Dynamic Tree Based Classification of Web Queries Using B-Tree and Simple Ordinal Classification Algorithm

L. Lakshmi, P. Bhaskara Reddy
MLR Institute of Technology
Hyderabad-43, India
laxmi.slv@gmail.com
pbhaskararedddy@rediffmail.com

C.Shoba Bindu
JNTUA, Anantapur
India
shobabindhu@gmail.com

Journal of Digital Information Management

ABSTRACT: *Queries submitted by users to search engines might be ambiguous, concise and their meaning may change over time. Web query classification is emphasized by various search engines nowadays due to the increase in the size of the web, as millions of web pages are added to it every day. Some of the current Information Retrieval (IR) systems like Library Online Public Access Catalog (OPAC), dialog system and numerous web search engines need classical Boolean approaches in addition to the current supervised methods. Document retrieval is a process which mainly involves retrieval of relevant documents for user queries and matching the results using efficient algorithms like 'page rank' and 'learn to rank' algorithms. In this paper, to retrieve more relevant documents against user query by reducing non-relevant documents, we proposed a tree based classification of web queries using Simple Ordinal Classification (SOC) and navigation of search keywords is performed dynamically with search session time of users. This method reduces the retrieval of most of the non-relevant documents and navigation cost using efficient B-tree data structure. It provides documents that match all keywords present in the user query and best resulting web pages for users with different categories of interest. We built a prototype application to evaluate the proposed approach. Our experimental results revealed that SOC has significant performance improvement with existing approaches.*

Subject Categories and Descriptors
[H.3.1 Content Analysis and Indexing]: [H.3.5 Online Information Services]: [H.3.3 Information Search and Retrieval]; Query formulation

General Terms
Query Classification, Web Query, Tree-based Classification, B Tree Indexing

Keywords: Query classification, Dynamic navigation, Learn to rank systems, Simple Ordinal Classification, Session log

Received: 7 January 2017, Revised 12 March 2017, Accepted 28 March 2017

1. Introduction

Search engines over the World Wide Web (WWW) such as Google, play a vital role in providing required information to people in all walks of life. With the exponential growth of data, WWW has become data-rich. It has wealth of data in the form of text documents, multimedia files containing images, audio and video, web directories and so on. Due to the voluminous data available over WWW, search engines often produce numerous records against a user query. For instance, the query with the word "Java" given in Google search engine has resulted in 51,80,00,000 hits. It shows the bulk of information output provided by

to browse the results to know most relevant ones unless the results are optimized or shown according to certain ranking. This is a major issue in the web research arena. Another important problem encountered with user queries is the inability of users to formulate good queries. Therefore, the application that processes user requests needs to have know-how to understand and interpret user queries in a meaningful way.

These problems can be minimized by using a web query classification system. With classification, there are many advantages such as reduction in navigation time and achieve minimal loss function. This is a challenging phenomenon especially due to massive size of the web and its dynamic nature in content and queries encountered from users from time to time. In this context, understanding the intention of the user when a query is made is very important. Query classification is the process of dividing web queries into pre-defined categories. As mentioned above, this task is tedious as there might be phrases and short words involved in user queries. A word can have different meanings based on context. For instance, the word apple refers to a fruit in one context but in another context, it refers to the Apple Company. This discrimination is another difficulty involved in the web query classification. Query classification has many utilities in the real world. One such utility is context-based advertisement where advertisements are posted based on the context.

Many researchers contributed towards information retrieval research and web query classification. It is understood as explored in [21] that information retrieval approaches based on single source suffer from the performance issues. On one hand link-based approaches result in the problem "*richer will get richer"* [25] and on the other hand session-based approaches need a more sophisticated approach to hand diversity of users. Different query classification approaches are found in [13], [16], [22], [26]. The problem with most of the approaches is that there is a lack of efficiency when session-based approach is followed. We consider this issue as an optimization problem and propose a new classification approach known as Simple Ordinal Classification (SOC) which is based on B-Tree. Towards this end, as part of SOC, we proposed two algorithms for calculating the loss function and dynamic navigation of B-tree.

We built a prototype application that demonstrates the proof of the concept. Experiments are made to have session- based queries, computation of loss function, dynamic navigation, keywords, dynamics of retrieved documents and performance comparison with other techniques. The remainder of the paper is structured as follows. Section 2 reviews literature on information retrieval and web query classification. Section 3 presents an overview of the proposed web classification system. Section 4 presents the proposed algorithm for computing loss function. Section 5 presents the navigation process of B-tree. Section 6 provides experimental results of loss function with session-based queries. Section 7 presents results of dynamic navigation of B-tree with SOC. Section 8 provides performance analysis. Section 9 concludes the paper while section 10 throws light into the possible future work.

