

ABSTRACT

Search engine companies collect the “database of intentions”, the histories of their users search queries. These search logs are a gold mine for researchers. Search engine companies, however, are wary of publishing search logs in order not to disclose sensitive information.

In this paper we analyze algorithms for publishing frequent keywords, queries and clicks of a search log. We first show how methods that achieve variants of k -anonymity are vulnerable to active attacks. We then demonstrate that the stronger guarantee ensured by ϵ -differential privacy unfortunately does not provide any utility for this problem. We then propose a novel algorithm ZEALOUS and show how to set its parameters to achieve probabilistic privacy. We also contrast our analysis of ZEALOUS with an analysis that achieves indistinguishability.

Our paper concludes with a large experimental study using real applications where we compare ZEALOUS and previous work that achieves k -anonymity in search log publishing. Our results show that ZEALOUS yields comparable utility to k -anonymity while at the same time achieving much stronger privacy guarantees.

CHAPTER 1

INTRODUCTION

Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men. Search engines play a crucial role in the navigation through the vastness of the Web. Today's search engines do not just collect and index web pages, they also collect and mine information about their users. They store the **Queries, Clicks, IP-addresses**, and other information about the interactions with users in what is called a **Search log**. Search logs contain valuable information that search engines use to tailor their services better to their users' needs. They enable the discovery of trends, patterns, and anomalies in the search behavior of users, and they can be used in the development and testing of new algorithms to improve search performance and quality.

Scientists all around the world would like to tap this gold mine for their own research; search engine companies, however, do not release them because they contain sensitive information about their users, for example searches for diseases, lifestyle choices, personal tastes, and political affiliations. The only release of a search log happened in 2007 by AOL, and it went into the annals of tech history as one of the great debacles in the search industry.

Content may change prior to publication and clicks of a search log. These methods vary in the guarantee of disclosure limitations they provide and in the amount of useful information they retain. We first describe two negative results. We show that existing proposals to achieve k -anonymity in search logs are insufficient in the light of attackers who can actively influence the search log. We then turn to differential privacy, a much stronger privacy guarantee; however, we show that it is impossible to achieve good utility with differential privacy. We then describe Algorithm ZEALOUS, developed independently by and us with the goal to achieve relaxations of differential privacy showed how to set the parameters of ZEALOUS to guarantee in distinguish ability and we here offer a new analysis that shows how to set the parameters of ZEALOUS to guarantee probabilistic differential privacy ,a much stronger privacy guarantee as our analytical comparison shows [1]. Our paper concludes with an extensive experimental evaluation, where we compare the utility of various algorithms that guarantee anonymity or privacy in search log publishing.

Our evaluation includes applications that use search logs for improving both search experience and search performance, and our results show that ZEALOUS output is sufficient for these applications while achieving strong formal privacy guarantees. We believe that the results of this research enable search engine companies to make their search log available to researchers without disclosing their users' sensitive information. Search engine companies can apply our algorithm to generate statistics that are probabilistic differentially private while retaining good utility for the two applications we have tested. Beyond publishing search logs we believe that our findings are of interest when publishing frequent item sets, as ZEALOUS protects privacy against much stronger attackers than those considered in existing work on privacy preserving publishing of frequent items/item sets.

1.2 BASIC CONCEPTS

1.2.1 SEARCH LOGS

Search engines such as Bing, Google, or Yahoo log interactions with their users. When a user submits a query and clicks on one or more results, a new entry is added to the search log. Without loss of generality, assume that a search log has the following schema: **USER-ID; QUERY; TIME; CLICKS**, where a **USER-ID** identifies a user, a **QUERY** is a set of keywords, and **CLICKS** is a list of url that the user clicked on.

The user-id can be determined in various ways; for example, through cookies, IP addresses, or user accounts. A user history or search history consists of all search entries from a single user. Such a history is usually partitioned into sessions containing similar queries [1]. A query pair consists of two subsequent queries from the same user within the same session.

We say that a user history contains a keyword k if there exists a search log entry such that k is a keyword in the history of a single user. We will refer to search logs that only differ in the search history of a single user as neighboring search logs.

Similar to the variants of k -anonymity, we could also define variants of differential privacy by looking at neighboring search logs that differ only in the content of one session, one query or one keyword. However, we chose to focus on the strongest definition in which an attacker learns roughly the same about a user even if that user's whole search history was omitted.

1.2.2 INFORMATION STORAGE AND RETRIVAL

Information storage and retrieval is the systematic process of collecting and cataloging data so that they can be located and displayed on request. Computers and data processing techniques have made possible the high-speed, selective retrieval of large amounts of information for government, commercial, and academic purposes. There are several basic types of information-storage-and-retrieval systems. Document-retrieval systems store entire documents, which are usually retrieved by title or by key words associated with the document.

Database systems store the information as a series of discrete records that are, in turn, divided into discrete fields (e.g., name, address, and phone number); records can be searched and retrieved on the basis of the content of the fields (e.g., all people who have a particular telephone area code). The data are stored within the computer, either in main storage or auxiliary storage, for ready access [1]. Reference-retrieval systems store references to documents rather than the documents themselves.

Such systems, in response to a search request, provide the titles of relevant documents and frequently their physical locations. Such systems are efficient when large amounts of different types of printed data must be stored. They have proven extremely effective in libraries, where material is constantly changing.

1.2.3 INFORMATION TECHNOLOGY AND SYSTEM

Information Technology (IT) is the application of computers and telecommunications equipment to store, retrieve, transmit and manipulate data, often in the context of a business or other enterprise. Document-retrieval systems store entire documents, which are usually retrieved by title or by key words associated with the document. The term is commonly used as a synonym for computers and computer networks, but it also encompasses other information distribution technologies such as television and telephones. Several industries are associated with information technology, such as computer hardware, software, electronics, semiconductors, internet, telecom equipment, e-commerce and computer services.

1.2.4 DATABASE MANAGEMENT

Database Management System (DBMS) is a collection of interrelated data and a set of programs to access those data. A collection of programs enables you to store, modify, and extract information from a database. There are many different types of DBMS's, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications: computerized library systems, automated teller machines, flight reservation systems, computerized parts inventory systems. From a technical standpoint, DBMSs can differ widely [2]. The terms relational, network, flat, and hierarchical all refer to the way a DBMS organizes information internally. The information from a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report.

1.2.5 SECURITY AND INTEGRITY

Integrity, in terms of data and network security, is the assurance that information can only be accessed or modified by those authorized to do so. Measures taken to ensure integrity include controlling the physical environment of networked terminals and servers, restricting access to data, and maintaining rigorous authentication practices.

Data integrity can also be threatened by environmental hazards, such as heat, dust, and electrical surges. Practices followed to protect data integrity in the physical environment include: making servers accessible only to network administrators, keeping transmission media (such as cables and connectors) covered and protected to ensure that they cannot be tapped. Security is based on user accounts. Every process gets copy of token .System checks token to determine if access allowed or denied [2]. Uses a subject model to ensure access security. A classified site goes to extraordinary lengths to keep things physically tight. Among the issues to be considered :Unauthorized access Mechanism assuring only authorized individuals see classified materials, Malicious modification or destruction, Accidental introduction of inconsistency.

CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

2.2 KNOWLEDGE AND DATA ENGINEERING

An individual is typically referred by numerous name aliases on the web. Accurate identification of aliases of a given person name is useful in various web related tasks such as information retrieval, sentiment analysis, personal name disambiguation, and relation extraction. We propose a method to extract aliases of a given personal name from the web. Given a personal name, the proposed method first extracts a set of candidate aliases. Second, we rank the extracted candidates according to the likelihood of a candidate being a correct alias of the given name [3]. We propose a novel, automatically extracted lexical pattern-based approach to efficiently extract a large set of candidate aliases from snippets retrieved from a web search engine.

The information published in this journal is designed to inform researchers, developers, managers, strategic planners, users, and others interested in state-of-the-art and state-of-the-practice activities in the knowledge and data engineering area. We are interested in well-defined theoretical results and empirical studies that have potential impact on the acquisition, management, storage, and graceful degeneration of knowledge and data, as well as in provision of knowledge and data services.

