# CHAPTER 3

**B-TREE BASED CLASSIFICATION OF WEB QUERIES**

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Name of the sub-title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. 1</td>
<td>Fundamental Definitions</td>
<td>41</td>
</tr>
<tr>
<td>3. 2</td>
<td>Overview of the proposed system</td>
<td>43</td>
</tr>
<tr>
<td>3. 3</td>
<td>Calculation of Loss Function</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>3.3.1 Algorithm for Computing Loss Function</td>
<td>46</td>
</tr>
<tr>
<td>3. 4</td>
<td>Navigational B-tree Model</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>3.4.1 Algorithm for Dynamic Navigation of B-tree</td>
<td>48</td>
</tr>
<tr>
<td>3. 5</td>
<td>Experimental Evaluation of Loss Function</td>
<td>50</td>
</tr>
<tr>
<td>3. 6</td>
<td>Experimental Evaluation of Dynamic Navigation of B-tree</td>
<td>52</td>
</tr>
<tr>
<td>3. 7</td>
<td>Performance Analysis</td>
<td>54</td>
</tr>
<tr>
<td>3. 8</td>
<td>Summary</td>
<td>58</td>
</tr>
</tbody>
</table>
CHAPTER 3
B-TREE BASED CLASSIFICATION OF WEB QUERIES

Web queries submitted by users to search engines mainly contain ambiguity, conciseness and are subject to change over time. Most of the users are unable to formulate the query appropriately. Search engines produce the results by analyzing the keywords present in the given query. For analyzing the keywords present in the given query an efficient query processing system is required. Nowadays most of the search engines use classification of web queries for analyzing the keywords present in the query and navigating to the corresponding web pages as a result.

In this chapter, a classification method named Simple Ordinal Classification (SOC) is used for classification of web queries. A B-tree based navigation is used for efficient classification of web queries. It provides navigation of search keywords dynamically by considering session time of users who used search engines. The SOC method is suitable for reducing irrelevant document retrieval and navigation cost. This method retrieves the documents that match with all keywords present in the given query. The subsequent sections of this chapter are structured as follows. Firstly this chapter offers some formal definitions followed by an overview of the proposed system. After that, it presents the calculation of loss function using simple ordinal classification algorithm. Then the navigational B-tree model is proposed for dynamic navigation of web queries. Then it performs the experimental evaluation of loss function using sessions. After that, it presents an empirical evaluation of dynamic navigation B-tree model and performance of proposed B-tree model compared to listwise [132] and pairwise [15] approach.

3.1 FUNDAMENTAL DEFINITIONS

Before introducing proposed methodology and algorithms, here are the definitions used in this chapter. These definitions help in understanding the proposed system easily.

Definition 1. (Session): For web applications, a session is a series of requests that come from a particular browser (user) from a sign in to sing out time. The session provides a set
of observations that are made by the end user. Generally, in a session, a user may search for similar information and vocabulary in a given session.

**Definition 2. (Title):** It refers to the name of the given URL.

**Definition 3. (Clicked Title):** It refers to the URL that has been clicked by the user in a given session. Thus the number of titles clicked by the user during a session assumes significance in this research.

**Definition 4. (Unclicked Title):** It refers to the fact that user has searched for information. Any title that is not clicked by the user is called unclicked title.

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

\[
LF(X,Y) = PS \sum_{x \sim D} [H(x) \neq y_i]^2
\]  

(3.1)

Where \(X\) represents a set of query-document pair vector

\(Y\) denotes relevancy vector,

\(PS\) indicates predictive success function in which

\(D\) means uniform distribution

\(H(x)\) denotes the output of classifier algorithm

Query belongs to the specific category, and that is determined by computing loss function based on ordinal classification function associated with a user session.

\[
Y_i = \begin{cases} 
1 & \text{Title Clicked by user} \\
0 & \text{Title Unclicked by user} 
\end{cases}
\]

\(N\) denotes the number of documents, and the loss function is computed as follows.

\[
LF = \frac{1}{n} \sum_{i=1}^{n} (Y_i - H(x_i))^2
\]  

(3.2)

If there is a single document, the loss function can be computed using Eq. 3.1. Here \(Y_i\) is the number of titles clicked by the end user. From the standpoint of Eq. 3.1, the \(H(x)\) denotes output classifier.
Definition 6. (Navigation B-tree): A B-Tree meant for effective navigation is denoted by \( T(\{V, E, C\}) \). The tree has \( V \) nodes, \( C \) classification and \( E \) links and degree of \( C \) which indicates as query categories. The construction of B-Tree can start with root node denoted as \( C \), and the tree is built iteratively in the pre-order traversal.

Definition 7. (Remove edge): It is a process used to remove edges that are not relevant and do not depend on keywords in the query dynamically. Figure 3.3 illustrates the remove edge process.

