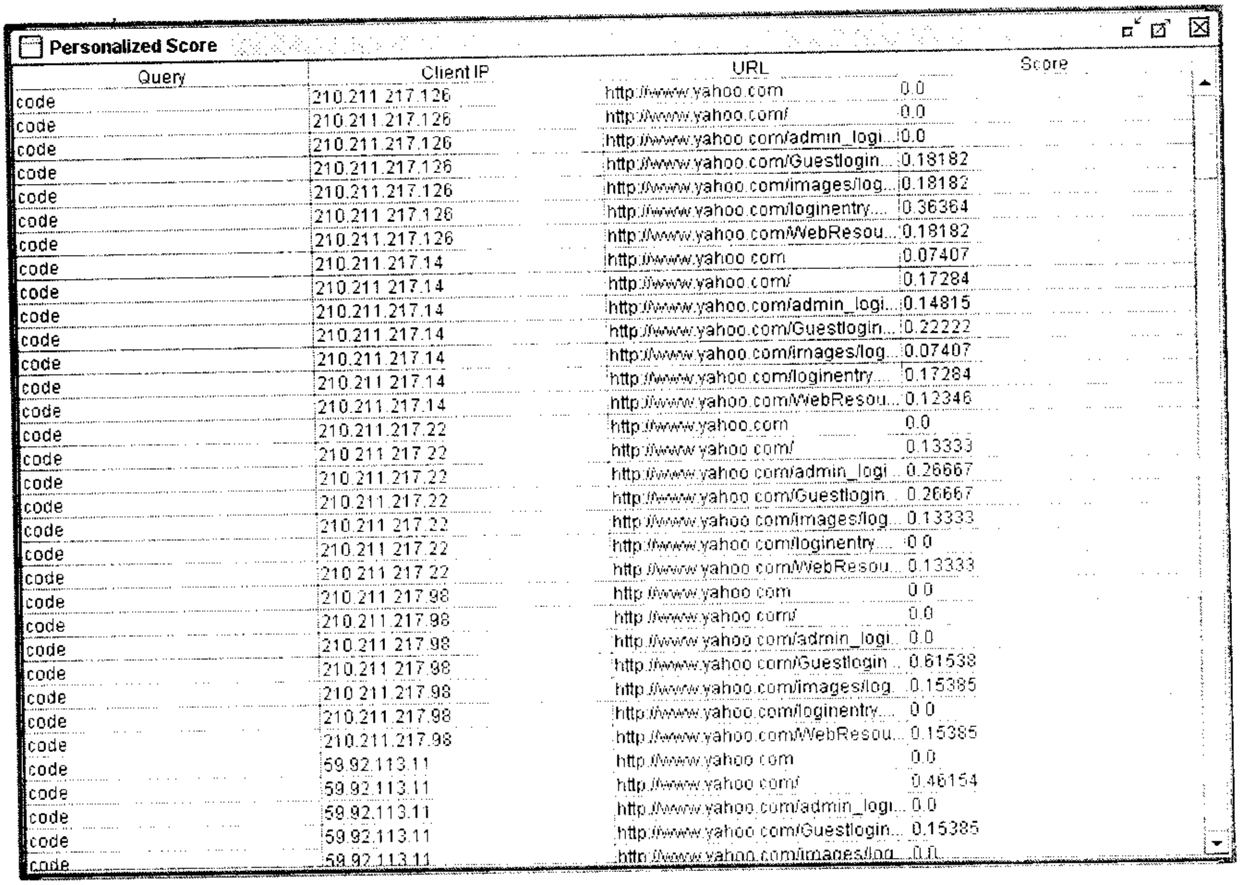
The following screen shot shows the updation of log file from the sample web log data.