2. Related Work

Web surfs in these days enable customers to pose queries conveniently in terms of key terms. Most of the search queries are expressed by a couple of keyword phrases. It leads to a condition that the search queries are also ambiguous [1] [16]. It is understood that web users decide on terms to represent their understanding and desires while search queries display implicit and subjective consumer intents [22]. User query classification is mainly categorized into three types according to their purpose, such as informational queries, navigational queries and transactional queries [5]. Informational queries cover topics broadly resulting in thousands of web pages. Navigational queries cover fewer topics resulting in few web page links. Transactional queries cover topics related to the purpose of user intent, resulting in transactional web page links [2] [8]. There are many algorithms for classification of web queries. Early models like query dependent models such as Boolean models mainly retrieve documents based occurrences of query terms in the document [17]. These models use vector space model, and term frequency-inverted document frequency; but using these models, it is very difficult to retrieve efficient results [6]. Query independent models are mainly the Page Rank models and are based on the importance of the documents, but these models face the problem of *'richer get richer'* [10].

Beitzel proposed automatic web query classification using labeled and unlabeled training data but this method was unable to tackle web query classification precision problem [21]. Diemert proposed unsupervised query categorization using automatically-built concept graphs but this method results in more number of non-relevant documents [16]. Lovelyn proposed a classification of web queries using normalized web distance but this method is unable to find a search context, if the query is present at the beginning of the session [11]. Eda Baykan proposed the performance of the dissimilar URL-based language classifiers on a range of dimensions like options, algorithms, and training size, tested on the Open Directory Project for Search Engine Results dataset, but they do not deal with special characteristics of data [6].

There are many *learn to rank* machine learning algorithms used for reinforcement, semi-supervised and supervised learning in recent years. Mainly *learn to rank* algorithms are classified into pointwise, pairwise and listwise approaches [10]. Pointwise approach mainly consists of three sub categories such as regression, classification and ordinal regression based algorithms, but the problem with pointwise approach is the relative order or rank or distance between the documents is not properly modeled. Pairwise approaches mainly focus on relative order or rank between the documents. In this instance, the ranking

problem is reduced to the classification problem between document pairs [15]. Most used algorithms in this approach are RankNet, FRank, RankBoost, and Ranking SVM. The problem with Pairwise approach, if the documents are distributed in an imbalanced manner across the queries, will present inaccurate results than pointwise approaches [12].

Listwise approaches focus on relative order or rank between the documents and positional information for web documents. Mainly used algorithms in this approach are SoftRank, SVMmap, SVMndcg, AdaRank and ListMLE. The problem with listwise approaches is the high training complexity as their loss function uses permutation [12], [19]. The Web query classification can be considered as a multilevel categorization problem, as there are a numerous machine learning algorithms to categorize the web queries [4]. The main purpose of all these machine learning algorithms is to achieve two important objectives first to provide an accurate ranking for web pages and second to minimize the loss function with less training complexities [12].

To overcome the drawbacks in aforementioned works, in this paper, we propose an efficient and effective combined classification in regression and B-tree navigation algorithm to optimize the loss function and to provide an accurate ranking to the web documents. Here, we use regression with simple ordinal classification algorithm to reduce the loss function values in regression.

3. Overview of The Proposed System

In this paper, we propose an efficient methodology for web query classification. The overview of the proposed methodology is as shown in Figure 1. When the user issues a query, first we collect the training data from LETOR (Learn to Rank data set) data set [14]. The data set mainly consists of queries, urls, features, training, testing and validation of queries [14]. This training data also has session logs for various queries. In the proposed method, the queries are mainly classified into three categories. Determination of the user-issued query belongs to a specific category and can be done using Simple Ordinal Classification algorithm.

The Simple ordinal classification algorithm is mainly a combination of classification in regression and calculation of pointwise loss function. The main goal of this algorithm is to minimize the loss or error function. Using this algorithm, we determine the query belongs to a specific category based on the calculation of loss function using sessions. Categories with minimum loss function are chosen to retrieve the documents. We use B-tree to represent the categories of web documents, and after calculation of categories of the query, we apply edge removal algorithm based on the phrases that match the query.

The session logs collected from LETOR data set [14] are used as training data. The training step uses *q* queries, *x* documents and *y* relevant documents represented as *I* (*q, x, y*). For the calculation, a query belongs to *a* specific category, we use support vector machines classification regression model. Using this model all relevant documents having y_i=1and non-relevant documents having y_i=0, the loss function is defined as, for set of *n* documents, number of matched documents subtracted from classification of *x* documents. Using only support vector model does not produce effective results and loss function is the sum of squared errors for standard Information Retrieval measures like Normalized Discounted Cumulative Gain, Mean Average Precision, and Mean Reciprocal Rank. So we combined our support vector model with the simple ordinal classification algorithm. It is a simple iterative model to minimize the loss function and sum of squared errors. Using this query classification algorithm, the time complexity for the retrieval of documents is < (log n), and it reduces most of the non-relevant documents. In this section, we introduce some important notations and definition used in the classification of web queries.

