Journal of Web Engineering, Vol. 15, No.5&6 (2016) 465-483

TOTAL PRIVACY PRESERVATION AND SEARCH QUALITY IMPROVEMENT IN PERSONALIZED WEB SEARCH

LEEMA MATHEW
PG Scholar, Dept. of Comp. Sci. and Engg., TKM Institute of Technology, Kollam, Kerala, India
leemasinoj@gmail.com

ARUN ELIAS *
Professor, Dept. of Comp. Sci. and Engg., TKM Institute of Technology, Kollam, Kerala, India
drearun@yahoo.com

CHINNU RAVI
Asst.Professor, Dept. of Comp. Sci. and Engg., TKM Institute of Technology, kerala, India
chinnuajitha@gmail.com

Received December 15, 2015
Revised April 12, 2016

Personalized Web Search (PWS) has dramatically improved the quality and effectiveness of web search nowadays. Its effectiveness totally depends on the user personnel data collection. But collecting personnel data possess a major threat of privacy risk. Even the search engine side can misuse this personnel information. Evidences show that users are more concerned about their privacy than search quality. We here propose a new system which offers improved search quality and complete privacy protection. A pattern based user interested topic presentation enhances the personalized search. When the user submits a query, our system creates a vector space model with a weight of query related frequent pattern words and then sent to the server side. To avoid the vulnerability of eavesdropping of this vector space model, a modified version of fully homomorphic encryption scheme is used. Experimental results of performance analysis show that our system improves the effectiveness of PWS and execution time.

Key words: Personalized search, Topic model, Relevance ranking, Vector Space Model, Homomorphic encryption
Communicated by: B. White & Y. Deshpande

1 Introduction

Search engine acts as an important gateway for the people to explore the internet. The vast information available in the World Wide Web exists in the form of web pages, images, and other types of files. Search engines include web search engines, desktop search tools, web portals, etc. That has a search provision for online databases. Personalized web search is the ability to identify the different needs of different people who issue the same query for quality search. User interest varies from an individual to another one. There are two basic approaches to personalize search results, one by re-ranking the search results and the other by refining the user's query [2]. Profile based PWS are considered to be the most effective one in improving the quality of web search. For each user in order to provide an effective PWS, it maintains a separate user profile. To create a user profile, search engines have to collect personal and behavioral information about the user. Previously visited pages, browsing history, personal details which include age, gender, education etc., click through data and so forth is stored in the database for user profiling [3] [4] [5]. But the evaluation of the effectiveness of search is difficult [17].

Web search engine collects our personal data to ensure a good quality search. We have to compromise our privacy in order to get an effective personalized web search. Nowadays, cyber-attacks have dramatically increased. As a result, people are more aware in protecting their personal data [6]. A security measure is needed to protect against the loss and misuse of the personal information. Web search engine also misuse our personal information for their personal benefits. For getting the web search, the user submits a query with its runtime profile to the server. Such a search is vulnerable to eavesdropping. As a result an eavesdropper builds a profile of users and thereby receives his/her personal details. In order to overcome all these attacks, a total privacy protected search is a necessity.

In the fast moving world, nobody is ready to waste their valuable time. In the case of web search, user needs a search result according to their interest in a short period of time. If more personalized search, then more personal details of a user is exposed. Such a personalization hinders the privacy. A user needs a system that provides both personalization and privacy. The best way to prevent others accessing our private data is to use an encryption technique. Homomorphic encryption is one of the efficient encryption technique used. Here, calculations carried out on the encrypted data rather than on the plain text. Complex operations can be carried out on the cipher text without having knowledge of its real time data.

2 Related Works

The history of PWS starts from 2004, when famous search engine Google first introduced it. It made a revolutionary change in the web mining area. From those days, many researches and studies are carried out to improve the quality of PWS. But at the start, the concept of web security was not that much seemed to be important. As years passed, the importance of preserving personal web data also increased and studies in improving privacy preservation in PWS also got momentum. In this session, we walk through the previous works related to our paper.

In the primitive stages of PWS, user interaction was inevitable. That is either user's feedback or user ratings were required for personalisation. Then Kazunari Sugiyama and Kenji Hatano proposed a new PWS model based on user profile construction without user efforts [7]. They introduced two types of profile creation in their system. One is content based and the other is based on collaborative filtering. But there were many limitations in this system. This system is not concerned about security at all.

Although privacy in PWS discussed earlier from 2004, an elaborate study of privacy concern in PWS was first done by Xunehua Shen, Bin Tan and Cheng Xing Zhai [8]. In their paper, they clearly defined the four levels of privacy protection in personalized search. UCAIR was a client-side personalized search tool popular in the mid of the last decade. UCAIR is introduced as a browser plug-in. Its main feature is a user profile module which captures user's search queries and clicked search results to form user profiles. Level ш privacy protection can be achieved in UCAIR if we combine it with an anonymous communication system called Tor tool.

J. Castelli-Roca, A. Viejo, and J. Herrera-Joancomarti proposed a system specially designed to protect the users' privacy in front of web search profiling [9]. The system consists of users and a central node. The central node groups $'n'$ users who want to submit a query. User queries are first shuffled and then distributed back by the central node. This process is performed using encryption. Then each user submits the received query to the web search engine. The answer from the search engine is broadcasted to all the group members. Each user takes only his/her answer. The main limitation of this model was the lack of personalisation but it ensured full privacy protection.

3 System Architecture and Contributions

Our proposed Total Privacy Preserved Quality Search System (TPQS) system framework tackles almost all problems related to the privacy preservation in personalized web search. This is successfully done through a client-server system model. The system structure of the proposed TPQS framework is illustrated in Figure 1. In this model, the user using this search service trust only a third party key generator other than himself / herself. Although this is a client-server system, the user does not consider the search server as trustworthy.