Transid	ClientIP	URL_Stem	Status	Date_Time	Method	Web Addre...	Keyword	Click
1	61.11.43.88	/favicon.ico	80	2008-06-2...	GET	http://www....	java sourc...	0
4	210.211.21...	/WebReso...	80	2008-06-2...	GET	http://www....	java	1
8	210.211.21...	/WebReso...	80	2008-06-2...	GET	http://www....	code	1
9	210.211.21...	/admin_lo...	80	2008-06-2...	GET	http://www....	projects	1
37	210.211.21...	/admin_lo...	80	2008-06-2...	GET	http://www....	java sourc...	1
38	210.211.21...	/WebReso...	80	2008-06-2...	GET	http://www....	projects	0
51	210.211.21...	/images/lo...	80	2008-06-2...	POST	http://www....	code	1
90	210.211.21...	/Default.as...	80	2008-06-2...	GET	http://www....	java sourc...	1
93	210.211.21...	/WebReso...	80	2008-06-2...	GET	http://www....	projects	0
126	210.211.21...	/images/lo...	80	2008-06-2...	POST	http://www....	code	1
152	210.211.21...	/download...	80	2008-06-2...	GET	http://www....	java sourc...	1
185	59.96.28.83	/Guestlogi...	80	2008-06-2...	POST	http://www....	java sourc...	0
193	210.211.21...	/loginentry...	80	2008-06-2...	POST	http://www....	example c...	1
226	210.211.21...	/admin_lo...	80	2008-06-2...	GET	http://www....	example c...	0
257	59.96.28.83	/Guestlogi...	80	2008-06-2...	POST	http://www....	java	1
299	210.211.21...	/admin_lo...	80	2008-06-2...	POST	http://www....	example c...	0
318	210.211.21...	/favicon.ico	80	2008-06-2...	GET	http://www....	projects	0
334	210.211.21...	/download...	80	2008-06-2...	GET	http://www....	example c...	1
367	210.211.21...	/WebReso...	80	2008-06-2...	GET	http://www....	projects	0
368	210.211.21...	/WebReso...	80	2008-06-2...	POST	http://www....	example c...	1
371	210.211.21...	/admin_lo...	80	2008-06-2...	GET	http://www....	projects	1
387	210.211.21...	/loginentry...	80	2008-06-2...	POST	http://www....	code	1
390	210.211.21...	/loginentry...	80	2008-06-2...	GET	http://www....	java	0
392	210.211.21...	/download...	80	2008-06-2...	GET	http://www....	code	1

10.2.3 CALCULATING PERSONALIZATION SCORE:

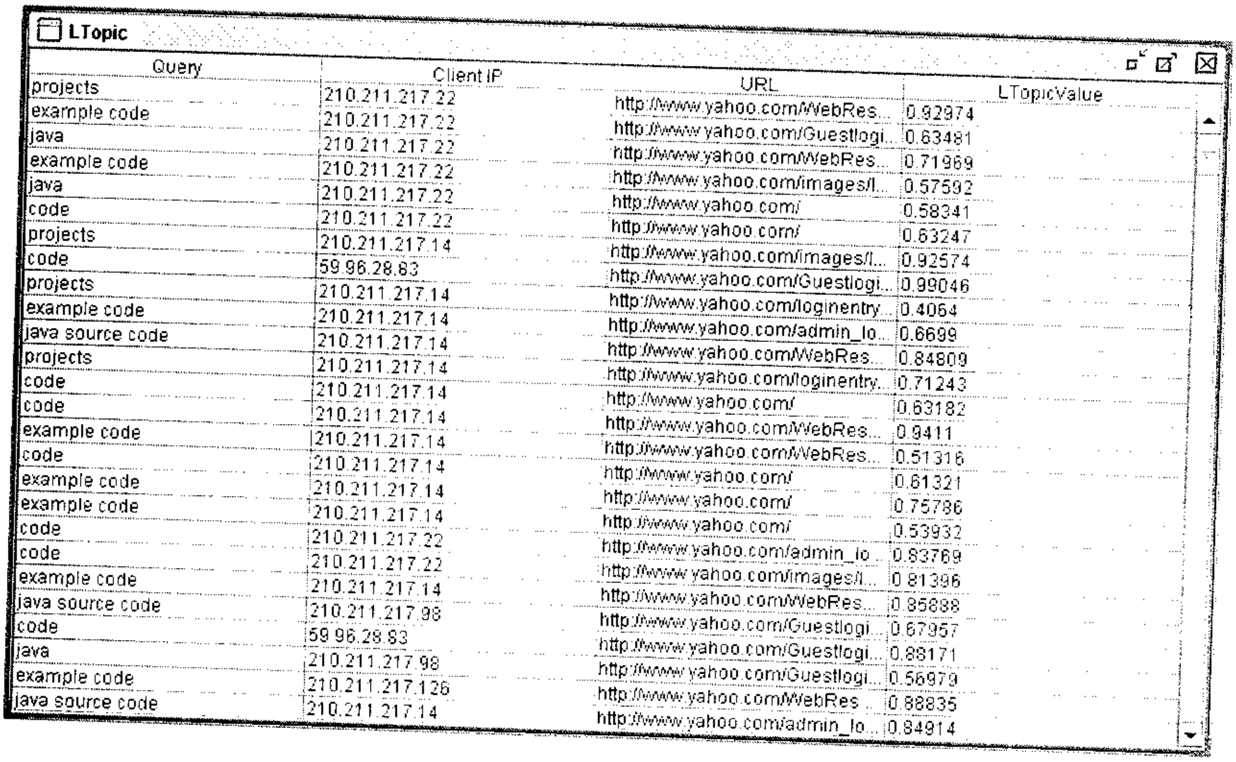
The following screen shot shows the personalization score for the query.