Definition 1. (Session): The Session is defined as list of observations made by the user over a period of time.

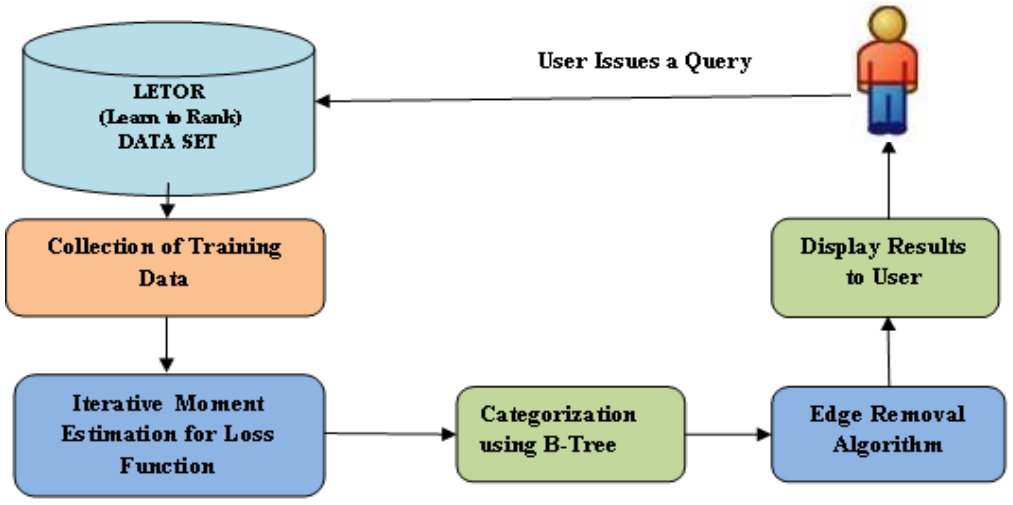

Figure 1. Architecture of Proposed Dynamic Navigation System

Generally in a session, user searches for same information and vocabulary used by *the* user is same.

Definition 2. (Title): Name of the given URL.

Definition 3. (Clicked Title): No of titles clicked by the user during a session.

Definition 4. (Unclicked Title): Searches for the query but not clicked on any title.

Definition 5. (Loss Function**)**: Loss function in simple ordinal classification is defined as

$$LF\,[X,Y] = PS_{xi\sim D}[H(x) \neq y_i] \quad (1)$$

Where
$X = \{x_1, x_2, x_3, \ldots\ldots, x_n\}$ represents set of documents vector.

$Y = \{y_1, y_2, y_3, \ldots\ldots, y_n\}$ represents relevancy vector contains set of documents visited by user in a session.

$$y_i = \begin{cases} 1 \text{ Title Clicked by user} \\ 0 \text{ Title Unclicked by user} \end{cases} \quad (2)$$

H:X→R where R represents relevant documents that is H:X→{ $y1, y_2, y_3, \ldots\ldots, y_n$}

H(x) is the classifier output by simple ordinal classification algorithm.

D is *the* uniform distribution over x_i documents. D∈ [$x_1, x_2, x_3, \ldots\ldots, x_n$]

PS represents predictive success function, used to count number of documents relevant to the given query The Loss function for *N* documents can be calculated as

$$LF = \frac{1}{n} \sum_{i=1}^{n} (y_i - H(x_i))^2 \quad (3)$$

Loss function for *N* documents can be calculated by using *the* equation (1) for a single document, where y_i represents *the* number of titles clicked by the user using equation (2).

Definition 6. (Navigation B-tree): A Navigation B-tree *T*(*V*,*E*,*C*) is a tree with *V* nodes, *E* links and a degree of *C*, where *C* represents the categories of queries. The construction of a tree *T* starts with root as *C*, outside and enclose the navigation tree is iteratively computed in a pre-order (root-left-right) traversal.

Figure 2 shows navigation B-tree for simple one phrase query "*web mining*". Basically, web mining is a query related topic relevance, and then the navigation process in B-tree starts with the informative category. As the web contains billions of web pages, if we construct a simple tree, the depth of the tree is more, due to which time and space complexity of the navigation process is high. So the construction of B-tree for navigational hierarchy reduces the size of the tree and number of levels in the tree as it is a height balanced tree.