Gang Chen, He Bai and Lidan Shou [1] proposed a new system known as UPS (User customizable Privacy-preserving Search) to improve the efficiency of privacy protection in PWS. UPS assumes that the queries given for search do not contain any sensitive information and also ensured an effective search. In this system, an online profile creates a generalized runtime profile in accordance with the query. Then at the time of query submission, with the query instead of full profile this short generalized profile is sent to the search engine. Thus, it can successfully counter any eavesdropping effect. But this system also has many limitations. This system stands on a strong assumption that the eavesdropping starts and ends with a single query session Adversaries with broader background knowledge or capability to capture a series of queries can cause major privacy risk in the UPS framework. That means the attacker can collect these short profiles and can cause privacy attacks.

Traditional information filtering models [16] were developed using a term-based approach. To give an effective web search in UPS frame work [1], both the query and the runtime profile are considered in the server side during query submission. The proposed TPQS framework works effectively, independent of query length, since it sends only the frequent patterns to the search server. These patterns are generated based on individual user interests. During the query submission, the system performs an effective search based on the frequent patterns.

In our system framework, the client side performs the following sequence of operations:

1. Pattern based user interest topic modelling is the first step carried out by the client side. For this, our system fully analyze the data in frequently visited web pages. Then a frequent word pattern representation is generated under each user interested topics. After this a score of importance is assigned to each pattern of this frequent pattern set.
2. When a user issues a query, the client side server searches the pattern model and collects all frequent patterns containing query words. Then creates a set of all words in these patterns. These set of words are represented as a vector space model using their score of importance. Using a modified version of fully homomorphic encryption over integers, this vector space model is encrypted. The encrypted vector space model is then sent to the search engine side for getting the personalized web search results.
3. The search engine serves the results to the user in encrypted form only. Client side performs the decryption of search results.

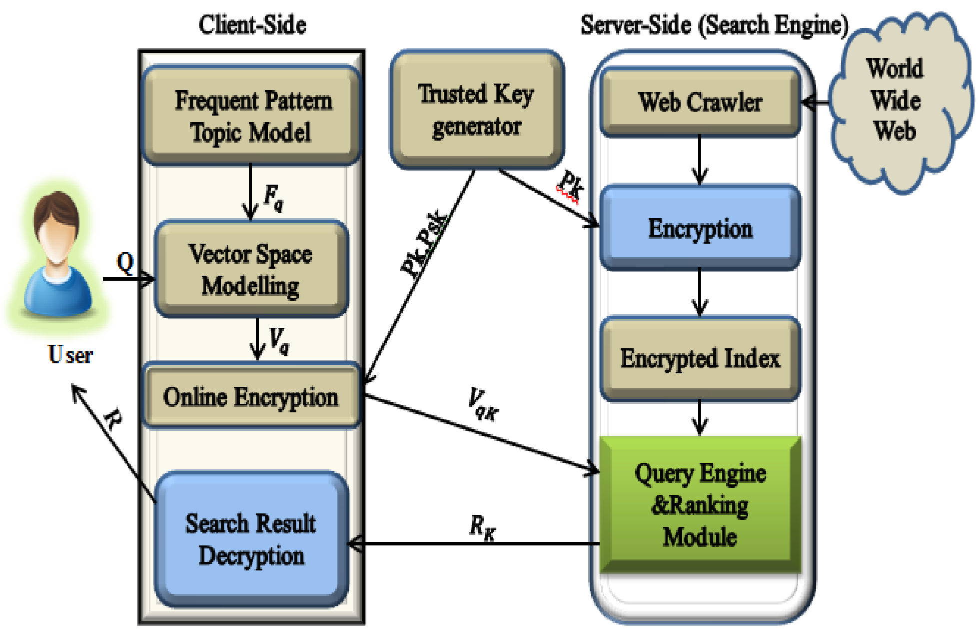

Figure 1 System architecture of TPQS

The search engine side fully co-operate with the client side for a secure and an efficient quality search experience. The server side operation sequence is summarized as follows:

1. A web crawler or spider program continuously crawls the World Wide Web to build/rebuild the searchable index. That is, this program visits the various web pages, reads the data and adds entries to the Index.

2. Server side stores the searchable index as a vector space model of web pages for a quick retrieval of relevant information. That is, each word on the web page is represented by its $tf - idf$ weightage.
3. The vector space model of this searchable index is then encrypted using a modified fully homomorphic encryption scheme over integers. The reason behind selecting fully homomorphic encryption is it allows computations in encrypted form.
4. The server then checks the cosine similarity of the query vector and the Document/web page vectors. Then, the lists of web pages relevant to the query are served to the client side according to their cosine similarity rankings. Result sends to the client side is also in the encrypted form.

   As we said earlier, this system ensures total privacy preservation in personalized web search. Our contributions are as follows:

- A new method of user interested modelling is proposed in this system. By introducing a frequent pattern based user interest modelling, quality of the search will be dramatically increased. Our experimental results show that time lag is also reduced when compared to other similar systems.
- Our modified homomorphic encryption method ensures full privacy protection and thus totally eliminates the threat of eavesdropping.

4 Procedures in Total Privacy Preserved Quality Search System (TPQS)

In this session, we discuss the system model and the procedures associated with the same. We generally use LDA topic modelling and Homomorphic Encryption Technique to lay the base of our proposed Total Privacy preserved Quality Search System (TPQS). As this is a Client-Server System, both sides have to perform many procedures. The main Phases or Procedures are as follows:

1. Web Crawling and Indexing of Web Pages in Encrypted Form.

2. Pattern modelling of User Interested Topics.

3. Vector Space Modelling Of Generalized Frequent Pattern Set and Its Encryption.

4. Cosine similarity analysis of Query and Index vector space models.

5. Decryption of Search Results.

Client side and Server side simultaneously perform the above procedures in TPQS. In the below sessions, we discuss the above procedures in detail.