Query	Client IP	URL	Score
code	210.211.217.126	http://www.yahoo.com	0.0
code	210.211.217.126	http://www.yahoo.com/	0.0
code	210.211.217.126	http://www.yahoo.com/admin_logi...	0.0
code	210.211.217.126	http://www.yahoo.com/Guestlogin...	0.18182
code	210.211.217.126	http://www.yahoo.com/images/log...	0.18182
code	210.211.217.126	http://www.yahoo.com/loginentry....	0.36364
code	210.211.217.126	http://www.yahoo.com/WebResou...	0.18182
code	210.211.217.14	http://www.yahoo.com	0.07407
code	210.211.217.14	http://www.yahoo.com/	0.17284
code	210.211.217.14	http://www.yahoo.com/admin_logi...	0.14815
code	210.211.217.14	http://www.yahoo.com/Guestlogin...	0.22222
code	210.211.217.14	http://www.yahoo.com/images/log...	0.07407
code	210.211.217.14	http://www.yahoo.com/loginentry....	0.17284
code	210.211.217.14	http://www.yahoo.com/WebResou...	0.12346
code	210.211.217.22	http://www.yahoo.com	0.0
code	210.211.217.22	http://www.yahoo.com/	0.13333
code	210.211.217.22	http://www.yahoo.com/admin_logi...	0.26667
code	210.211.217.22	http://www.yahoo.com/Guestlogin...	0.26667
code	210.211.217.22	http://www.yahoo.com/images/log...	0.13333
code	210.211.217.22	http://www.yahoo.com/loginentry....	0.0
code	210.211.217.22	http://www.yahoo.com/WebResou...	0.13333
code	210.211.217.98	http://www.yahoo.com	0.0
code	210.211.217.98	http://www.yahoo.com/	0.0
code	210.211.217.98	http://www.yahoo.com/admin_logi...	0.0
code	210.211.217.98	http://www.yahoo.com/Guestlogin...	0.61538
code	210.211.217.98	http://www.yahoo.com/images/log...	0.15385
code	210.211.217.98	http://www.yahoo.com/loginentry....	0.0
code	210.211.217.98	http://www.yahoo.com/WebResou...	0.15385
code	59.92.113.11	http://www.yahoo.com	0.0
code	59.92.113.11	http://www.yahoo.com/	0.46154
code	59.92.113.11	http://www.yahoo.com/admin_logi...	0.0
code	59.92.113.11	http://www.yahoo.com/Guestlogin...	0.15385
code	59.92.113.11	http://www.yahoo.com/images/log...	0.0

10.2.4 L-TOPIC :

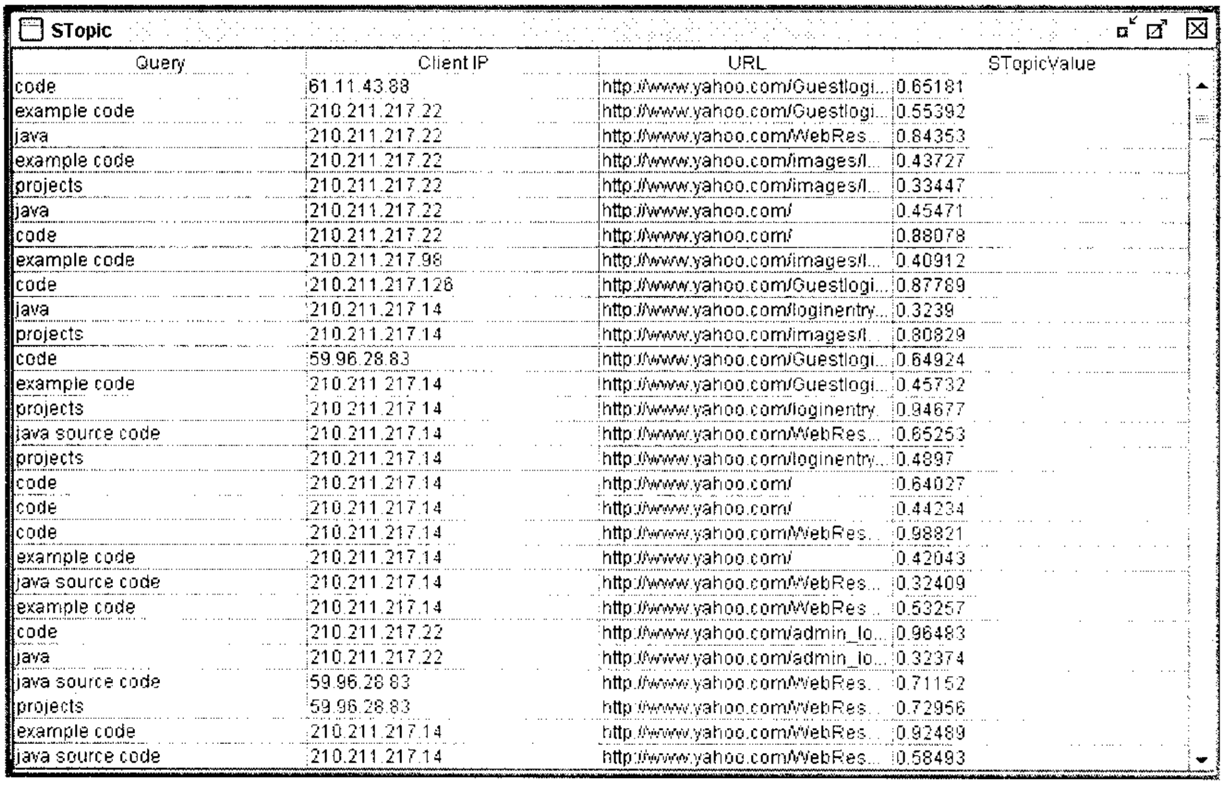
The following screen shots shows the L-Topic values that are calculated from the personalization algorithms.