Definition 7. (Remove edge): Remove edge is a process of removing non-relevant edges and dynamically depends on keywords present in the given query. In the Figure 3, there are three edges for various types of queries, while the user issues a query depends on keywords present in the query and based loss function for the query must

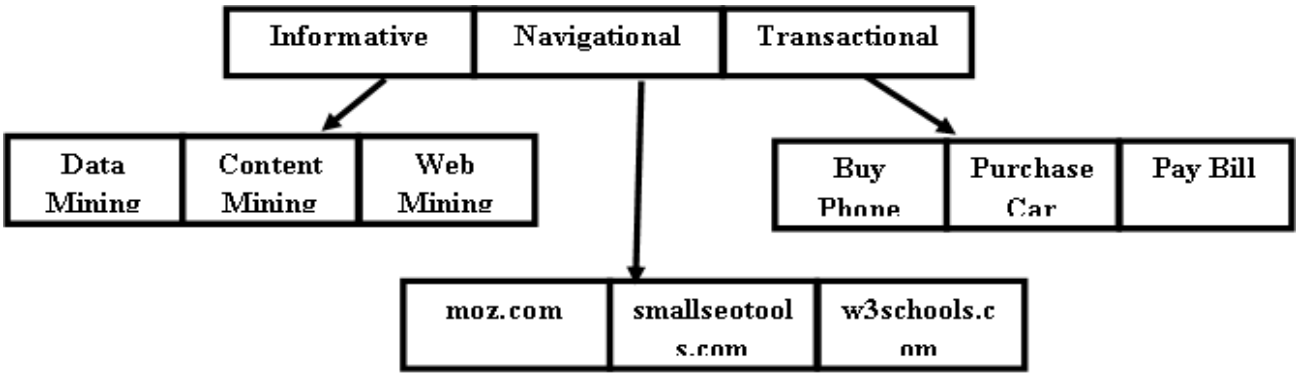

Figure 2. Construction of B-tree for a given query

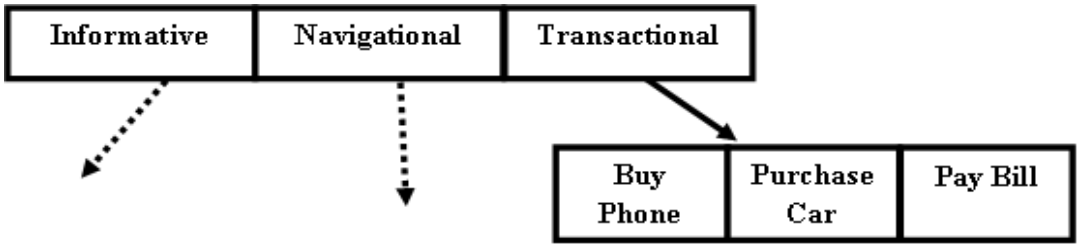

Figure 3. Navigation of B-tree dynamically

belong to one of the three types, then remaining edges are removed due to non-relevance. For example, when the user issues a query "*buy a phone",* then it belongs to transactional query, so we no longer need documents related to informational and navigational queries. By applying Remove edge process, we remove the edges of informative and navigational queries. The result lists of these categories are omitted and represented with dotted lines.

Remove edge process reduces the size of navigation tree, to minimize the effort required to reach the desired citation results for the given query. It minimizes the overall expected navigation cost, by hiding unimportant nodes leading to important nodes.

4. Algorithm For Calculation of Loss Function Using Simple Ordinal Classification (LF)

In our proposed algorithm loss function is defined between 0 and 1 that is $0 \leq LF \leq 1$, the minimum loss function is 0 and maximum loss function is 1. The resulting categories are examined for finding relevant documents related to the user query. We used S=2,00,000 session logs for testing our algorithm as the web contains millions of sessions, we cannot use all sessions at the same time. First we kept 2,00,000 sessions in a queue, as the processing of these sessions is completed; then we use another set two hundred thousand sessions. Billions of search queries posed to search engine which are mainly categorized into three types*: informational search queries based on finding possible information for the particular subject, Navigational search queries based on navigation particular website and transactional search queries based on to acquire something*.(C=3).

1. initialize S=200000
2. initialize C=3;
3. input the query(Q)
4. for i=1 to C do //C represents categories
5. for j=1 to S do// S represents no of sessions
6. for k= 1 to j do // for query in every session
7. Session_start()
8. if($_SESSION['hasVisited']="yes") as defined in equation (2)
9. Y[j][k]=1
10. otherwise
11. Y[j][k]=0
12. end if
13. end for
14. end for
15. $LF[i] = 1/n \sum_{i=1}^{n} (y_i - H(x_i))^2$ as defined in equation (3) // Loss function of query belong to specific category
16. end for
17. Finding minimum Loss function
18. min=LF[1]
19. for i=1 to C do
20. if(LF[i]<min)
21. min=CF[i]
22. end for
23. The given input query belongs to minimum loss function present in min.