4.1. Web Crawling and Indexing of Web Pages in Encrypted Form**.**

On the server side or the search engine side, Web Indexing is the primary process to be carried out. For that purpose an internet bot called Web Crawler or Spider [10] is used. The spider starts browsing with a list of popular sites, called seeds, and identifies every link found within the site page and adds them to the list of URLs to visit. This list of URLs is called the crawl frontier. The crawler then visits them according to some set of crawling policies. Crawler performs archiving of websites, copies and savesthe information as it goes. The archives are usually are preserved as 'snapshots' [11]. A typical Web crawler system architecture is illustrated in Figure 2.

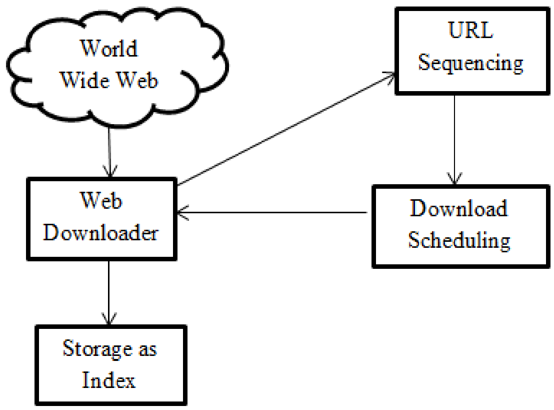

Figure 2 Typical Web Crawler System architecture.

4.1.1. Indexing, Relevance Scoring and Vector Space Modelling of Web Pages

Web indexing is the term used to describe various methods for indexing the contents of web pages on the World Wide Web as a whole. Search engines use keywords and metadata indexing to provide fast and accurate information retrieval. While usually most of the search results are relevant to a user's query, due to how the results are ranked, users often are still not totally satisfied with them. In the work [18], a newly identified web data-quality dimension, appropriateness is based on the linguistic and visual complexity of a web page. Here we first incorporate an inverted index to locate web pages containing the words in a particular query and then rank these web pages by relevance. The Table 1 is a simplified illustration of an inverted index:

Inverted Index Word	Web pages
w1	Web page 1, Web page 2, Web page 3, Web page 5
w2	Web page 2, Web page 5, Web page 7
w3	Web page 4
w4	Web page 7

Table 1 A simplified inverted index.

This index only consists whether a word exists in a particular document, since it contains no information about the frequency and position of the word; so we need a better design to rank the relevance of a web page to a particular query. It is the most challenging parts to include ranking to obtain a better and ordered list of results for a particular query. The basic scheme implements for our ranking is tf-idf Term Weighting and Cosine Similarity. Then we used a vector to represent the document in a bag of words model. For each and every term in the web pages we assign a value called its weight score.

The term frequency, $\mathrm{tf}_{t,p}$ of term *t* in web page *p* is defined as the number of times that *t* occurs in *p*.

$$tf_{t,p} = N_{t,p} \tag{1}$$

If we use pure frequency counts of words, longer documents with word repetitions will be favoured more. To overcome this, term frequencies [9] are normalized. So, the term frequency of a term *t* in web page *P* now becomes:

$$tf_{t,p} = N_{t,p} / \parallel D \parallel \tag{2}$$

||D|| is known as the Euclidean norm

$$\parallel D \parallel = \sqrt{(tfw1,p)^2 + (tfw2,p)^2 + \cdots\cdots\cdots + (tfwn,p)^2} \tag{3}$$

We want to use term frequency, tf when computing the query-document match scores. But raw term frequency is not what we want. For example, a web page with 5 repetitions of a term may be more important than a web page with one occurrence of that term. That means relevance does not increase proportionally with term frequency. So here we take the Log-frequency weighting instead of simple term frequency.

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,p}, & if\ tf_{t,p} > 0 \\ 0, & otherwise \end{cases} \tag{4}$$

The score of a web page-query pair: sum over terms *t* in both *q* and *p* are:

$$Score = \sum_{t \in q \cap p} (1 + \log tf_{t,p}) \tag{5}$$

The score is 0 if none of the query terms are present in the document

Hence, using term frequencies only to measure the weight of a term in the web page is not adequate as it considers all terms equally important. This is because some rare terms are more discriminating than others. For a remedy for this effect, inverse document frequency are used. The document frequency of a term *t* is the number of documents containing the term. In frequent terms, we want positive weights for common words, but lower weights than other rare terms.

We define the *idf* (inverse document frequency) of *t* by

$$idf_t = \log_{10} {}^{N}/_{df_t} \tag{6}$$

The *tf-idf* weight of a term is the product of its *tf* weight and its idf weight [12].

$$W_{t,d} = \left(1 + \log tf_{t,d}\right) \times \log N/df_t \tag{7}$$

This term weightage will increases with both the number of occurrences within a web page and the increases with the rarity of the term in the collection.

For example, consider the data in the following table as an example of a term and document frequencies for three terms in three web pages. Binary term-document incidence matrix of these web pages is at the table 2. Here each document is represented by a binary vector $\in \{0\ ,1\}^V$

Web pages / Words	Web page 1	Web page 2	Web page 3
w1	1	1	1
w2	0	1	0
w3	0	0	1

Table 2 Binary term-document incidence matrix of the Web pages.

This matrix is used for finding the document frequency df_t and thus idf_t of the web pages. Now the term-document count matrices of the same web pages are shown in the table 3 and it is used to find the term frequency of web pages.

From the above table 3 we get the term frequency and thus log frequency weight of term t. Then the weight of the terms in webpages is calculated using equation (7).

Web pages / Words	Web page 1	Web page 2	Web page 3
w1	155	74	1
w2	4	135	0
w3	0	0	57

Table 3 Term-document count matrices of the web pages

Then these weights are represented as a vector space model as in Table 4. So we get a vector space model of dimension |V|. In this vector spae model, words are the axes and web pages are the points or vectors. This is a very high dimensional vector with millions of dimensions. Most entries of this vector are zero.