Query	Client IP	URL	LTopicValue
projects	210.211.217.22	http://www.yahoo.com/WebRes...	0.92974
example code	210.211.217.22	http://www.yahoo.com/Guestlogi...	0.63481
java	210.211.217.22	http://www.yahoo.com/WebRes...	0.71969
example code	210.211.217.22	http://www.yahoo.com/images/l...	0.57592
java	210.211.217.22	http://www.yahoo.com/	0.58341
code	210.211.217.22	http://www.yahoo.com/	0.63247
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.92574
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.99046
projects	210.211.217.14	http://www.yahoo.com/loginentry...	0.4064
example code	210.211.217.14	http://www.yahoo.com/admin_lo...	0.6699
java source code	210.211.217.14	http://www.yahoo.com/WebRes...	0.84809
projects	210.211.217.14	http://www.yahoo.com/loginentry...	0.71243
code	210.211.217.14	http://www.yahoo.com/	0.63182
code	210.211.217.14	http://www.yahoo.com/WebRes...	0.9411
example code	210.211.217.14	http://www.yahoo.com/WebRes...	0.51316
code	210.211.217.14	http://www.yahoo.com/	0.61321
example code	210.211.217.14	http://www.yahoo.com/	0.75786
example code	210.211.217.14	http://www.yahoo.com/	0.53932
code	210.211.217.22	http://www.yahoo.com/admin_lo...	0.83769
code	210.211.217.22	http://www.yahoo.com/images/l...	0.81396
example code	210.211.217.14	http://www.yahoo.com/WebRes...	0.85888
java source code	210.211.217.98	http://www.yahoo.com/Guestlogi...	0.67957
code	59 96.28.83	http://www.yahoo.com/Guestlogi...	0.88171
java	210.211.217.98	http://www.yahoo.com/Guestlogi...	0.56979
example code	210.211.217.126	http://www.yahoo.com/WebRes...	0.88835
java source code	210.211.217.14	http://www.yahoo.com/admin_lo...	0.84914

10.2.5 S-TOPIC :

The following screen shots shows the S-Topic values that are calculated from the personalization algorithms.