5. Navigational B - Tree Model

Construction of initial active B-tree of order **four** with key values as **three** that is informational search queries, Navigational search queries, and transactional search queries. After finding the query belongs to, one of the three types of web search queries, next the user performs navigation in the B-tree. Navigation is performed on subtree of the root as we obtained using Loss function. The navigation of subtree involves the following actions.

1. Expansion (n): Expand the current node to reveal new set of concept nodes.

2. Displayresult (n): The users want to visit the list of titles attached to the sub tree.

3. Neglectresult (n): Title of the sub tree is unimportant and the user can ignore the sub tree and move onto another sub tree.

4. Retract (n): Perform undo of last edge removal operation.

5. Removeedge (n): Selectively reveals a subset of nodes out of large set of descendent nodes by removing unmatched child nodes.

The navigation of B-tree process continues until the user finds all citations with maximum relevancy and matches all key words in user query.

5.1 Algorithm for Dynamic Navigation of B-tree and Expansion of Nodes in B-tree:

Expansion (n)

1. if (n=root node)
2. m=Removedge(n)
3. for i=1 to m do
4. Expansion(i)
5. else if(n=leaf node)
6. select one of the following
7. i. Displayresult(n)
8. ii. Neglectresult(n)
9. else
10. select one of the following

11. i. Displayresult(n)

12. ii. Neglectresult(n)

13. iii. m=Removedge(n)
fori=1 to m do
Expansion (i)

While the user proffers a query, based on the keywords present in the query, using the calculation of loss function, we decide the query belongs to one of the three types of categories. Now employ dynamic construction of B-tree with keywords present in the query. During crawling of the web, it results in all titles that match to the keywords present in the query based on the count of word frequency present in the document. To reduce non-relevant titles, we apply first expansion of root node of B-tree. In this paper, we use B-tree to reduce the number of levels while revealing the children of the tree, so that navigation cost of the query will be O (*log n)*. The Root of the B-tree contains three types of navigational queries. Based on Loss function, first, we perform expansion of one of the three nodes present in the root. Thereafter by applying Remove edge process and based on a calculation of Loss function, we will skip children belongs to the remaining two types of queries, to reduce navigational cost and non-relevant results. Then, we perform matching of keywords present in the query with the children of the root node. Now, we calculate the loss function to check the keywords present in the query matched with children of B-tree. Based on the loss function value, we perform expansion of the child node with minimal loss function and remaining children of the B-tree are neglected. This process is continued until all relevant documents related to the query are retrieved and we process all child, sub-child and sub-child nodes, till we reach the leaf nodes of the B-tree.

6. Experimental Evaluation of Loss Function In A Session

We used LETOR4.0 data set from Microsoft Learn to Rank data sets with session log of users who accessed the LETOR data set [14]. Here we calculate the Loss Function for the given query using session logs collected from LETOR data set [14] and represent them here with their session ids and loss functions in each session. Likewise, we are represent a sample of thousand sessions as shown in Table1 for an example User query: *Information storage and retrieval in web mining".*

As presented in Table 1, the loss function is computed and presented for each category of queries. The results of loss function for all categories are presented against a range of sessions. Session 1 to 100 is considered as a range. In this fashion, the last range of sessions in the 1000 sessions considered is 901 to 1000. For the given user query, the loss function is the minimum for C1 that is 0.281. The maximum loss function is recorded for C3. Therefore, the navigation starts from C1 (informational queries). The descendants of C1 are activated and descendants of C2 and C3 are skipped. This process is repeated until we reach leaf nodes of B-tree and all keywords in the given query are processed. The loss function for all sessions includes 0.281, 0.338 and 0.497 for C1, C2 and C3 respectively. These values also show the same trends where C1 exhibits least loss function while C3 shows highest loss function.

7. Experimental Evaluation of Dynamic Navigation of B- Tree Using Simple Ordinal Classification

We built a web-based prototype application using PHP technology for proof of the concept. The application helps to have a number of user queries and observe the performance of the proposed system. We present an illustration on how our application renders desired functionality and performance expected. The user issues a query: "*Information storage and retrieval in web mining*", the keywords present in the query are *information*, *storage, retrieval*, *information storage*, *information retrieval* and *web mining*. Initially the number of resulting web pages is

SNO	SESSION-ID	C1[Loss Function]	C2[Loss Function]	C3[Loss Function]
1	S1-S100	0. 126	0.225	0. 460
2	S101-S200	0.242	0.526	0.562
3	S201-S300	0.295	0.236	0.456
4	S301-S400	0.258	0.407	0.489
5	S401-S500	0. 139	0.292	0.454
6	S501-S600	0.257	0.373	0.468
7	S601-S700	0.353	0.357	0.529
8	S701-S800	0. 193	0.394	0.483
9	S801-S900	0. 156	0.229	0.504
10	S901-S1000	0.275	0. 448	0.399
Loss Function of Sessions		0.281	0.338	0.497

Table 1. Calculation of Loss Function for thousand Sessions

145760. The initial navigation starts from the calculation of Loss function. For all the sessions, LF1=0.281, LF2=0.338, and LF3=0.497 are loss function values, so we navigate through *informational queries* and all child nodes of informational queries are activated and remaining queries child nodes are skipped. Now the descendants of informational queries about 120550 nodes are expanded whose B-tree main nodes are *information, web, and storage*.