Web pages / Words	Web page 1	Web page 2	Web page 3
w1	5.25	3.18	0.35
w2	1.21	5.9	0
w3	0	0	2.8

Table 4 Weight matrix of the web pages

Thus an index, I is built as a vector space representation. $I = \{v_i | 1 \leq i \leq n\}$, where $v_i = \{ip_i, tw_{i,1}, tw_{i,2}, tw_{i,3}, \ldots . tw_{i,n}\}$ and $tw_{i,j}$ = Term Weight of w_j in P_i.

4.1.2. Homomorphic Encryption of Searchable Index

We here use a Fully Homomorphic Encryption over Integers for the encryption of our searchable index. The reason is, this encryption scheme allows certain computations over the ciphertext which are inevitable for our program execution. The search engine side server can carry out these computations without knowing anything about the plain text. The result obtained from these computations over ciphertext is the same result when we perform the same operations over the plain text. In our case, we only require addition and multiplication operations of integers as we represent the Web page index in the vector space model of integers. So our homomorphic encryption scheme is less complex than the original fully homomorphic scheme. A modified version of “somewhat homomorphic encryption scheme” [13] is used in our system.

The steps performed in this session are KeyGen and Encryption.

KeyGen: The key is a random odd integer of size η bits, chosen from interval $p \epsilon [2^{\eta-1}, 2^{\eta})$

The public key is $Pk = \{X_0, X_1, \ldots . X_n\}$ and the secret key is $sk = p$

Where $X_i = p.q_i + r_i$, $0 \leq i \leq n$, *p*- secret integer.

Encrypt (pk, m): To encrypt a bit $m \epsilon \{0,1\}$, Choose a random subset $R \subseteq \{1,2, \ldots \ldots n\}$ and a random integer *r* in $(-2^{\beta}, 2^{\beta})$, Where $\beta -$ is the bit length of the noise ri,

Then, output the cipher text:

$$c = [m + 2r + 2\sum_{i \in R} xi]_{Xo} \quad (8)$$

But the main problem of this type of encryption is the large public key size. To overcome lattice-based attacks and for the reduction to approximate GCD, the size of the X_i ’s should be at least γ = 2^{23} bits to prevent lattice attacks. The public key size increases to more than 2^{46} bits, which is impractical for any practical system. So we modify our scheme to meet our requirements.

Output of the cipher text is as follows:

$$c = pq + yr + m \tag{9}$$

where y=$2^{2\|2m\|}$ and p≫ r and $r \gg x$

Now the searchable index I encrypted to I', I'= $\{v'_i | 1 \leq i \leq n\}$, where

$v'_i = \{ip'_i, tw'_{i,2}, \ldots . tw'_{i,n}\}$ and $tw'_{i,j} = Encrypt(PK_{i,j}, tw_{i,j})$,where $PK_{i,j} \in PK$.

4.2. Pattern Modelling Of User Interested Topics

A topic model is a form of statistical model used to discover the relevant topics from a set of documents, were the topics constitutes distribution of words. Latent Dirichlet allocation (LDA) is the most commonly used statistical topic modelling at present. Here, we also use this technique as a base to collect the hidden topics present in our previous click through data sets. Using LDA, the user interested topics are modelled as two representations, i.e., topic distribution and probability distribution over words. This developed model is enhanced to a pattern enhanced form to overcome the disadvantages of ordinary LDA topic modelling technique. In the following sessions, the above steps discussed are elaborated.

4.2.1. User Interested Topic Modelling Using LDA

LDA is a common probabilistic modelling of a document [14]. In LDA, documents are represented as mixtures over latent topics, where each topic constitutes a distribution over words. Here in our case the documents are the click through data or the data in the URLs we visited earlier. User interested topics in that pre visited web pages are modelled here.
Formally, here we define some terms which repetitively come across in our discussion:

Definition 1 (Word): A *word* is the defined as the basic unit of a particular data. Words are represented using unit basis vectors that have a single component equal to one and all other components equal to zero. Thus k^{th} word in the vocabulary is represented by a *V*-vector *w* such that $w_k = 1$ and $w_l = 0$ for $k \neq l$.

Definition 2 (Web page): A *web page* is an online document visited by a user which contains a sequence of m words, P = (P_1; P_2;::: ; P_n), where w_m is the m^{th} word in the sequence.

Definition 3 (History): A *History* is a collection of *n* web pages visited by the user denoted by $H = (P_1; P_2;$ :::: $P_n)$.

LDA assumes the following generative process which is carried out for every web page *w* in our search history *H*:

1. Choose the number of words 'm', the Web page will have.
   Choose $m \sim$ Poisson (ξ) (according to a Poisson distribution)
2. Choose a topic mixture for the document.
   Choose θ~Dir (∝) (according to a Dirichlet distribution over a fixed set of K topics).
3. For each of the *m* words $\boldsymbol{P_m}$:
   (a) First picking a topic Z_m ~ Multinomial (θ). (According to the multinomial distribution)
   (b) Then, using the topic to generate the word itself (according to the topic's multinomial distribution conditioned on the topic Z_n).

We have to make many more assumptions in this LDA model.

Assumption 1: The dimensionality *i* of topic variable 'Z' is assumed as known and fixed.

Assumption 2: The word probabilities are represented by a matrix β), which is considered as a fixed quantity that is to be calculated, where $\beta_{i,j} = p(P_{\ j=1,} Z_{i=1})$.

The LDA model gives three levels of representations. One is the topic representation, the other is a web page representation and the third is the history representation. The topic representation is carried out by the distribution of words and the document representation is carried out using Topic distribution. That is, each web page is represented by topic distribution as follows:

$$\theta_{wi} = (\upsilon_{wi,1}, \upsilon_{wi,2}, \upsilon_{wi,3}, \ldots \upsilon_{wi,k})$$, where k- Number of topics.