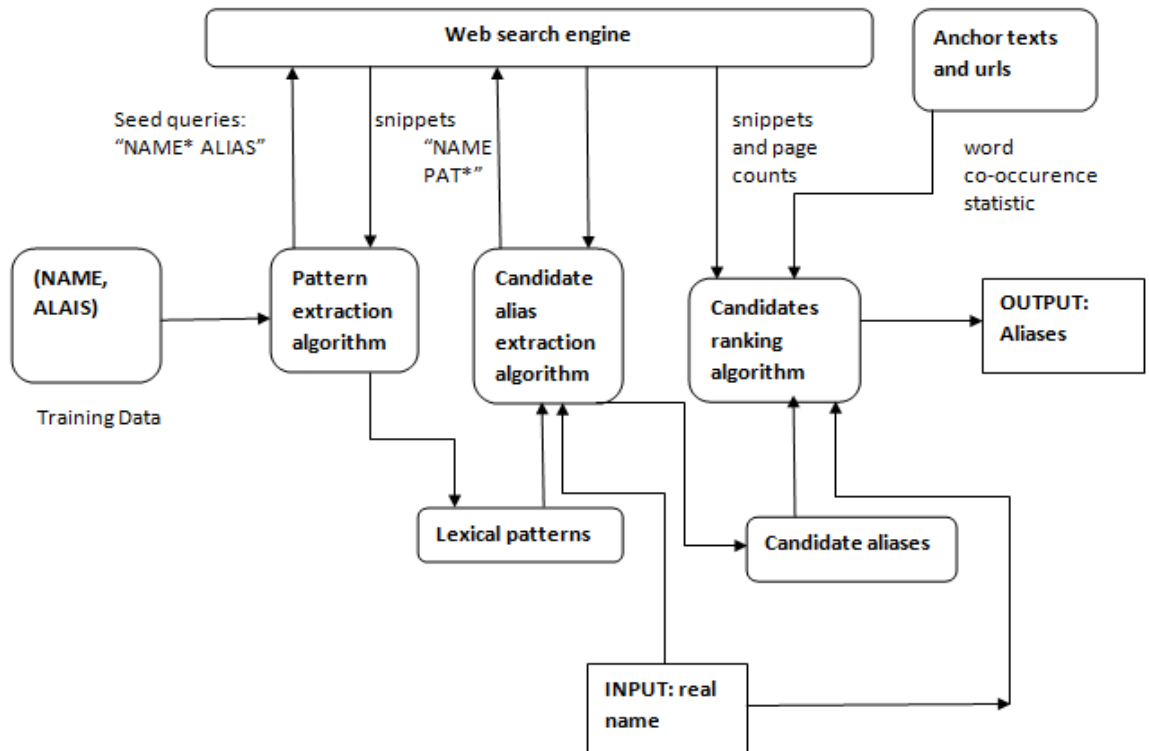


Figure 1: Web Search Engine

We welcome treatments of the role of knowledge and data in the development and use of information systems and in the simplification of software and hardware development and maintenance. Since the journal is archival, it is assumed that the ideas presented are important and have been well analyzed or empirically validated, and are of value to the knowledge and data engineering research community. Specific topics are not limited to: (a) knowledge discovery and data mining, (b) data modeling and management, (c) underlying computational platforms for knowledge and data engineering tools, techniques and systems, and (d) emerging knowledge and data engineering applications.

2.3 BENEFITS OF DATA ENGINEERING

Database Systems and Knowledgebase Systems share many common principles. *Data & Knowledge Engineering* stimulates the exchange of ideas and interaction between these two related fields of interest. *DKE* reaches a world-wide audience of researchers, designers, managers and users [3]. The major aim of the journal is to identify, investigate and analyze the underlying principles in the design and effective use of these systems. *DKE* achieves this

aim by publishing original research results, technical advances and news items concerning data engineering, knowledge engineering.

2.4 BENEFITS OF KNOWLEDGE ENGINEERING

- **Representation and Manipulation of Data & Knowledge:** Conceptual data models. Knowledge representation techniques [3].Data/knowledge manipulation languages and techniques.
- **Architectures of Database, Expert, or Knowledge-Based Systems:** New architectures for database / knowledge base / expert systems, design and implementation techniques, languages and user interfaces, distributed architectures.
- **Construction of Data/Knowledge Bases:** Data / knowledge base design methodologies and tools, data/knowledge acquisition methods, integrity/security/maintenance issues.
- **Applications, Case Studies, and Management Issues:** Data administration issues, knowledge engineering practice, office and engineering applications.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

We show that existing proposals to achieve *anonymity* in search logs are insufficient in the light of attackers who can actively influence the search log. However, we show that it is impossible to achieve good utility with differential privacy.

3.1.1 DRAWBACKS

Existing work on publishing frequent item sets often only tries to achieve anonymity or makes strong assumptions about the background knowledge of an attacker.

3.2 PROPOSED SYSTEM

The main focus of this paper is search logs, our results apply to other scenarios as well. For example, consider a retailer who collects customer transactions. Each transaction consists of a basket of products together with their prices, and a time-stamp. In this case ZEALOUS can be applied to publish frequently purchased products or sets of products. This information can also be used in a recommender system or in a market basket analysis to decide on the goods and promotions in a store.

ADVANTAGES

Our results show that ZEALOUS yields comparable utility to k -anonymity while at the same time achieving much stronger privacy guarantees.

3.3 FEASIBILITY STUDY

The development and implementation of a new system is definitely expensive. It requires system resources, manpower, time and money, so it improves the necessity of the feasibility study based on the proposed system requirements. The main objective of this study is to determine whether the proposed system is feasible or not. The study can be categorized into three types. They are

- Technical Feasibility
- Behavioral Feasibility
- Economic Feasibility

3.3.1 Economic Feasibility

This study is carried out to check the economic impact that the system will have. The technique of cost benefit analysis is often used a basis for assessing economic feasibility. The system can be developed at a reasonable cost with the available hardware, software and manpower. So its benefits overweigh the cost.

3.3.2 Operational Feasibility

This aspect of study is to check the level of acceptance of the system by the user. The levels of the acceptance by the users solely depend on the methods that are employed to educate the user about the system and to make him familiar with it.

3.3.3 Technical Feasibility

This study is carried out to check the technical facility, i.e. the technical requirements of the system. The assessment of technical feasibility must be based on an outline design of system requirements in terms of input, output, files, programs and procedures. This can be qualified in terms of volumes of data, trends, frequency of updating, cycles of activity etc, in order to give an introduction of technical system [4].The developed system has a modest technical requirement, as only minimal or null changes are required in implementing the designed system.

CHAPTER 4

SYSTEM SPECIFICATION

SYSTEM REQUIREMENTS

Hardware Requirements

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

Software Requirements

- Operating system : Windows XP.
- Coding Language : Java ,J2EE(JSP, Servlet , JSTL)
- Data Base : MySQL

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 FRONT END

JAVA

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

Java, a platform independent programming language helps in building any kind of application of our interest; Java uses a compiler to convert the source code into architectural independent byte code. These are executed over a Java Virtual Machine (JVM), which is an idealized java processor chip usually implemented in software rather than hardware .Java was developed to include methods for Internet data manipulation [7]. Java applications can be written once and run on any machine having a Java Virtual Machine as part of its operating system.

Features of Java Programming Language

Java technology is both a programming language and a platform.

The Java Programming Language

The Java Programming language is a high-level language that can be characterized by all of the following buzzwords

Simple	Architecture neutral
Object oriented	Portable
Distributed	High performance
Multithreaded	Robust
Dynamic	Secure

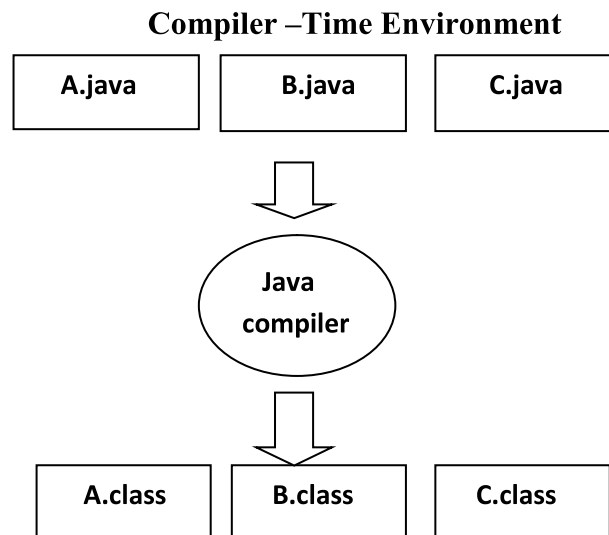


Figure 2: Java Execution Environment

In the Java Programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the Java compiler (javac). A .class file does not contain code that is native to your processor; it instead contains byte codes – the machine language of the Java Virtual Machine. The Java launcher tool (java) then runs your application with an instance of the Java Virtual Machine.