The calculation of the loss function for child node (*information, web, storage)* of Informative queries are LF1=0.197, LF2=0.562, and LF3=0.387, so we navigate through *information* node and children of *web* and *storage* nodes are skipped, so the number of nodes to be revealed are reduced from 120550 to 97825 and navigational and transactional nodes whose children are not considered for expansion are skipped. After information, retrieval is expanded and the number of titles reduced is from 97825 to 85700. Later, the storage is expanded and the number of titles is reduced from 85700 to 65250. Then web mining is expanded and the numbers of titles is reduced from 65250 to 57650. Finally, 57650 are relevant documents out of the 145760 documents under informative queries. This process is repeated until all the keywords present in the query are covered and we reach the leaf node of B-tree.

8. Performance Analysis

We evaluated the performance of dynamic B-tree navigation in terms of the numbers of citations and average navigation cost using the number of levels of B-tree revealed. We compared our results with pairwise and Listwise [19] learn to rank approaches as shown in Table 2, where in the pairwise approach, it is very difficult to derive the final ranked list of documents and the Listwise approach removes all data if one or more values are missing. In the existing methods to retrieve the relevant documents, we should process all documents pairs, so the numbers of documents to process a query is high. In our proposed method, we reduced the number of documents by using the B-tree data structure.

As shown in Table 2, an excerpt from experimental results is presented for 10 queries. Each query has complexity with different number of keywords. The observations from those experiments include the number of documents retrieved with the proposed method, listwise and pairwise approaches. There are some trends observed in the results. The first trend is that the number of keywords has its impact on the quantity of resultant documents. The second trend is that for the same query, different techniques produced different number of documents. From this condition, it is clearly understood that the proposed method showed performance over its predecessors in terms of reducing the number of documents retrieved.

In this implementation, the user follows dynamic navigation of B-tree, where the user chooses to expand the right node in order to reveal resulting relevant documents. We compare the number of documents retrieved in our method with pairwise [15] and Listwise [19] learn to rank approaches used by most of the search engines. We observe that the number of documents retrieved in our B-tree method are less compared to pairwise [15] and Listwise [19] learn to rank approaches. Figure 4 compares the number of documents retrieved of these three methods. We observe that performance of our B-tree method is high compared to pairwise [15] and Listwise [19] learn to rank approaches.

As shown in Figure 4, it is evident that from the experimental results an excerpt with 1-10 user queries is

SNO	Query No	No of Keywords	Number of documents retrieved in Pair Wise	Number of documents retrieved in List Wise	Number of documents retrieved in our proposed method	Max height of B-tree/No of levels
1	Q1	6	75869	65523	58235	7
2	Q2	7	95567	86590	70562	8
3	Q3	8	89650	80780	75680	9
4	Q4	6	79655	70639	61560	7
5	Q5	6	75890	69476	60875	7
6	Q6	5	69498	60250	57650	6
7	Q7	4	40362	35200	25250	5
8	Q8	9	99568	90850	87623	9
9	Q9	4	56789	45620	23759	5
10	Q10	6	78520	65987	59870	7

Table 2. Comparison of Query Navigation of B-tree method with Pair Wise and List Wise Approaches