Then we can represent the history H by a set of topics, each of which is the probability distribution of words.

$$\Phi = (\Phi_1, \Phi_2, \Phi_3 \ldots \Phi_k)$$

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of *n* topics *z*, and a set of *m* words *w* is given by [14]:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(Z_{Pn}|\theta) p(w_{Pn}|Z_{Pn}, \beta) \tag{10}$$

The parameters ∝ and $\beta$are history level parameters, assumed to be sampled in the process of generating a History. The variables θ are web page-level variables, sampled once per web page.

Finally, the variables Z_{Pn} and w_{Pn} are word-level variables and are sampled once for each word in each web page. Let's discuss the above processes in detail. Consider a web history $H = \{P_1, P_2, \ldots P_n\}$ consists of *'j'* previously visited web pages. Now assume these web pages contains 'n' topics say $Z = \{Z_1, Z_2, \ldots \ldots Z_n\}$ and each documents contains *'m'* words each. The programs check each word on every web page and assigns under any of the given topics. Then for each topic *Z*, calculate two things:

1) $p(topic\ Z|web\ page\ P)$ = the proportion of words in the Web page *P* that are currently assigned to topic *Z*, and

2) $p(word\ W|topic\ Z)$ = the proportion of assignments to topic *Z* over all Web pages that come from this word *W*.

Note that this word assignment already does the first two levels of LDA that is topic representations of all the documents and word distributions of all the topics. Then the iterative step checks how suitable each word in the given topic and each topic in a given web page. This will improve the quality of the result.

For example, history contains three web pages with *'n'* words already visited. First, train the algorithm by describing how many topics in these web pages, say three topics, i.e., Topic 1, Topic 2, and Topic 3. Then the algorithm checks every word and assigns under any of these three topics. If the first word is in Topic 1, it was assigned under that topic. Then check the next word. If the first web page contains 50% words under Topic 1, 30% under Topic 2 and 20% under Topic 3, the probability distribution of topics in the given web page will be like {.5, 0.3 and .2}. Table 5 shows the topic distribution over the web pages and word topic assignments of the given example.

Topic	Topic 1		Topic 2		Topic 3	
Web pages	$\upsilon_{w,1}$	Words	$\upsilon_{w,2}$	Words	$\upsilon_{w,3}$	Words
P_1	0.4	w_1, w_5, w_8, w_9	0.4	w_8, w_7, w_8, w_{10}	0.2	w_1, w_4
P_2	0.3	w_5, w_8, w_5	0.2	w_5, w_6	0.5	w_1, w_5, w_6, w_9, w_1
P_3	0.4	w_3, w_5, w_6, w_7, w_8	0.4	w_1, w_2, w_6, w_1, w_8	0.2	w_3, w_7

Table 5 Result of Word-Topic assignment

From the results of LDA model, we will get the distribution of topic in each web page and also they get topic distribution of whole History.

$$\theta_H = \{\upsilon_{H,1}, \upsilon_{H,2}, \upsilon_{H,3}, \ldots \ldots \upsilon_{H,k}\}, \text{ where K- Number of topics}$$

Topical n-Gram model [15] is a newly proposed model in which the algorithm itself identifies the topic and topically relevant phrases. Thus, this method marginally improves the LDA by including the phrase based topic representation along with word based ones. But this method also has some drawbacks. The low occurrence of phrases in documents is one of that. In order to overcome all challenges of existing model, we use a pattern based approach of topic modelling in this paper to represent the web pages.

4.2.2. Pattern Based Topic Modelling of User Interests.

A pattern type representation of the topics seems to be more precise and meaningful than word type representations. A two-step procedure is required for finding meaningful pattern type topic representations. A transferable dataset construction is the first step and then from this set, frequent pattern based representations of topics are generated.

Let $wt_{Pi,Zj}$ be the set of word sequence in a word-topic representation of the topic Z_j. For example, from the table 5 for topic Z_1, $wt_{P1,Z1} = \{w_1, w_5, w_6, w_9\}$. But for pattern mining, frequency of word within the set is not significant. Transferable set $T_{i,j}$ is a set of words which are in a particular web page P_i assigned under a topic,Z_j $T_{i,j} = \{w/w \in wt_{Pi,Zj}\}$, i.e. $T_{i,j}$ is set of words under a topic Z_j without duplicates. Let $H = \{P_1, P_2, \dots \dots . P_n\}$ be the set of web pages in the history, for the topic Z_j, we can construct a transferable dataset as $\underline{T}_{j=}\{T_{1,}T_{2,} \dots . T_n\}$. For all '$k$' number of topics in H, we can construct transferable datasets as ($\underline{T}_1, \underline{T}_2, \dots . \underline{T}_k$). From the above described example, we get the transferable dataset as in Table 6.

Web pages	Topic 1 dataset, $\underline{T}_1$,	Topic 2 dataset, $\underline{T}_2$	Topic 3 dataset, $\underline{T}_3$
P_1	$\{w_1, w_5, w_8, w_9\}$	$\{w_8, w_7, w_{10}\}$	$\{w_1, w_4\}$
P_2	$\{ w_5, w_8\}$	$\{ w_5, w_6\}$	$\{w_1, w_5, w_6, w_9 \}$
P_3	$\{ w_1, w_6, w_7, w_5, w_8\}$	$\{w_1, w_2, w_6, w_8\}$	$\{ w_3, w_7\}$

Table 6 Transferable Dataset generated from Table 5.

Finding frequently used word patterns by the user is our next main aim. For that, our system analyzes every transferable dataset and generates frequently used word patterns under each topic. Then each user interested topic is represented by the set of these frequent word patterns.