The Java Platform

A platform is the hardware or software environment in which a program runs. We have already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, MacOS. Most platforms can be described as a combination of the operating system and underlying hardware.

The Java platform has two components:

- The Java Virtual Machine.
- The Java Application Programming Interface(API)

The API is a large collection of ready-made software components that provide many useful capabilities. Such as graphical user interface (GUI) widgets. It is grouped into libraries of related classes and interfaces; these libraries are known as packages.

As a platform-independent environment, the Java platform can be a bit slower than native code [7].However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

- The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives you the following features:
- **Development Tools:** The development tools provide everything you'll need for compiling, running, monitoring, debugging, and documenting your applications. As a new developer, the main tools you'll be using are the Java compiler (javac), the Java launcher (java), and the Java documentation tool (javadoc).
- **Application Programming Interface (API):** The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access.
- **Development Technologies:** The JDK provides standard mechanisms, such as Java Web Start and Java Plug-In, for deploying your applications to end users.
- **User Interface Toolkits:** The Swing and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).
- **Integration Libraries:** Integration libraries such as IDL, JDBC, JNDL, RMI, and RMI-IIOP, enable database access and manipulation of remote objects.

Features of the Java Technology

- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks [7]. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++.
- **Avoid platform dependencies:** You can keep your program portable by avoiding the use of libraries written in other languages.

JSP (JAVA SERVER PAGES)

Introduction

While there are numerous technologies for building web applications that serve dynamic content, the one that has really caught the attention of the development community is Java Server Pages TM (JSP). JSP not only enjoys cross-platform and cross-Web-server support, but effectively melds the power of server-side Java technology with the WYSIWYG features of static HTML pages. JSP pages typically comprise of:

- Static HTML/XML components.
- Special JSP tags
- Snippets of code written in the Java programming language called "script lets".

Consequently, you can create and maintain JSP pages by conventional HTML/XML tools. It is important to note that the JSP specification is a standard extension defined on top of the Servlet API [8] . Thus, it leverages all of your experience with servlets.

There are significant differences between JSP and servlet technology. Unlike servlets, which is a programmatic technology requiring significant developer expertise, JSP appeals to a much wider audience.

It can be used not only by developers, but also by page designers, who can now play a more direct role in the development life cycle. Another advantage of JSP is the inherent separation of presentation from content facilitated by the technology, due its reliance upon reusable component technologies like the JavaBeans TM component architecture and Enterprise JavaBeans TM technology.

While this may not seem to be much of a problem for Java developers, it is certainly an issue if your JSP pages are created and maintained by designers, which is usually the norm on large projects. JSP is certainly better for HTML output because you can write a lot of the HTML as plain HTML instead of having to put everything in `out.println ()` statements and escape all the quotes.

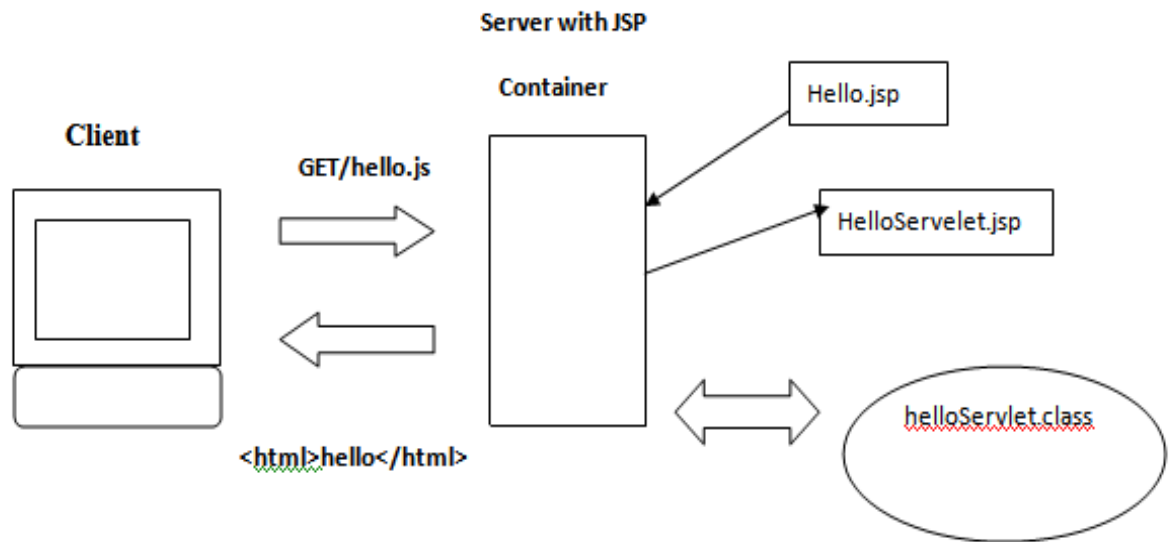


Figure 3: JSP Execution and Processing

The Future of JSP

The future of JSP is not entirely clearly mapped out. As of the time of writing, we are working with the JSP 1.0 Public Draft. This has introduced a number of things (particularly XML syntax and buffered output to help with redirection) and removed other items (particularly extension tags for HTML which has a number of developers wanting to stick with the 0.92 specification).

There is an indication in the Public Draft that XML will become required in the 1.1 specification, and that JSP will be tied more heavily into the Enterprise Java architecture (J2EE –Java 2 Enterprise Edition) that Sun is working on.

JSP Advantages

Separation of Static from Dynamic content

With servlets, the logic for generation of the dynamic content is an intrinsic part of the servlet itself, and is closely tied to the static presentation templates responsible for the user interface. Thus, even minor changes made to the UI typically result in the recompilation of the servlet. This tight coupling of presentation and content results in brittle, inflexible applications.

With JSP, the logic to generate the dynamic content is kept separate from the static presentation templates by encapsulating it within external JavaBeans components. These are then created and used by the JSP page using special tags and script lets. When a page designer makes any changes to the presentation template, the JSP page is automatically recompiled and reloaded into the web server by the JSP engine.

Write Once Run Anywhere

JSP technology brings the "Write Once, Run Anywhere" paradigm to interactive Web pages. JSP pages can be moved easily across platforms, and across web servers, without any changes.

Dynamic Content can be served in a variety of Formats

There is nothing that mandates the static template data within a JSP page to be of a certain format. Consequently, JSP can service a diverse clientele ranging from conventional browsers using HTML/DHTML, to handheld wireless devices like mobile phones and PDAs using WML, to other B2B applications using XML.

Completely Leverages the Servlet API

If you are a servlet developer, there is very little that you have to "unlearn" to move over to JSP. In fact, servlet developers are at a distinct advantage because JSP is nothing but a high-level abstraction of servlets [8]. You can do almost anything that can be done with servlets using JSP--but more easily.

JSP Syntax Basics

JSP syntax is fairly straightforward, and can be classified into **directives**, **scripting elements**, and **standard actions**.

Directives

JSP directives are messages for the JSP engine. They do not directly produce any visible output, but tell the engine what to do with the rest of the JSP page. JSP directives are always enclosed within the, `<%@ ... %>` tag. The two primary directives are page and include.

Page Directive

Typically, the page directive is found at the top of almost all of your JSP pages. There can be any number of page directives within a JSP page, although the attribute/value pair must be unique. Unrecognized attributes or values result in a translation error. For example,

```
<%@ page import="java.util.*, com.foo.*" buffer="16k" %>
```

makes available the types declared within the included packages for scripting and sets the page buffering to 16K.