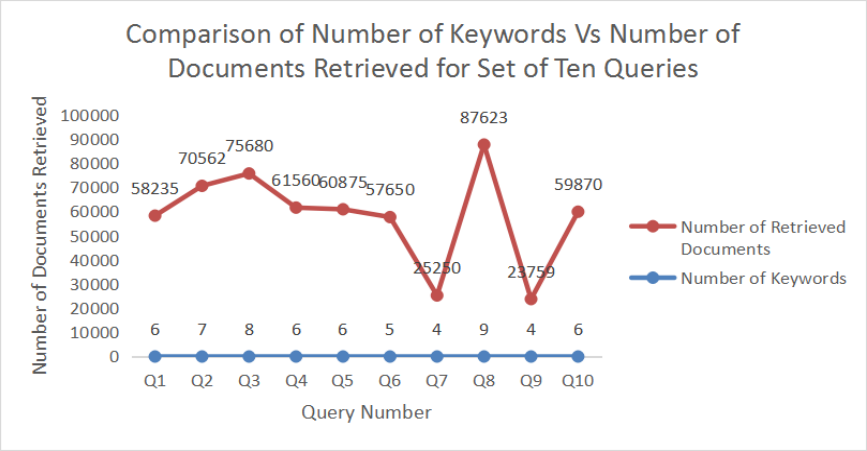

Figure 4. Comparison of Number of Keywords Vs Number of Documents Retrieved

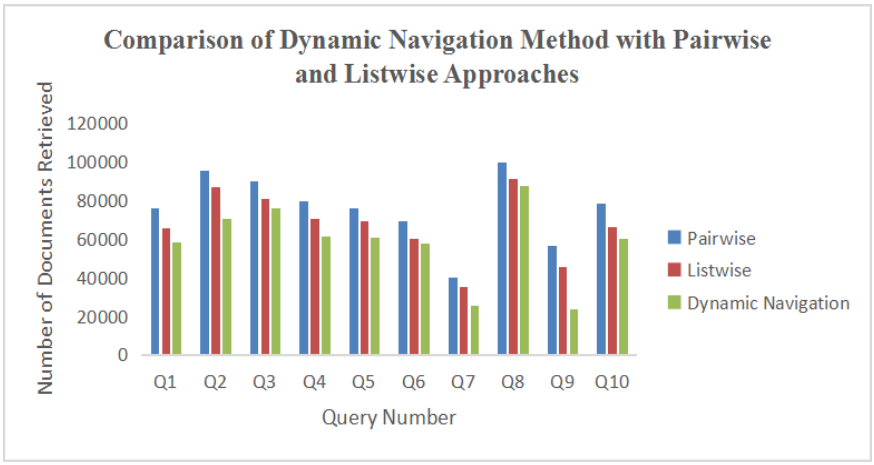

Figure 5. Comparison of Dynamic Navigation method with Pairwise and Listwise

considered and presented. Queries from 1 to 10 are presented in horizontal axis while the vertical axis shows the number of documents retrieved. The number of keywords and number of retrieved documents are the two data series presented. The minimum number of documents retrieved is 23750 against query 7, while the maximum number of documents retrieved is 87623 against the query 8. The results obtained by using dynamic navigation of B-tree reveal that we have reduced the navigation cost in terms number of levels to be processed while executing the query over data base. We retrieved most of relevant documents for the given query by reducing the non-relevant documents during the retrieval.

As shown in Figure 5, an excerpt of results for first 10 queries is presented. The proposed navigation method is compared against other existing methods like listwise and pairwise approaches. Consistently for all queries, the proposed method showed higher performance with less number of documents. It is based on the proposed algorithms used for calculation of loss function and dynamic navigation of B-tree employed as the part of the proposed SOC. Another observation is that the listwise approach performs better than that of pairwise approach. Nevertheless, both of them are inferior to the proposed approach.

As far as performance is concerned, the proposed B-tree based SOC with dynamic navigation showed better performance over the listwise and pairwise approaches. The rationale behind this condition is that the proposed approach is able to overcome the drawback of them using efficient and dynamic navigation. Particularly pairwise approaches are poor when there is uneven distribution of documents. On the other hand, high training complexity is the problem with listwise approaches. The proposed method employs regression with simple ordinal classification with two algorithms to reduce the loss function values in regression. Thus, the proposed method showed superior performance over the two existing methods.

9. Conclusion

Web structure mining plays an important role in retrieving information for users. Web query classification is very useful nowadays for search engines. In addition to document classification, it is important to retrieve more

relevant results to satisfy the users' requirements. The current algorithms are time consuming and vulnerable in steadiness and low efficiency. In this paper, we proposed a B-tree based web query classification algorithm using simple ordinal classification. Experimental results on real LETOR data set session log shows our approach for web query classification is proficient and provides more number of relevant web pages compared to the existing with pairwise [15] Listwise [19] and '*learn to rank*' approaches. The usage of B-tree reduces the number of levels during classification, as document retrieval is performed in depth first manner.

10. Future Work

Web document retrieval is a process which involves mainly, retrieval of relevant documents for user queries and matched results are sorted using page rank or '*learn to rank*' algorithms. In our proposed method, we implemented an efficient way of retrieving relevant documents for the user query, but we have only applied linear regression algorithm which produces relevant results but it also includes some of the non-relevant results. In our future work, we would like implement an efficient learn to rank algorithm as combined linear regression and gradient descent algorithm to sort the results obtained through dynamic classification of web queries using session log and loss function to increase more relevant results and to reduce non-relevant documents.

Acknowledgment

This research was supported by the Department of Science and Technology under WOS-A (SR/WOS-A/ET-1071/2014). I would like to thank my supervisor Dr P Bhaskara Reddy for his support and help through the year. Finally, I would like to thank our colleagues from MLRIT who provided insight and expertise that greatly assisted the research.