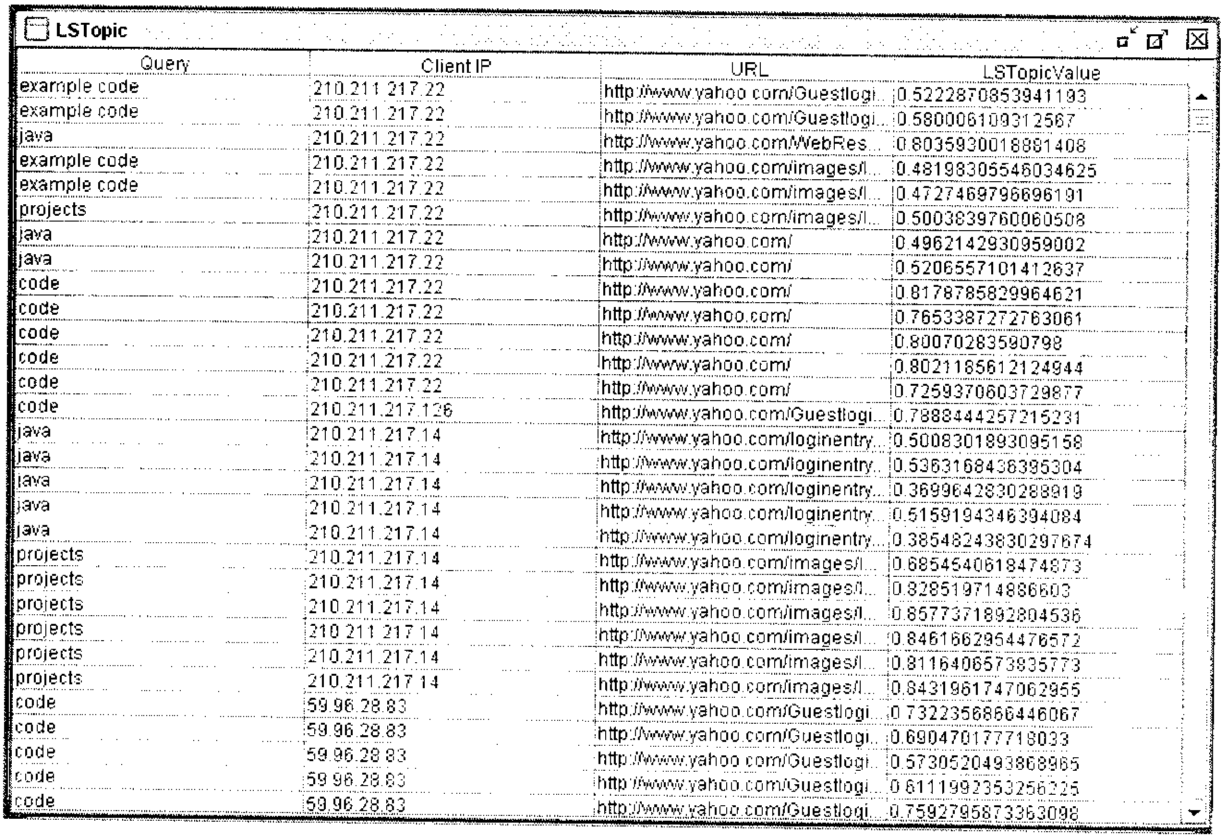


STopic

Query	Client IP	URL	STopicValue
code	61.11.43.88	http://www.yahoo.com/Guestlogi...	0.65181
example code	210.211.217.22	http://www.yahoo.com/Guestlogi...	0.55392
java	210.211.217.22	http://www.yahoo.com/WebRes...	0.84353
example code	210.211.217.22	http://www.yahoo.com/images/l...	0.43727
projects	210.211.217.22	http://www.yahoo.com/images/l...	0.33447
java	210.211.217.22	http://www.yahoo.com/	0.45471
code	210.211.217.22	http://www.yahoo.com/	0.88078
example code	210.211.217.98	http://www.yahoo.com/images/l...	0.40912
code	210.211.217.126	http://www.yahoo.com/Guestlogi...	0.87789
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.3239
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.80829
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.64924
example code	210.211.217.14	http://www.yahoo.com/Guestlogi...	0.45732
projects	210.211.217.14	http://www.yahoo.com/loginentry...	0.94677
java source code	210.211.217.14	http://www.yahoo.com/WebRes...	0.65253
projects	210.211.217.14	http://www.yahoo.com/loginentry...	0.4897
code	210.211.217.14	http://www.yahoo.com/	0.64027
code	210.211.217.14	http://www.yahoo.com/	0.44234
code	210.211.217.14	http://www.yahoo.com/WebRes...	0.98821
example code	210.211.217.14	http://www.yahoo.com/	0.42043
java source code	210.211.217.14	http://www.yahoo.com/WebRes...	0.32409
example code	210.211.217.14	http://www.yahoo.com/WebRes...	0.53257
code	210.211.217.22	http://www.yahoo.com/admin_lo...	0.96483
java	210.211.217.22	http://www.yahoo.com/admin_lo...	0.32374
java source code	59.96.28.83	http://www.yahoo.com/WebRes...	0.71152
projects	59.96.28.83	http://www.yahoo.com/WebRes...	0.72956
example code	210.211.217.14	http://www.yahoo.com/WebRes...	0.92489
java source code	210.211.217.14	http://www.yahoo.com/WebRes...	0.58493

10.2.6 LS-TOPIC:

The following screen shots shows the LS-Topic values that are calculated from the personalization algorithms