Include Directive

The include directive lets you separate your content into more manageable elements, such as those for including a common page header or footer. The page included can be a static HTML page or more JSP content. For example, the directive:

```
<%@ include file="copyright.html" %>
```

can be used to include the contents of the indicated file at any location within the JSP page.

Declarations

JSP declarations let you define page-level variables to save information or define supporting methods that the rest of a JSP page may need. While it is easy to get led away and have a lot of code within your JSP page, this move will eventually turn out to be a maintenance nightmare.

For that reason, and to improve reusability, it is best that logic-intensive processing is encapsulated as JavaBeans components. Declarations are found within the `<%! ... %>` tag. Always end variable declarations with a semicolon, as any content must be valid Java statements: `<%! int i=0; %>`

You can also declare methods. For example, you can override the initialization event in the JSP life cycle by declaring:

```
<%! public void jspInit () { //some initialization code } %>
```

Expressions

With expressions in JSP, the results of evaluating the expression are converted to a string and directly included within the output page. Typically expressions are used to display simple values of variables or return values by invoking a bean's getter methods. JSP expressions begin within `<%= ... %>` tags and do not include semicolons:

```
<%= fooVariable %>
```

```
<%= fooBean.getName () %>
```

Scriptlets

JSP code fragments or script lets are embedded within `<% ... %>` tags. This Java code is run when the request is serviced by the JSP page. You can have just about any valid Java code within a script let, and is not limited to one line of source code. For example, the following displays the string "Hello" within H1, H2, H3, and H4 tags, combining the use of expressions and script lets:

```
<% for (int i=1; i<=4; i++) { %>
    <H <%=i%> > Hello </H <%=i%> >
<% } %>
```

Comments

Although you can always include HTML comments in JSP pages, users can view these if they view the page's source. If you don't want users to be able to see your comments, embed them within the `<%-- ... -- %>` tag:

```
<%-- Comment for server side only -- %>
```

A most useful feature of JSP comments is that they can be used to selectively block out script lets or tags from compilation [8]. Thus, they can play a significant role during the debugging and testing process.

Object Scopes

Before we look at JSP syntax and semantics, it is important to understand the scope or visibility of Java objects within JSP pages that are processing a request. Objects may be created implicitly using JSP directives, explicitly through actions, or, in rare cases, directly using scripting code.

JSP Implicit Objects

As a convenience feature, the JSP container makes available implicit objects that can be used within script lets and expressions, without the page author first having to create them. These objects act as wrappers around underlying Java classes or interfaces typically defined within the Servlet API.

The nine implicit objects:

- **request:** represents the `HttpServletRequest` triggering the service invocation. Request scope.
- **response:** represents `HttpServletResponse` to the request. Not used often by page authors. Page scope.
- **pageContext:** encapsulates implementation-dependent features in `PageContext`. Page scope.
- **application:** represents the `ServletContext` obtained from servlet configuration object. Application scope.
- **config:** represents the `ServletConfig` for the JSP. Page scope.
- **page:** synonym for the "this" operator, as an `HttpJspPage`. Not used often by page authors. Page scope.
- **session:** An `HttpSession`. Session scope. More on sessions shortly.
- **exception:** the uncaught `Throwable` object that resulted in the error page being invoked. Page scope.

Note that these implicit objects are only visible within the system generated `_jspService()` method. They are not visible within methods you define yourself in declarations.

Synchronization Issues

By default, the service method of the JSP page implementation class that services the client request is multithreaded. Thus, it is the responsibility of the JSP page author to ensure that access to shared state is effectively synchronized. There are a couple of different ways to ensure that the service methods are thread-safe. The easy approach is to include the JSP page directive:

```
<%@ page isThreadSafe="true" %>
```

This causes the JSP page implementation class to implement the `SingleThreadModel` interface, resulting in the synchronization of the service method, and having multiple instances of the servlet to be loaded in memory.

Exception Handling

JSP provides a rather elegant mechanism for handling runtime exceptions [8]. Although you can provide your own exception handling within JSP pages, it may not be possible to anticipate all situations. By making use of the page directive's `errorPage` attribute, it is possible to forward an uncaught exception to an error handling JSP page for processing.

For example,

```
<%@ page isErrorPage="false" errorPage="errorHandler.jsp" %>
```

informs the JSP engine to forward any uncaught exception to the JSP page `errorHandler.jsp`. It is then necessary for `errorHandler.jsp` to flag itself as a error processing page using the directive:

```
<%@ page isErrorPage="true" %>
```

This allows the `Throwable` object describing the exception to be accessed within a script let through the implicit exception object.

Session Management

By default, all JSP pages participate in an HTTP session. The `HttpSession` object can be accessed within script lets through the session implicit JSP object. Sessions are a good place for storing beans and objects that need to be shared across other JSP pages and servlets

that may be accessed by the user. The session objects is identified by a session ID and stored in the browser as a cookie.

If cookies are unsupported by the browser, then the session ID may be maintained by URL rewriting. Support for URL rewriting is not mandated by the JSP specification and is supported only within a few servers. Although you cannot place primitive data types into the session, you can store any valid Java object by identifying it by a unique key. For example:

```
<%    Foo foo = new Foo();  
  
    session.putValue("foo",foo); %>
```

makes available the Foo instance within all JSP pages and servlets belonging to the same session. The instance may be retrieved within a different JSP page as:

```
<%    Foo myFoo = (Foo) session.getValue("foo"); %>
```

The call to `session.getValue ()` returns a reference to the generic Object type. Thus it is important to always cast the value returned to the appropriate data type before using it.

JAVA SCRIPT

Introduction

Java Script is an interpreted programming language with object oriented capabilities. The general purpose of the language has been embedded in Netscape Navigator, Internet Explorer and other web browsers. The client side version of the JavaScript allows executable content to be included in web pages- it means that a web page need no longer be static HTML but can include programs that interact with the user, control the browser and dynamically create HTML content.

JavaScript is an un typed language that means that variables do not need to have a type specified [9]. JavaScript resembles C, C++, Java with programming constructs such as IF statement, WHILE loop etc. JavaScript was originally called Live Script and its name was changed to JavaScript and it is purely a marketing strategy.

Client side JavaScript

When a JavaScript interpreter is embedded in a web browser, the result is client side JavaScript. Browsers like Netscape Navigator, Internet Explorer, and Opera 3 supports client side JavaScript. Languages like JavaScript, VBScript can be included in a web browser.

Server side JavaScript

Fewer programmers use it. Netscape initially called their server side JavaScript as “LIVEWIRE”. Microsoft also supports server side programming with JavaScript in its web server-using ASP.

5.2 FEATURES

1. Control Document appearance and content

JavaScript Document object, through its Write () method, allows you to write arbitrary HTML into a document. Properties of the Document object allow you to specify colors for the document background, the text and for the hypertext links with in it.

2. Control the browser

Java Script controls the behavior of the browser. The window object supports methods to pop up dialog boxes to display simple messages to the user and to get simple input from the user.

This object also defines a method to create and open/close entirely new browser windows, which can have any specified size and any combination of user controls [9]. JavaScript also allows control, which web pages are displayed in the browser.

3. Interact with HTML forms

Another important aspect of JavaScript is its ability to interact with HTML forms. The form object and the form element objects provide this capability. It can contain Button, Checkbox, Hidden, Password, Radio, Reset, Select, and Submit, Text and Text Area objects. Another common use of client side Java Script with forms is for verification of a form before it is submitted. If client side JavaScript is able to perform all necessary error checking in the client and there is no round trip to the server to detect and inform the user of the errors. Thus it reduces the amount of data that must be transmitted to the server.

4. Interact with the User

JavaScript feature is the ability to define event handlers – arbitrary pieces of code to be executed when a particular event occurs. Usually the user initiates these events.

5. Read and Write Client State with Cookies

Cookie is Netscape's term for a small amount of state data stored permanently or temporarily by the client. Cookies are transmitted to and from the server and allow a web page or web site to remember things about a client.

6. Embedding JavaScript in HTML

There are actually seven ways that JavaScript code can be embedded into HTML documents.