References

[1] Jadav, Jigar., Burke., Andrew., Dhiman, Pratik. (2017). Classification of student web queries, *In*: IEEE Computer and Communication Workshop and Conference.

[2] Yao, Yazhou., Zhang., Jian., Shen, Fumin., Hua, Xian-Sheng., Xu, Jingsong., Tang, Zhenmin. (2017). Exploiting Web Images for Dataset Construction: A Domain Robust Approach, *IEEE Transaction on Mutlimedia.*

[3] Amir, Samir., Aýt-Kaci, Hassan. (2016). An efficient large scale reasoning method for the semantic web, *Journal of Intelligent Information Systems*, 46 (135).

[4] Zhang, Wei-Nan., Ming, Zhao-Yan., Zhang, Yu., Liu, Ting., Chua, Tat-Seng (2016). Capturing the Semantics of Key Phrases Using Multiple Languages for Question Retrieval, *IEEE Transactions on Knowledge and Data* Engineering, 28 (4).

[5] Figueroa, Alejandro., Atkinson, John . (2016). Ensembling Classifiers for Detecting User Intentions behind Web Queries, *IEEE Internet Computing*, 20 (2).

[6] Waller, Vivienne. (2016). Making knowledge machine - processable: some implications of general semantic search, *Taylor & Francis Behavior and Information Technology,* 35 (10).

[7]Xia,Chunwei.,Wang, Xin.(2015).Graph-Based Web Query Classification *In*: IEEE Conference Publications.

[8] Baykan, Eda., Henzinger, Monika., Weber, Ingmar. (2013). A Comprehensive Study of Techniques for URL-Based Web Page Language Classification, *ACM Transactions on the Web*, 7 (1).

[9] Li, L., Zhong, L., Xu, G., Kitsuregawa, M. (2012). A feature free search query classification approach using semantic distance, *Expert Systems with Application*, 39 (12).

[10] Navigli, R., Ponzetto, S.P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, *Artificial Intelligence*, 193 (1).

[11] Kashyap, Abhijith., Hristidis, Vagelis., Petropoulos, Michalis, Tavoulari, Sotiria. (2011). Effective Navigation of Query Results Based on Concept Hierarchies, *IEEE Transactions on Knowledge and Data Engineering*, 23 (10).

[12] Hang, L.I. (2011). A Short Introduction to Learning to Rank, *EICE Transactions on Information and Systems*, 94 (10).

[13] Lovelyn Rose, S., Chandran, K.R., Nithya, M. (2011). An Efficient Approach to Web Query Classification using State Space Trees, *IJCST*, 2 (2).

[14] Sculley, D. (2010). Combined Regression and Ranking, *In*: KDD '10 Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining.

[15] Sundararajan, S. (2011). A Pairwise Ranking Based Approach to Learning with Positive and Unlabeled Examples, *In*: CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management.

[16] Diemert, E., Vandelle, G. (2009). Unsupervised query categorization using automatically-built concept graphs, *In*: Proceedings of the 18th international conference on World Wide Web.

[17] Lashkari, Arash Habibi. (2009). A Boolean Model in Information Retrieval for Search Engines, *In*: IEEE International Conference on Information Management and Engineering.

[18] Barr, Cory., Jones., Rosie., Regelson, Moira. (2008). "The Linguistic Structure of English Web-Search Queries, *In*: Proceedings of the Conference on Empirical Methods in Natural Language Processing.

[19] Cao, Zhe. (2007). Learning to rank: from pairwise approach to listwise approach, *In*: ICML '07 Proceedings of the 24th international conference on Machine learning.

[20] Shen, Dou., Pan., Rong., Sun, Jian-Tao., Jeffrey Pan, Junfeng. (2006). Query enrichment for web-query classification, *ACM Transactions on Information Systems,* 24 (3).

[21] Beitzel, S. (2005). Automatic web Query Classification Using Labeled and Unlabeled Training data, *In*: SIGIR '05 Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval.

[22] Daniel, E., Rose, Danny Levinson. (2004). Understanding user goals in web search", in World Wide Web Conference.

[23] Andrew MacFarlane. (2004). Introduction to Modern Information Retrieval. (2nd edition) 38 (3).

[24] Kang, In-Ho. (2003). Query Type Classification for Web document Retrieval, SIGIr'03, ACM.

[25] Frank, Eibe., Mark Hall. (2001). A Simple Approach to Ordinal Classification, *In*: EMCL '01 Proceedings of the 12th European Conference on Machine Learning, *2167*.

[26] Herbrich, R., Graepel, T., Obermayer, K. (2000). Large Margin rank boundaries for ordinal regression, *In*: Advances in neural information processing systems.