According to the number of occurrences of each word pattern in the transferable data set, each word pattern is assigned by a Frequency Rank, $\underline{r}_f$. For example, by using Table 6 the frequency patterns and its frequency ranks for topic 1 are represented in Table 7. Here the minimum frequency rank, $\underline{r}_{fm}$ is taken as two. That means word patterns with frequency below two are omitted.

Patterns	Frequency Rank, $\underline{r}_f$
$\{w_5\}, \{w_8\}, \{w_5, w_8\}$	3
$\{w_1\}, \{w_1, w_5\}, \{w_1, w_8\}$	2

Table 7 Frequent patterns for Topic 1. $for\ \underline{r}_{fm} = 2$

4.3. Vector Space Modelling Of Generalized Frequent Pattern Set and Its Encryption

Whenever user issues a query, our system starts a search through the frequent patterns of every user interested topics. Then pick up each and every word pattern which contains the words in our query. This set of words is represented as a vector space model. In this vector space model, each word is represented by its Frequency Rank, $\underline{r}_f$. This vector representation is then encrypted and is sent to the search engine side.

Words	Frequency Rank, $\underline{r}_f$
w_5	3
w_8	3
w_1	2

Table 8 Frequency Ranks of Query vector terms

Let us discuss this with the above example in Table 7. If the query word is w_5 , the set of word patterns related query under Topic 1 are $\{ \{w_5\}, \{w_5, w_8\}, \{w_1, w_5\}\}$. From the table 7, we get the corresponding frequency ranks of these three word patterns as {3, 3, 2}. Each word is assigned by a frequency rank using this data. This is illustrated in Table 8.

4.4. Cosine similarity analysis of Query and Index vector space models.

The cosine similarity score between two documents on the Vector Space is a measure that calculates the cosine of the angle between them [16]. This score is used to rank the web pages with respect to the query. The cosine of angle zero degree is 1 and the cosine of a large angle near 180 degrees is close to –1.That is small angles means high similarity near to 1, and large angles should map near to –1.

Figure 3 illustrates the angle between query vector q Web page vectors $P_{1,}P_{2,}P_3$. Here the angle between web page vector P_2 and query vector q is smaller, hence the cosine similarity is higher.

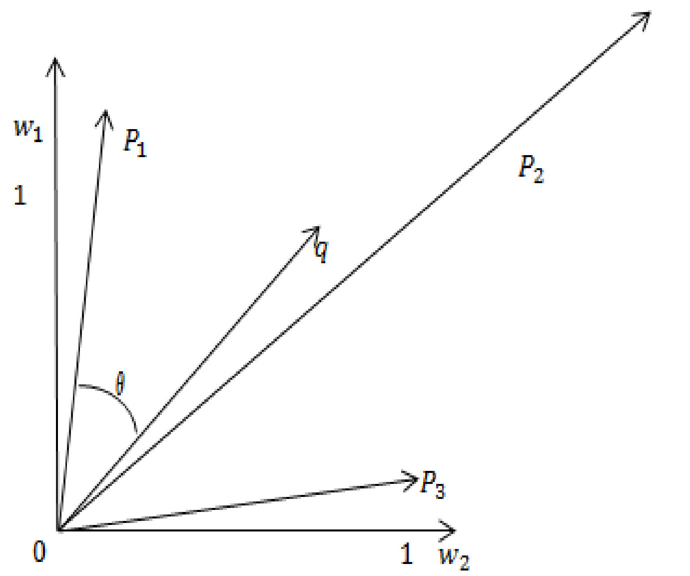

Figure 3 Angle between query vector q and web page vectors $P_{1,}P_{2,}P_3$

Given that encrypted query vector is $\overrightarrow{q'}$ and encrypted web page vector of web page P_i in index I' is $\overrightarrow{P_i'}$, then the cosine similarity $\cos(\theta)$, is represented using the below formula,

$$\cos\left(\overrightarrow{q'},\overrightarrow{P_i'}\right)=\frac{\overrightarrow{q'}.\overrightarrow{P_i'}}{|\overrightarrow{q'}||\overrightarrow{P_i'}|}=\frac{\overrightarrow{q'}}{|\overrightarrow{q'}|}.\frac{\overrightarrow{P_i'}}{|\overrightarrow{P_i'}|}=\frac{\sum_{i=1}^{n}q'_j.P'_{i,j}}{\sqrt{\sum_{i=1}^{n}q'_j}\sqrt{\sum_{i=1}^{n}P'_{i,j}}} \tag{11}$$

Where, q'_j is the frequency rank of term j in the encrypted query vector. $P'_{i,j}$ is the tf-idf weight of term j in web page P_i in the encrypted Index, I'.

Using the above formula, our system calculates the cosine similarity scores between query vector and web pages. A ranking of the web pages are done in accordance with the resulting scores.

5 Evaluation

5.1. Implementation

All algorithms in TPQS framework are implemented in Java language. For Encryption, homomorphic encryption technique is used. The web data and its index are stored on a PC with an Intel core-I5 2.33GHZ CPU, 4 GB Ram running on windows 8 operating system. Two other machines with the same configuration are used as search engine server and client-server respectively. For better evaluation, we use user authentication for server. For that each user has their own login credentials.

The experiments tested across the 15,000 web pages under 6 different categories on one of our PC. The 6 categories are sports, music, education, politics, agriculture and computer science. Index of these web pages is maintained and thereby stored in encrypted form. Login credentials are given to 20 users of various interests that come under these categories. We allow the users to use the system without enabling our framework (pattern based topic modelling) on client side for some days. Following that, we enable our framework on client side and allowed them to use it.

5.2. Experimental Results

In this session, we present the results of tests conducted to measure the performance of our TPQS framework**.**

5.2.1. Experiment 1: Search Quality Analysis of TPQS

From our experimental results, it is evident that the quality of web search is increased when using our TPQS system. In this analysis, we compare the precision and sensitivity of the retrieved results with and without our system framework enabled. Here 'precision' is defined as the fraction of relevant results in retrieved results, while sensitivity is the ratio of relevant results that are retrieved. Our result also shows that the quality of search increases as the query length increases.