- Between a pair of <SCRIPT> and </SCRIPT> tags.
- From an external file specified by the SRC attributes of a <SCRIPT> tag.
- In an event handler specified as the value of an HTML attribute such as On Click or OnMouseOver etc.
- As the body of a URL that uses the special JavaScript: protocol.
- In a style sheet, between <STYLE TYPE="text/JavaScript"> and </STYLE> tags.
- In a JavaScript entity as the value of an HTML attributes [9].
- In a conditional comment that comments out HTML text unless a given JavaScript evaluates to true

➤ JavaScript in Web Browsers

The features of the programming environment provided by the web browser.

1. The Window object that serves as the global object and global execution context for client side JavaScript code
2. The client side objects Hierarchy.

5.3 BACK END

5.3.1 MYSQL

A Database Management System (DBMS) requires a query language to enable users to access data. Structured Query Language (SQL) is the language used by most relational

database system. SQL is a structured query language and not a procedural language. The SQL language was developed by IBM in the mid-1970s. In 1979, Oracle Corporation introduced the first commercially available implementation of SQL.

The following are the features of SQL:

- SQL is an English-like language. It uses words such as select, insert, and delete as part of the command set.
- SQL processes set of records rather than a single record at a time. The most common form of a set of records is a table [11].
- SQL can be used by a range of users including DBAs, application programmers, and many other types of end users.

SQL provides commands for a variety of tasks including:

- Querying data
- Inserting, updating, and deleting rows in a table.
- Creating, modifying, and deleting database objects.
- Controlling access to the database and database objects.
- Creating tables and other database structures.

5.3.2 SQL COMMANDS

SQL Commands	Description
SELECT	It is used to retrieve data from the database.
INSERT UPDATE DELETE	These are DML statements that insert new rows, change existing rows, and delete unwanted rows.
CREATE ALTER DROP	These are DDL statements that set up, change, or remove data structure, such as tables, views, and indexes.
GRANT REVOKE	These are DCL statements that provide or remove access rights to both the Oracle database and the structures within it.

CHAPTER 6

PROJECT DESCRIPTION

6.1 PROBLEM DEFINITION

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

6.2 OVERVIEW OF THE PROJECT

Search engine companies collect the “database of intentions”, the histories of their users search queries. These search logs are a gold mine for researchers. Search engine companies, however, are wary of publishing search logs in order not to disclose sensitive information. We first show how methods that achieve variants of k-anonymity are vulnerable to active attacks. We then demonstrate that the stronger guarantee ensured by differential privacy unfortunately does not provide any utility for this problem. We then propose a novel algorithm ZEALOUS and show how to set its parameters to achieve probabilistic privacy.

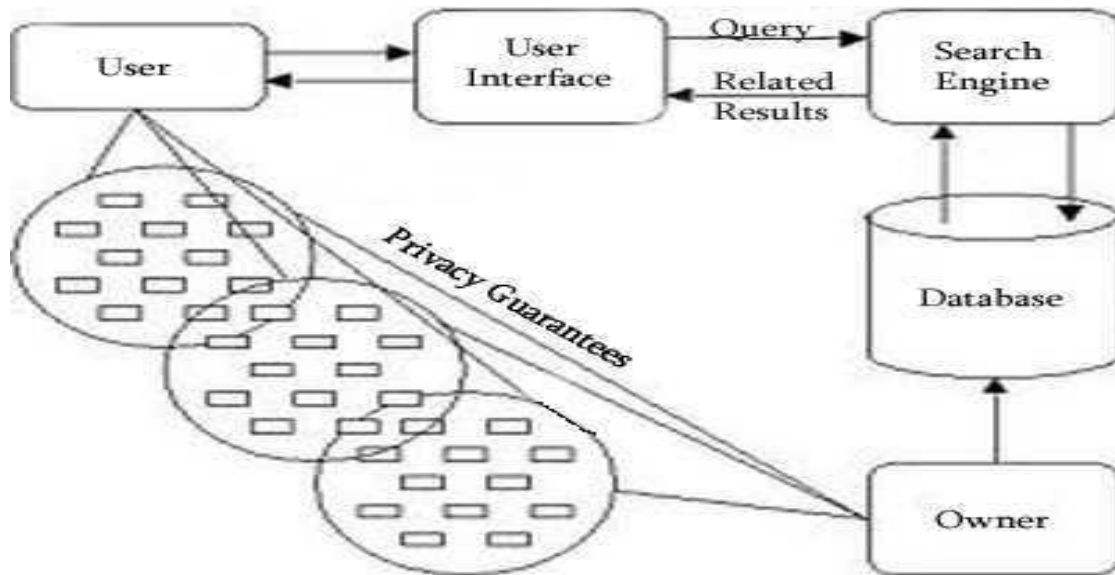


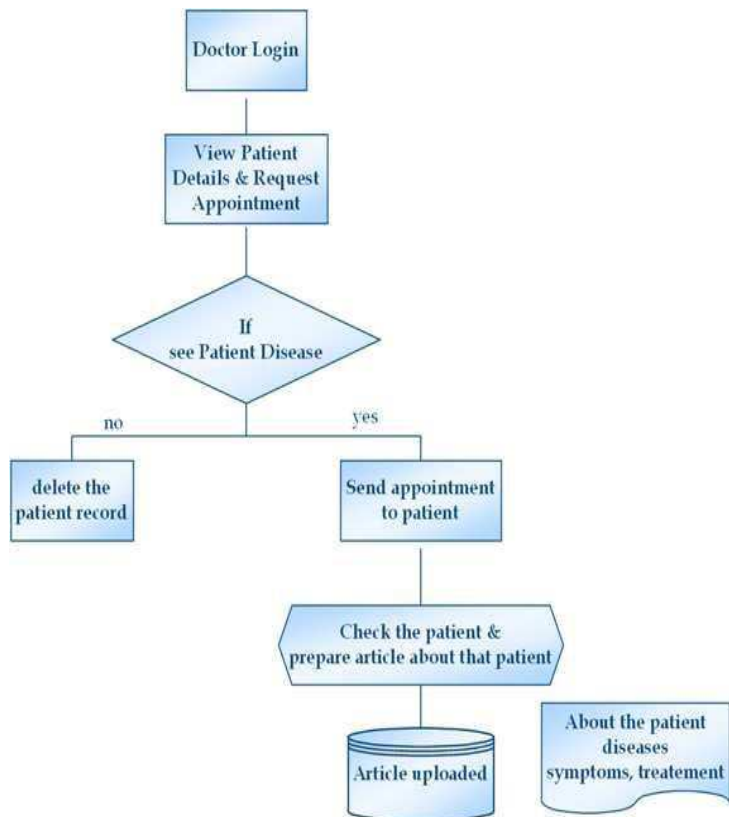
Figure 4: Block Diagram of Publishing Search Logs

We also contrast our analysis of ZEALOUS with an analysis that achieves in distinguish ability.

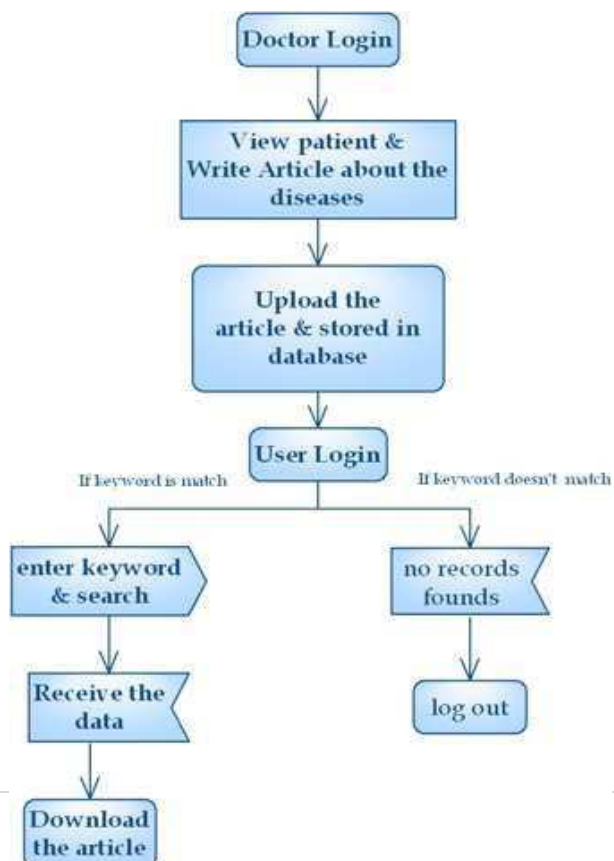
Our paper concludes with a large experimental study using real applications where we compare ZEALOUS and previous work that achieves k-anonymity in search log publishing. Our results show that ZEALOUS yields comparable utility to k-anonymity while at the same time achieving much stronger privacy guarantees.