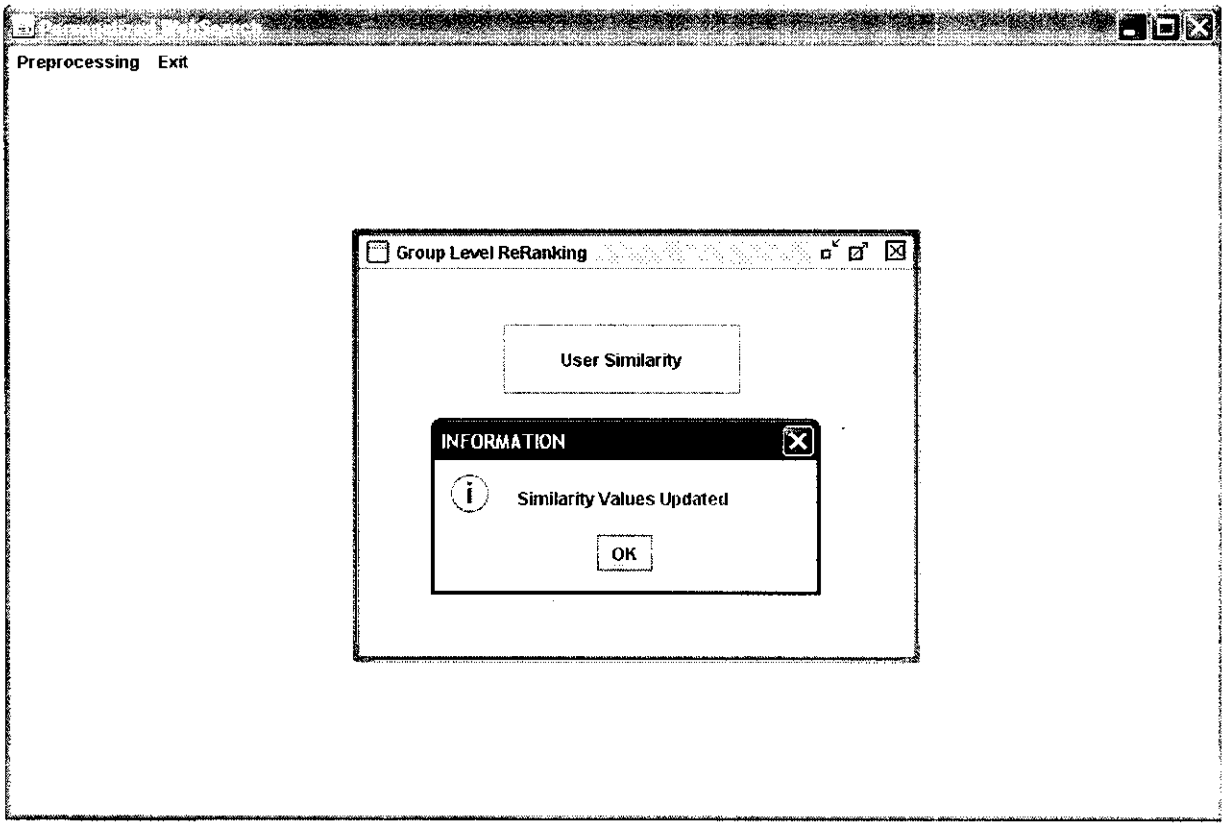


LSTopic

Query	Client IP	URL	LSTopicValue
example code	210.211.217.22	http://www.yahoo.com/Guestlogi...	0.5222870853941193
example code	210.211.217.22	http://www.yahoo.com/Guestlogi...	0.580006109312567
java	210.211.217.22	http://www.yahoo.com/WebRes...	0.8035930018881408
example code	210.211.217.22	http://www.yahoo.com/images/l...	0.48198305546034625
example code	210.211.217.22	http://www.yahoo.com/images/l...	0.4727469796696191
projects	210.211.217.22	http://www.yahoo.com/images/l...	0.5003839760060508
java	210.211.217.22	http://www.yahoo.com/	0.4962142930959002
java	210.211.217.22	http://www.yahoo.com/	0.5206557101412637
code	210.211.217.22	http://www.yahoo.com/	0.8178785829964621
code	210.211.217.22	http://www.yahoo.com/	0.7653387272763061
code	210.211.217.22	http://www.yahoo.com/	0.80070283590798
code	210.211.217.22	http://www.yahoo.com/	0.8021185612124944
code	210.211.217.22	http://www.yahoo.com/	0.7259370603729877
code	210.211.217.126	http://www.yahoo.com/Guestlogi...	0.7888444257215231
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.5008301893095158
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.5363168438395304
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.3699642830288919
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.5159194346394084
java	210.211.217.14	http://www.yahoo.com/loginentry...	0.38548243830297674
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.6854540618474873
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.828519714886603
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.8577371892804536
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.8461662954476572
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.8116406573935773
projects	210.211.217.14	http://www.yahoo.com/images/l...	0.8431961747062955
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.7322356866446067
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.690470177718033
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.5730520493868965
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.6111992353256225
code	59.96.28.83	http://www.yahoo.com/Guestlogi...	0.7592795873363098