For this analysis, we allowed one of our regular users to login with his credentials and let him search a topic with a number of queries with our system. We also had done the same without enabling our system. Then we asked the same user to categorize the top fifteen results under relevant and non-relevant categories. Based on the query and previously visited webpages, frequent words are stored on the database. While increasing the description or the length of the query, frequent patterns with more user interested topics can be stored. Quality of search will be increased with more specific frequent patterns stored. Then we calculate the precision score of the searches using the following formula.

$$Precision\ score = \frac{|\{relevent\ web\ pages\} \cap \{retrieved\ web\ pages\}|}{|\{retrieved\ web\ pages\}|} \tag{12}$$

From the results of the experiment,a graph(Figure 4) of precision vs number of webpages for both the above cases is plotted. In comparison, the following two results were observed:

1. Precision increased while using our system framework.
2. Precision increased with more query description or query length.

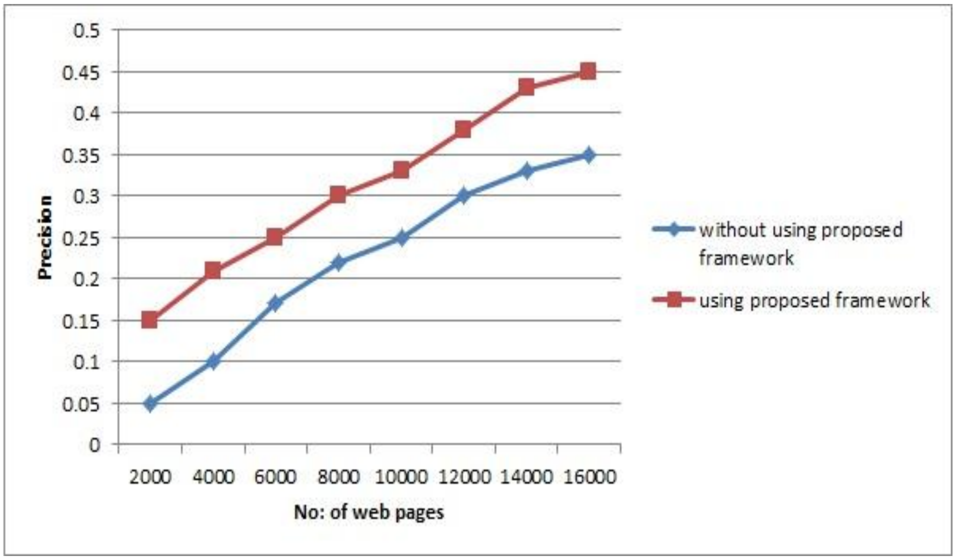

Figure 4 Analysis of search quality in terms of precision

Now for measuring the sensitivity of the search results, we asked the user to analyze the total 15,000 web pages in our database and select the relevant results for his respective queries. Then, calculate the sensitivity ratio of webpages with and without our Pattern Based Topic Modelling system. From our experiments conducted, we plot a graph (figure 5).

$$Sensitivity = \frac{|\{relevent\ web\ pages\} \cap \{retrieved\ web\ pages\}|}{|\{relevant\ web\ pages\ in\ database\}|} \tag{13}$$

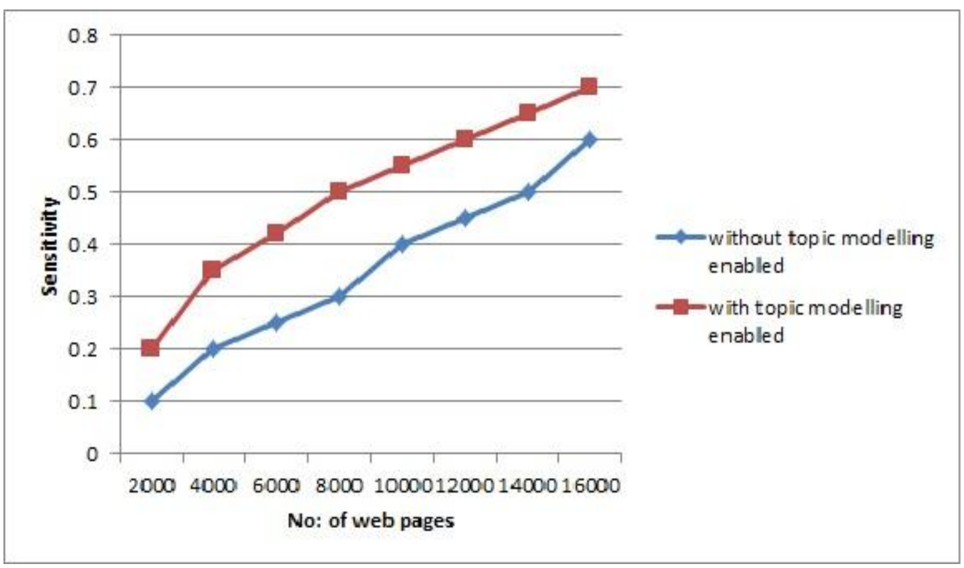

Figure 5 Analysis of search quality in terms of sensitivity

5.2.2. Experiment 2: Analysis of query execution time

The introduction of pattern based topic modelling of user interests dramatically decreased the query execution time in our system. In the proposed TPQS framework, frequently used word patterns have been analyzed and such a word pattern is sent to the server. That seems to be simpler and faster than the online profiling system introduced in the existing system i.e., the UPS framework[1]. Then,the query execution time of queries with different sizes are analyzed. Figure 6 shows the results of query execution time analysis. For the personalized web search, web server considers all the topics in the existing system and hence it is a time consuming process. In TPQS, a frequent word pattern representation is generated under each user interested topics. This frequent pattern set is used for generalizing profile. Hence web server takes less time for giving personalized search results.