6.3 UML DIAGRAM

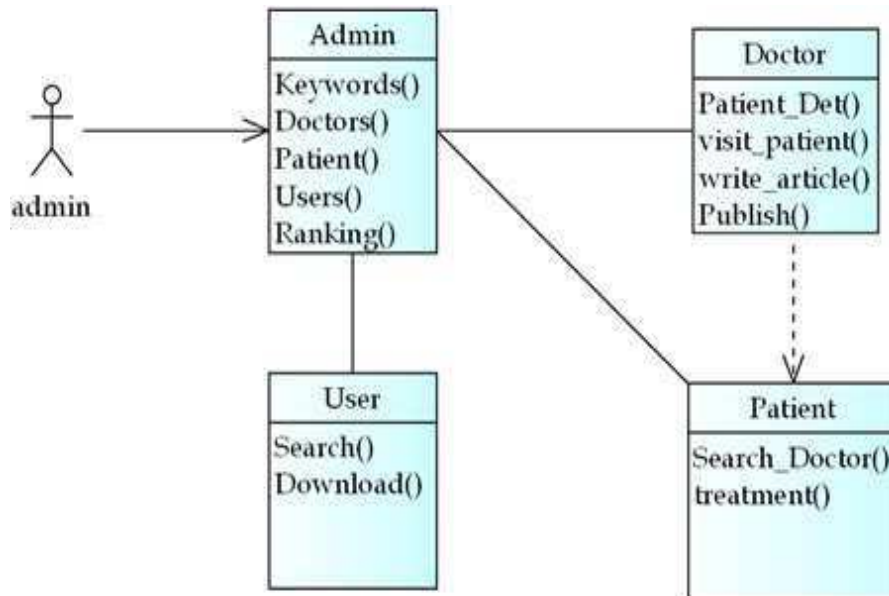
6.3.1 FLOW CHART



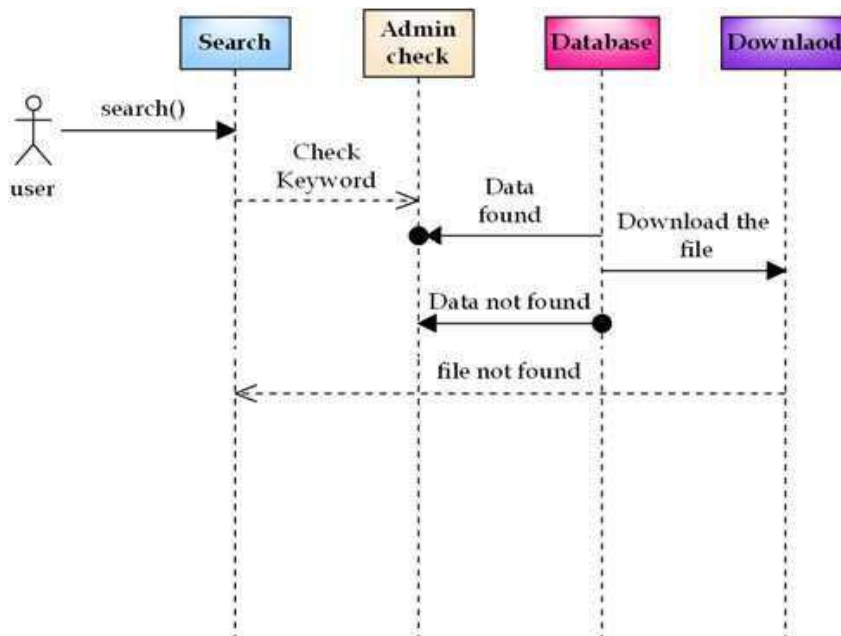
6.3.2 ACTIVITY DIAGRAM



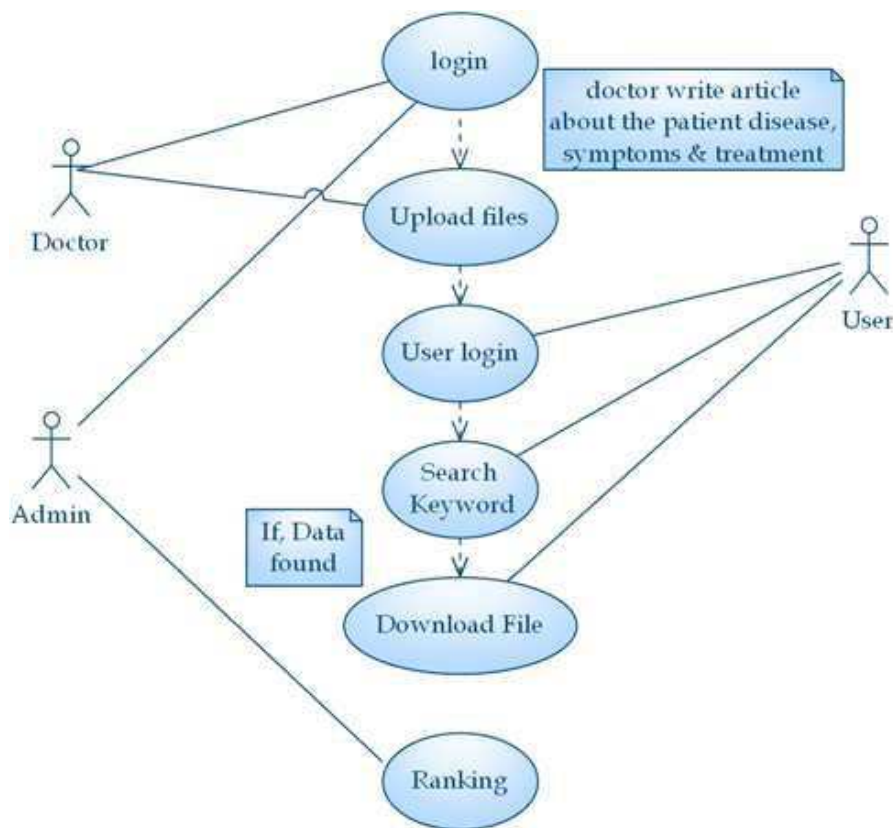
6.3.3 CLASS DIAGRAM



6.3.4 SEQUENCE DIAGRAM



6.3.5 USE CASE DIAGRAM



6.4 MODULES DISCRIPTION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

6.4.1 MODULES

Query Substitution

Query substitutions are suggestions to rephrase a user query to match it to documents or advertisements that do not contain the actual keywords of the query. Query substitutions can be applied in query refinement, sponsored search, and spelling error correction [12].

Query substitution as a representative application for search quality. First, the query is partitioned into subsets of keywords, called phrases, based on their mutual information. Next, for each phrase, candidate query substitutions are determined based on the distribution of queries.

Index Caching

Index caching, as a representative application for search performance. The index caching application does not require high coverage because of its storage restriction. However, high precision of the top-j most frequent items is necessary to determine which of them to keep in memory. On the other hand, in order to generate many query substitutions, a larger number of distinct queries and query pairs are required. Thus should be set to a large value for index caching and to a small value for query substitution. In our experiments we fixed the memory size to be 1 GB.

Item Set Generation and Ranking

All of our results apply to the more general problem of publishing frequent items / item sets or consecutive item sets. Our results (positive as well as negative) can be applied more generally to the problem of publishing frequent items or item sets. We then compare these rankings with the rankings produced by the original search log which serve as ground truth.

6.5 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

6.6 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections.