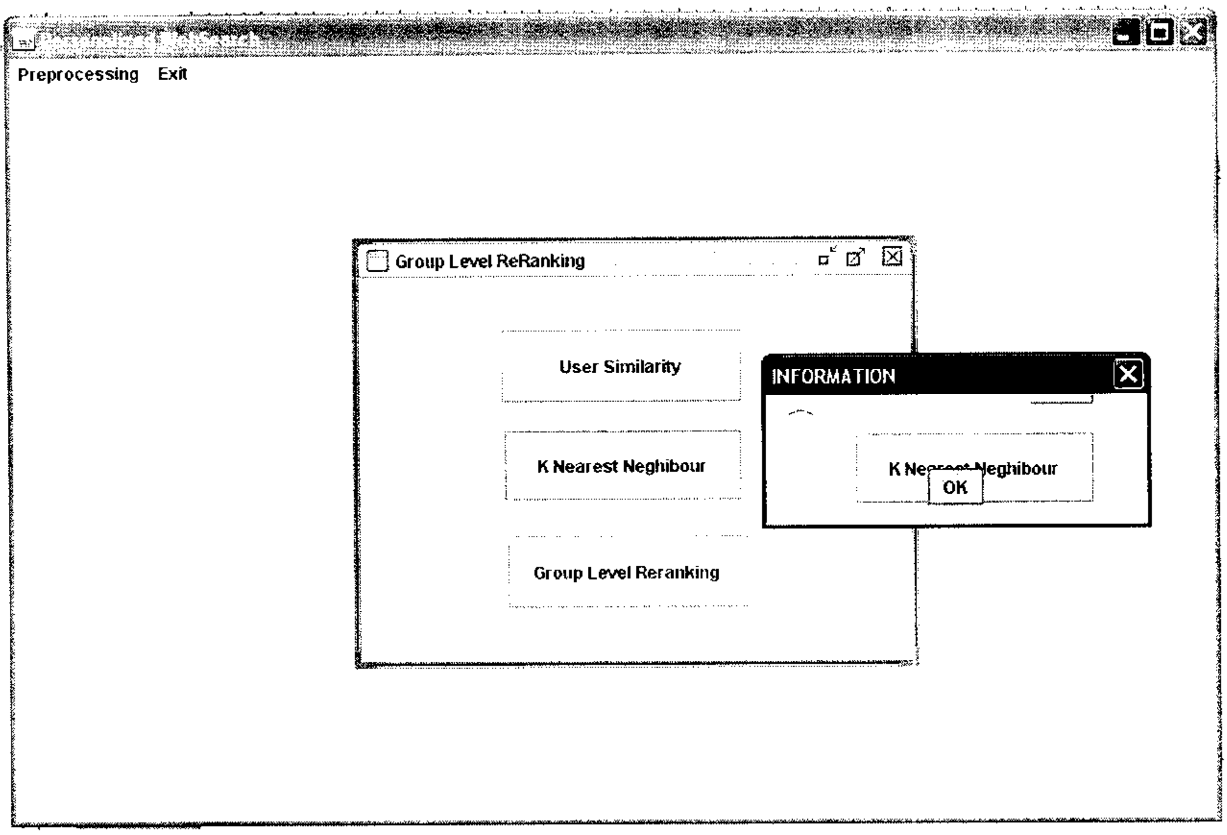
10.2.7 Group Level Re-Ranking:

The following screen shots shows the similarity of users and calculating the K-Nearest neighbours and G-Click values:

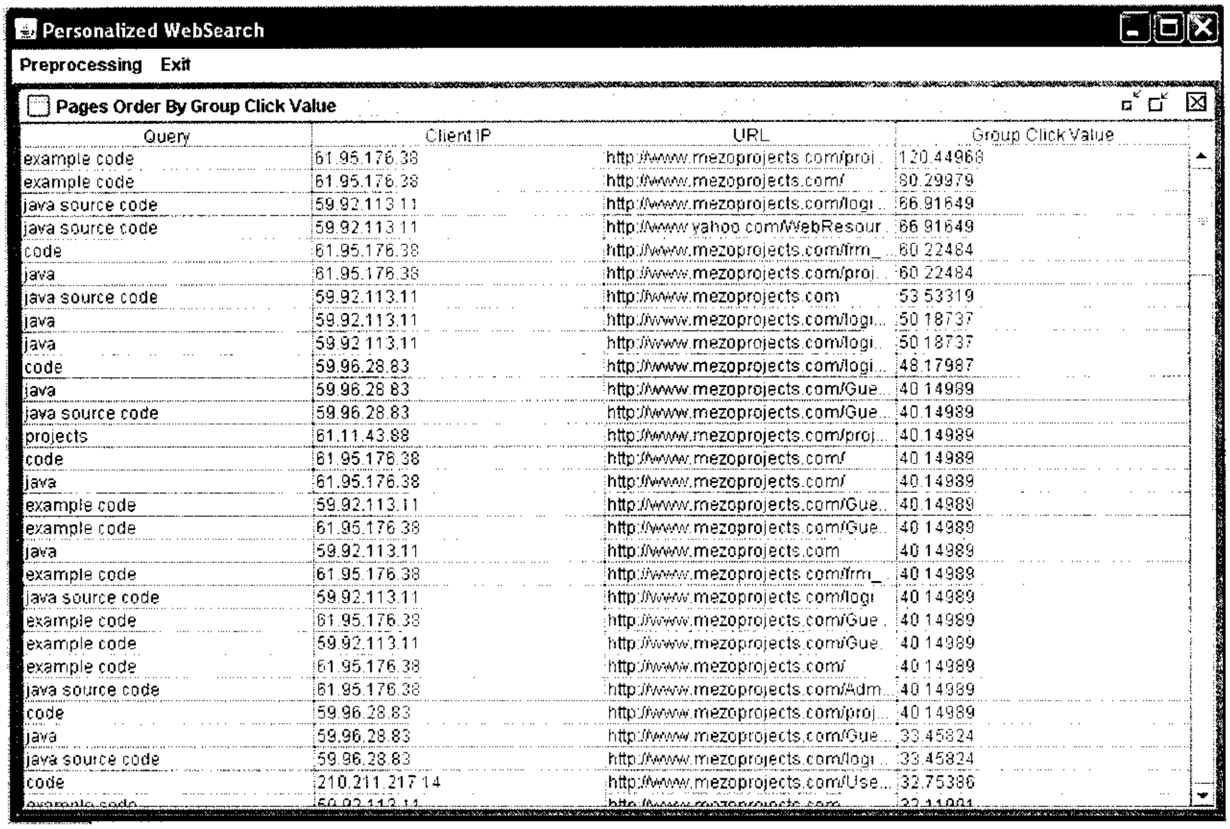


Similarity Values Updated

Group level re-ranking:



Group Click Value:



Pages Order By Group Click Value

Query	Client IP	URL	Group Click Value
example code	61.95.176.38	http://www.mezoprojects.com/proj...	120.44968
example code	61.95.176.38	http://www.mezoprojects.com/	80.29979
java source code	59.92.113.11	http://www.mezoprojects.com/logi...	66.91649
java source code	59.92.113.11	http://www.yahoo.com/WebResour...	66.91649
code	61.95.176.38	http://www.mezoprojects.com/frm_...	60.22484
java	61.95.176.38	http://www.mezoprojects.com/proj...	60.22484
java source code	59.92.113.11	http://www.mezoprojects.com	53.53319
java	59.92.113.11	http://www.mezoprojects.com/logi...	50.18737
java	59.92.113.11	http://www.mezoprojects.com/logi...	50.18737
code	59.96.28.83	http://www.mezoprojects.com/logi...	48.17987
java	59.96.28.83	http://www.mezoprojects.com/Gue...	40.14989
java source code	59.96.28.83	http://www.mezoprojects.com/Gue...	40.14989
projects	61.11.43.88	http://www.mezoprojects.com/proj...	40.14989
code	61.95.176.38	http://www.mezoprojects.com/	40.14989
java	61.95.176.38	http://www.mezoprojects.com/	40.14989
example code	59.92.113.11	http://www.mezoprojects.com/Gue...	40.14989
example code	61.95.176.38	http://www.mezoprojects.com/Gue...	40.14989
java	59.92.113.11	http://www.mezoprojects.com	40.14989
example code	61.95.176.38	http://www.mezoprojects.com/frm_...	40.14989
java source code	59.92.113.11	http://www.mezoprojects.com/logi	40.14989
example code	61.95.176.38	http://www.mezoprojects.com/Gue...	40.14989
example code	59.92.113.11	http://www.mezoprojects.com/Gue...	40.14989
example code	61.95.176.38	http://www.mezoprojects.com/	40.14989
java source code	61.95.176.38	http://www.mezoprojects.com/Adm...	40.14989
code	59.96.28.83	http://www.mezoprojects.com/proj...	40.14989
java	59.96.28.83	http://www.mezoprojects.com/Gue...	33.45824
java source code	59.96.28.83	http://www.mezoprojects.com/logi...	33.45824
code	210.211.217.14	http://www.mezoprojects.com/Use...	32.75386
example code	59.92.113.11	http://www.mezoprojects.com	32.11991

REFERENCES

11. REFERENCES

[1] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz, "Analysis of a Very Large Web Search Engine Query Log," ACM SIGIR Forum, vol. 33, no. 1, pp. 6-12, 1999.

[2] 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, vol. 36, no. 2, pp. 207-227, 2000.

[3] R. Krovetz and W.B. Croft, "Lexical Ambiguity and Information Retrieval," Information Systems, vol. 10, no. 2, pp. 115-141, 1992.

[4] S. Cronen-Townsend and W.B. Croft, "Quantifying Query Ambiguity," Proc. Second Int'l Conf. Human Language Technology Research (HLT '02), pp. 94-98, 2002.

[5] X. Shen, B. Tan, and C. Zhai, "Implicit User Modeling for Personalized Search," Proc. ACM Int'l Conf. Information and Knowledge Management (CIKM '05), pp. 824-831, 2005.