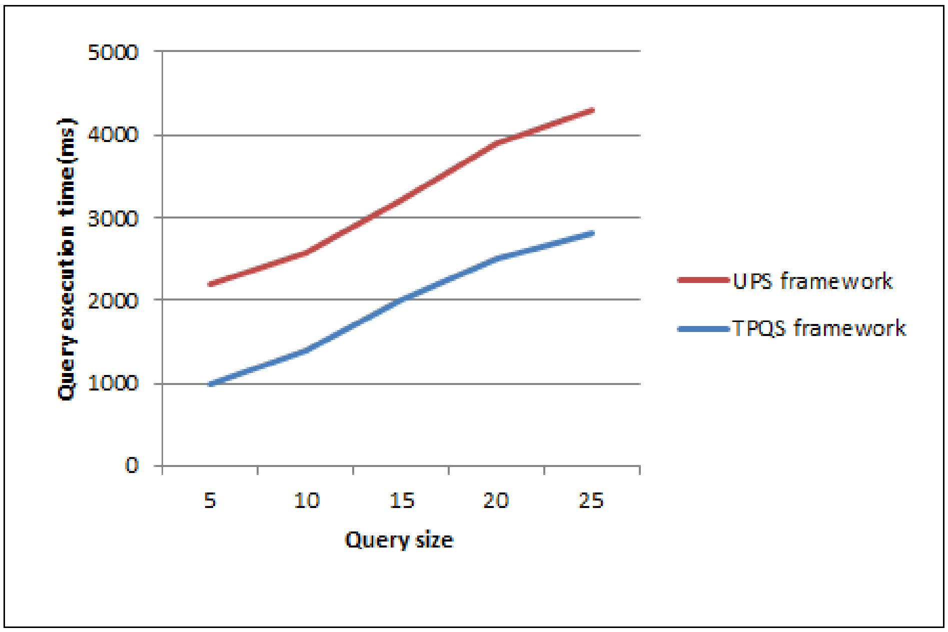

Figure 6 Query execution time analysis

4 Conclusion

Our proposed TPQS model generates pattern based representations to model user's interests across multiple topics. In comparison with other user profiling models, the proposed model demonstrates excellent quality of document modelling and relevance ranking. By proposing vector space modeling on both sides, the similarity analysis of queries and web pages became simpler and faster. From the above experimental results, it is evident that both the precision and sensitivity of the search results have increased with the introduction of the pattern based topic profiling in our system. Also, it must be noted that the execution time of queries is decreased here. Total privacy protection in personalized web searching is achieved in our system by using a simple form of fully homomorphic encryption scheme.

References

1. Lidan Shou, He Bai, Ke Chen, and Gang Chen, "Supporting Privacy Protection in Personalized Web Search", In Proc. IEEE Trans.on Knowledge and Data Eng., vol.26, issue 2, February 2014.
2. Pitokow James, Hinrich Schütze, Todd Cass, Rob Cooley, Don Turnbull, Andy Edmonds, Eytan Adar, and Thomas Breuel (2002), "Personalized search", Communications of the ACM (CACM) 45 (9): 50–55.
3. Jaime Teevan, Susan T. Dumais and Eric Horvitz (2005). "Personalizing search via automated analysis of interests and activities", In Proc. 28th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval, August 2005..
4. Zhicheng Dou, Ruihua Song and Ji-Rong Wen(2007). "A large-scale evaluation and analysis of personalized search strategies", In Proc. 16th Int'l Conf. World Wide Web, ACM Press.
5. Paul Alexandru Chirita, Claudiu S. Firan, Wolfgang Nejdl (2006). "Summarizing local context to personalize global Web search", In Proc.15th ACM Int'l Conf. on Information and knowledge management.
6. Harry, David. "Search Personalization and the User Experience". Retrieved 29 April 2014.
7. K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive Web Search Based on User Profile Constructed without any Effort from Users", In Proc. 13th Int'l Conf. World Wide Web (WWW), 2004.
8. Xuehua Shen, Bin Tan, and ChengXiang Zhai, "Privacy Protection in Personalized Search", ACM SIGIR Forum, vol. 41, no. 1, pp. 4-17, June 2007.
9. J.Castelli-Roca, A. Viejo, and J. Herrera-Joancomarti´, "Preserving User's Privacy in Web Search Engines", Computer Comm., vol. 32, no. 13/14, pp. 1541-1551, 2009.
10. Spetka and Scott, "The WWW Robot: Beyond Browsing", NCSA. Archived from the original on 3 September 2004. Retrieved 21 November 2010.
11. Castillo and Carlos, "Effective Web Crawling (Ph.D. thesis)", University of Chile, Retrieved 2010-08-03.
12. C. D. Manning, P. Raghavan and H. Schutze, (2008). "Scoring, term weighting, and the vector space model", Introduction to Information Retrieval . p. 100.
13. Marten van Dijk, Craig Gentry, Shai Halevi and Vinod Vaikuntanathan. "Fully Homomorphic Encryption over the Integers", June 8, 2010.
14. David M. Blei, Andrew Y. Ng, and Michael I. Jordan, "Latent dirichlet allocation", Journal of Machine Learning Research 3 (2003) 993-1022.
15. X. Wang, A. McCallum, and X. Wei, "Topical n-grams: Phrase and topic discovery, with an application to information retrieval", In Proc. 7th IEEE Int. Conf. Data Min., 2007, pp. 697–702.
16. Singhal and Amit, "Modern Information Retrieval: A Brief Overview",Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43.
17. D.E. Rose, "Why Is Web Search So Hard... to Evaluate?", Rinton Press,Vol.3, No.3&4 December, (2004) 171-181.
18. J.C.C. Pun and F.H. Lochovsky," Ranking Search Results by Web Quality Dimensions", Rinton Press., Vol.3 No.3&4 December, (2004) 216-235.