CHAPTER 7

SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

The implementation process begins with preparing a plan for the implementation of the system. According to the plan, the activities has to be carried out, discussion has been made regarding the equipment, resources and how to test the action [12].

The coding step translates a detail design representation into a programming language realization. The coding should have some characteristics:

- Ease of design to code translation
- Code efficiency
- Memory efficiency
- Maintainability

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

This paper contains a comparative study about publishing frequent keywords, queries, and clicks in search logs. We compare the disclosure limitation guarantees and the theoretical and practical utility of various approaches. Our comparison includes earlier work on anonymity; in distinguish ability and our proposed solution to achieve probabilistic differential privacy in search logs. In our comparison, we revealed interesting relationships between indistinguishability and probabilistic differential privacy which might be of independent interest. Our results (positive as well as negative) can be applied more generally to the problem of publishing frequent items or item sets.

All of our results apply to the more general problem of publishing frequent items/item sets/consecutive item sets. A topic of future work is the development of algorithms that allow publishing useful information about infrequent keywords, queries, and clicks in a search log.

APPENDIX

A1 SOURCE CODE

//Logs.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html> <head>
<title>SEARCH DOCTOR</title>
<link rel="stylesheet" href="style.css" type="text/css" />
<link type="text/css" href="menu.css" rel="stylesheet" />
<link rel="stylesheet" type="text/css" href="pagination/pagination.css" />
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript" src="menu.js"></script>
<style type="text/css">
div#menu {
    margin-top:-8px;
    width:100%;
        margin-left:-5px; }
</style>
<style type="text/css">
th {
    font-family: Arial, Helvetica, sans-serif;
    font-size: .7em;
    background: #666;
    color: #FFF;
    padding: 2px 6px;
    border-collapse: separate;
    border: 1px solid #000;
}

.style2 {
```

```

        color: #FFFFFF;
        font-weight: bold;
    }
    .style3 {color: #FFFFFF}
</style>
<script language="javascript">
history.forward();
</script>
<script language="javascript">
function Alertmsg(){
var r=confirm("SURE YOU WANT TO LOG OUT?" ) ;
if(r==true){
        window.location = "login.jsp";
    }
    else{    window.location = "logs.jsp";
        return false;
    }
}
</script>
</head><body><div align="center">
    <table border="5" cellpadding="0" cellspacing="0" style="border-collapse:
collapse;margin-top:5px" width="740" bgcolor="#EABE94" bordercolor="#DCDBDB">
    <tr><td><table border="0" cellpadding="10" style="border-collapse:
collapse;background-repeat:no-repeat;background-color:#FFFFFF"
background="images/6.JPEG" height="180px" width="987">
    <tr>
    <td>&nbsp;</td>
</tr> </table> </td> </tr> <tr>
<td><table border="0" cellpadding="6" style="border-collapse: collapse" width="100%">
    <tr>    <td><div id="menu" style="margin-left:0px">
    <ul class="menu">
    <li><a href="userhome.jsp"><span>HOME</span></a></li>
    <li><a href="patview.jsp"><span>OWN DETAILS</span></a></li>
    <li><a href="request.jsp"><span>REQUEST APPOINTMENT</span></a></li>
    <li><a href="logs.jsp"><span>SEARCH DOCTORS </span></a></li>

```

```

<li><a href="uppatient.jsp"><span>EDIT DETAILS </span></a></li>
<li><a href=""></a></li> </ul>
<ul class="menu" style="margin-left:-8px">
<li><a href="search.jsp"><span>ARTICLE DOWNLOAD</span></a></li>
<li><a href="repassword.jsp"><span>CHANGE PASSWORD</span></a></li>
<li><a href="login.jsp" onClick="return Alertmsg()"><span>LOGING</span></a></li>
</ul> </div> </td> </tr>
</table></td> </tr> <tr>
<td><table border="1" cellpadding="6" style="border-collapse: collapse" width="100%">
<tr>
<td align="left" valign="top">
<form action="" method="post" name="testform" id="Zealous"
onSubmit="return checkForm(this)">
<table width="522" border="0" align="center"> <tr>
<td width="135" height="44"><label><strong>SEARCH LOGS:</strong></label></td>
<td width="411"><label>
<input type="text" name="t1" style="height: 40px; width: 300px">
<input name="Submit" type="submit" value="Search" class="greenButton"/>
</label></td> </tr> </table>
<p>&nbsp;</p>
<table width="709" border="0" align="center">
<tr>
<td width="703"><table align="center" width="847" height="150" border="0"
cellpadding="1" cellspacing="1">
<tr> <td width="839" height="146">
<div align="center">
<table align="center" width="796" height="40" border="0"
cellpadding="1" cellspacing="1">
<tr bgcolor="#333333">
<td width="130"><div align="center" class="style2">DOC NAME </div></td>
<td width="97"><div align="center" class="style2">AGE</div></td>
<td width="133"><div align="center"><strong><span class="style16
style3">EXP</span></strong></div></td>

```

```

        <td width="128"><div align="center"><strong><span class="style16
style3">HNAME</span></strong></div></td>
        <td width="130"><div align="center"><strong><span class="style16
style3">HADDRESS</span></strong></div></td>
        <td width="145"><div align="center"><strong><span class="style16 style3">MAIL ID
</span></strong></div></td>
        <td width="145"><div align="center"><strong><span class="style16
style3">SPECIALIST </span></strong></div></td>
</tr></table> </div>

```

```

<table width="837" border="0" align="center">
<tr>
    <td width="818" height="35">
        <table width="101%" align="center" border="0">
            <%!int index = 1;%>
            <% try {int flag=0;
String a=request.getParameter("t1");
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection con= DriverManager.getConnection("jdbc:mysql://localhost:3306/logs",
"root", "root");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from doctors where special='"+a+"'");
if(rs!=null)
{
    <%> <%>
while(rs.next())
    {
        <%> <tr>
        <td width="135" ><div align="center"><%=rs.getString(2)%></div></td>
        <td width="95" ><div align="center"><%=rs.getString(7)%></div></td>
        <td width="132"><div align="center"><%=rs.getString(8)%></div></td>
        <td width="112"><div align="center"><%=rs.getString(9)%></div></td>
        <td width="137"><div align="center"><%=rs.getString(10)%></div></td>
        <td width="177"><div align="center"><%=rs.getString(11)%></div></td>
        <td width="177"><div align="center"><%=rs.getString(13)%></div></td>
        <% index++; } } }
catch (Exception e) { out.println(e); }

```

```

        %>          </tr>          </table>          </td>          </tr>          </table>
</td>          </tr>          </table>          </td>
        </tr>          </table>          </form>          </td>
        </tr>          <tr align="center">
                <td><strong>Copy Rights @ Mohanram All Rights Reserved</strong> </td>
        </tr>          </table></td> </tr> </table></div></body>

```

```
</html>
```

```
//Key.jsp
```

```
<html>
```

```
<head>
```

```
<title>KEY VALUE</title>
```

```
<link rel="stylesheet" href="style.css" type="text/css" />
```

```
<link type="text/css" href="menu.css" rel="stylesheet" />
```

```
<script type="text/javascript" src="jquery.js"></script>
```

```
<script type="text/javascript" src="menu.js"></script>
```

```
<style type="text/css">
```

```
th {
```

```
    font-family: Arial, Helvetica, sans-serif;
```

```
    font-size: .7em;
```

```
    background: #666;
```

```
    color: #FFF;
```

```
    padding: 2px 6px;
```

```
    border-collapse: separate;
```

```
    border: 1px solid #000;
```

```
}
```

```
</style>
```

```
<style type="text/css">
```

```
div#menu {
```

```
    margin-top:-8px;
```

```
    width:100%;
```

```
        margin-left:90px;
```

```
}
```

```
.style3 {font-weight: bold}
```

```
</style>
```

```

<script language="javascript">
history.forward();
</script>
<script>
window.oncontextmenu = noRightClick;
window.onkeypress = noRightClick;
function noRightClick (e) {
if(e.which==3) return false;
}
</script>
</head>
<body>
<div align="center">
<table border="5" cellpadding="0" cellspacing="0" style="border-collapse:
collapse;margin-top:5px" width="840"
bgcolor="#EABE94" bordercolor="#DCDBDB">
<tr>
<td><table border="0" cellpadding="10" style="border-collapse: collapse"
width="100%" background="images/6.JPEG"
height="180px" >
<tr> <td>&nbsp;</td>
</tr> </table>
</td> </tr>
<tr> <td><table border="0" cellpadding="6" style="border-collapse: collapse"
width="100%">
<tr>
<td><div id="menu" style="margin-left:0px">
<ul class="menu">
<li><a href="userhome.jsp"><span>HOME</span></a></li>
<li><a href="patview.jsp"><span>OWN DETAILS</span></a></li>
<li><a href="request.jsp"><span>REQUEST APPOINTMENT</span></a></li>
<li><a href="logs.jsp"><span>SEARCH DOCTORS </span></a></li>
<li><a href="uppatient.jsp"><span>EDIT DETAILS </span></a></li>
</ul>

```



```

        <ul class="menu" style="margin-left:-8px">
        <li><a href="search.jsp"><span>ARTICLE DOWNLOAD</span></a></li>
        <li><a href="repassword.jsp"><span>CHANGE PASSWORD</span></a></li>
            <li><a href="login.jsp"><span>LOGGING</span></a></li>
        </ul>        </div>        </td>        </tr>
</table></td>
</tr> <tr>
<td><table border="0" cellpadding="6" style="border-collapse: collapse" width="100%">
<tr>        <td>        <p>        <br>
            </p>        <form method="post" action="Logs">
        <%
                String flag1=(String)request.getAttribute("ss1");
                String a1="";
                String a2="";
                String a3="";
                if(flag1!=null)
                {
                        a1=(String)request.getAttribute("c1");
                        a2=(String)request.getAttribute("c2");
                        a3=(String)request.getAttribute("c3");
                        System.out.println(a1+a2+a3+"search");
                }
        %>
        <table width="713" border="0" align="center">
        <tr>
        <td width="200" height="42"><span class="style3">
<label><font face="Times New Roman, Times, serif" size="+1"
color="#990000">ENTER YOUR KEY :</font></label>
        </span></td> <td width="154" height="42"><label>
                <input type="text" name="a2" style="height: 28px; width: 150px">
        </label></td>
        <td width="410"><label>
                <input name="Search" type="submit" value="Search" class="greenButton"
style="height: 32px; width: 100px" />

```

```

        </label></td>
    </tr>
</table>
<p>&nbsp;</p>
<table border="0" cellpadding="6" style="border-collapse: collapse" width="77%"
align="center">
    <tr style="background-color:#663300;color:#FFFFFF">
        <td width="31%" height="40" align="center"><strong>YOUR
NAME</strong></td>
        <td width="36%" align="center"><strong>DISEASE</strong></td>
        <td width="33%" align="center"><strong>VIEW ARTICLE </strong></td>
    </tr>
    <tr>
        <td height="23" ><span class="style1"></span>
        <div align="center"><span class="style10" ><%=a1%>
        </span></div></td>
        <td height="23" ><div align="center"><span class="style10" >
<%=a2%> </span></div></td>
        <td height="23" ><div align="center" class="fontcolor2 fontcolor2"><span
class="style10" > <%=a3%> </span></div></td>
    </tr>
    <tr></tr>
    <tr>
        <td colspan="9" align="left">&nbsp;</td>
    </tr>
</table> </form>
    </td> </tr>
    <tr align="center">
        <td>&nbsp;</td>
    </tr>
</table></td> </tr>
</table> </div>
</body>
</html>

```

SCREEN SHOTS

1. HOME SCREEN



2. PATIENT REGISTRATION

The screenshot shows the patient registration form. At the top, there is a navigation bar with the following links: HOME, ABOUT US, DOCTOR 'S REGISTRATION, PATIENT 'S REGISTRATION, and LOGGING. Below the navigation bar, the form is titled "PATIENT REGISTRATION FORM". The form contains the following fields and options:

- PATIENT NAME *
- GENDER * with radio buttons for MALE and FEMALE
- DATE OF BIRTH * with a calendar icon
- AGE *
- STREET *
- CITY *
- STATE *
- POSTAL CODE *
- EMAIL ID *
- MOBILE NO *

At the bottom of the form, there are two buttons: SIGNUP and CLEAR.


3. DOCTOR'S REGISTRATION

HOME ABOUT US DOCTOR 'S REGISTRATION PATIENT 'S REGISTRATION LOGGING

DOCTOR REGISTRATION FORM

DOCTOR NAME *

GENDER * MALE FEMALE

DATE OF BIRTH * 

AGE *


EXPERIENCE*

HOSPITAL NAME*

HOSPITAL ADDRESS *

EMAIL ID *

MOBILE NO *

SPECIALIST * 

4. LOGIN



**PUBLISHING SEARCH LOGS -
PRIVACY GUARANTEE OF
PATIENTS PERSONAL
INFORMATIONS**

HOME ABOUT CONTACT DOCTOR REGISTRATION PATIENT REGISTRATION LOGGING

LOGIN FORM

USER NAME *

PASSWORD * 

DESIGNATION* 

ADMIN
DOCTOR
PATIENT

5. SEARCH LOGS



PUBLISHING SEARCH LOGS - PRIVACY GUARANTEE OF PATIENTS PERSONAL INFORMATIONS

- HOME
 - OWN DETAILS
 - REQUEST APPOINTMENT
 - SEARCH DOCTORS
 - EDIT DETAILS
- ARTICLE DOWNLOAD
 - CHANGE PASSWORD
 - LOGING

SEARCH LOGS:

DOC NAME	AGE	EXP	HNAME	HADDRESS	MAIL ID	SPECIALIST
Mahe	25	4	KG	cbe junction north	mahe@gmail.com	Cancer
Ram	24	4	Ganga	cbe	sam@gmail.com	Cancer

6. FIX APPOINTMENT

SEND APPOINTMENT

DOCTOR NAME :*

DOCTOR USERNAME:*

EMAIL ID:*

HOSPITAL NAME:*

SPECIALIST:*

USER PROFILE

PATIENT USERNAME:*

PATIENT NAME:*

EMAIL ID:*

DISEASE NAME:*

DISEASE DISCRPTION:*

7. ENCRYPTION

HOME OWN DETAIL REQUEST APPOINTMENT SEARCH DOCTOR EDIT DETAIL

ARTICLE DOWNLOAD CHANGE PASSWORD LOGGING

SEARCH LOGS:

[PRIVACY](#)

[SEARCH](#)

NAME	DISEASE	ARTICLE DETAILS
ragu	Mt8pZO6jYmReDh118clevw==	WrzMip2kODXd3NdvKchoMg==

8. RANDOM KEY



**PUBLISHING SEARCH LOGS -
PRIVACY GUARANTEE OF
PATIENTS PERSONAL
INFORMATIONS**

HOME OWN DETAILS REQUEST APPOINTMENT SEARCH DOCTORS EDIT DETAILS

ARTICLE DOWNLOAD CHANGE PASSWORD LOGGING

ENTER YOUR KEY :

[Search](#)

YOUR NAME	DISEASE	VIEW ARTICLE
Mahe	cancer	eat well

REFERENCES

- [1] Michaela Götze, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke. Privacy in search logs. *CoRR* abs/0904.0682v2, 2009.
- [2] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 1st edition, September 2000.
- [3] Yeye He and Jeffrey F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009. Yuan Hong, Xiaoyun He, Jaideep Vaidya, Nabil Adam, and Vijayalakshmi Atluri. Effective anonymization of query logs. In *CIKM*, 2009.
- [4] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "I know what you did last summer": query logs and user privacy. In *CIKM*, 2007.
- [5] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *WWW*, 2006.
- [6] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.
- [7] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately.
- [8] In *WWW*, 2009. Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, 2007.
- [9] Yongcheng Luo, Yan Zhao, and Jiajin Le. A survey on the privacy preserving algorithm of association rule mining. *Electronic Commerce and Security, International Symposium*, 1:241–245, 2009.
- [10] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.
- [11] Rajeev Motwani and Shubha Nabar. Anonymizing unstructured data. *arXiv*, 2008.
- [12] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.
- [13] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027.