3.2 OVERVIEW OF THE PROPOSED SYSTEM

This section provides a methodology for web query classification as a part of effective navigation of query results. It provides an architecture and describes its components to understand the approach used in a proposed system. Overview of the proposed methodology is shown in Figure 3.1. The datasets used for the experimental evaluation are taken from Learn to Rank (LETOR) corpora [136]. The dataset contains queries, URLs, features, training, testing, and validation of the queries. The training data is associated with various user sessions in the form of logs. Sessions contain different queries posed by the user. The proposed methodology considers queries in three categories. Simple Ordinal Classification (SOC) algorithm is used for classification of the user queries into the corresponding categories.

SOC algorithm has two-fold functionality. One is the computation of loss function, and another is the classification of queries using loss function. The primary objective of proposed SOC algorithm is to minimize the loss function. Determination of query belongs to a specific category using computation of loss function. The loss function is calculated by using session logs. B-tree is used to represent categories, that exhibit minimal loss function is used for retrieving documents. As the tree is constructed, edge removal function is employed based on the phrases that match given query.
The proposed architecture for dynamic navigation system proceeds with web query specified by the user to perform its functions and gives results to end user. The training data is taken from LETOR dataset [136] which contains session logs. The training process uses q queries, x documents, and y relevant documents denoted by I(q, x, y). SVM classification and the regression model is also used to determine the query to belong to a specific category. All relevant documents are said to have $y_i=1$ while irrelevant ones are supposed to have $y_i=0$. For a given set of documents, the loss function is computed as the number of matched documents that are subtracted from the classification of x documents.

When support vector model is used, it alone cannot produce effective results. Therefore loss function with support vector model is defined as the sum of squared errors in standard information retrieval practices. Standard IR metrics such as Mean Reciprocal Rank, Mean Average Precision, and Normalized Discounted Cumulative Gain can be used to evaluate the performance of SOC algorithm. It employs an iterative model that tries to minimize loss function and the overall sum of squared errors. With the proposed approach the time

**Figure 3.1:** Architecture of proposed dynamic navigation system
complexity for retrieval of relevant documents is $< O(\log n)$, and it minimizes retrieval of irrelevant documents.

The construction of B-tree for the query “web mining” is shown in Figure 3.2. “Web mining” belongs to a category of informational queries. The tree contains nodes at the top such as informative, navigational and transactional as three categories of web queries. The navigation process starts with the informative category. When a simple decision tree is constructed, its depth is more and space and time complexity are high. Therefore the construction of B-tree is considered with the hierarchy of nodes that support efficient navigation. Thus the size of the tree is reduced besides reducing the number of levels in the tree as it a height balanced tree. Non-relevant edges while navigating B-tree are removed. This phenomenon is illustrated in Figure 3.3.

![Figure 3.2: Construction of B-tree for given query](image)

![Figure 3.3: Navigation of B-tree dynamically](image)

The root node of B-tree contains three categories of queries. The query given by the user includes keywords or phrases. When loss function is computed to the query, it may belong to one of the categories such as informative, navigational or transactional. Other
edges that are non-relevant are removed. For instance, when "buy a phone" is the query issued by the end user, and it probably belongs to transactional query. Therefore, it is not required to have documents of informational and navigational. Here is the need to remove edges that are related to informational and navigational queries. In the list of results, these categories are omitted, and dotted lines represent the removal of edges. When edges are removed purposefully, the size of the tree constructed is reduced. Thereby it reduces the effort needed to navigate the results for a given query. Thus the overall expected navigation cost is reduced as well as removal of nodes that are not relevant.

### 3.3 CALCULATION OF LOSS FUNCTION (LF)

As per the loss function definition in this chapter, the computation of loss function returns a value between 0 and 1 \[ 0 \leq LF \leq 1 \]. The minimum loss function is denoted as 0 while the maximum is indicated as 1. For the given user query, based on the loss function value, the resulting categories are examined to extract relevant documents. Here S denotes session logs, and it initially has 2,00,000 items. These session logs are used to evaluate the proposed algorithm. As the processing of these sessions has been completed, then another 2,00,000 sessions present in the queue are to process. Thus incrementally all sessions are used for evaluation of loss function in a piecemeal approach.

Billions of queries posed by end users against search engines over WWW belong to three categories of the queries as mentioned earlier. Informational queries find possible information on the given subject. Navigational search queries are related to getting web pages to which user can navigate while transactional queries are used to acquire transactional web page links.

#### 3.3.1 ALGORITHM FOR COMPUTING LOSS FUNCTION

This subsection provides pseudo code that is used to compute loss function. C represents number of categories while S represents sessions. In the same fashion, Q denotes set of queries posed by the end users.

**Algorithm: Computation of Loss function**
**Input:** Read the number of sessions and queries in each session.

**Output:** Average Loss function for all queries in all sessions for each category.

1. start
2. initialize S=200000
3. initialize C=3
4. read the query(Q)
5. for i=1 to C do //C represents categories
6. for j=1 to S do // S represents no of sessions
7. for k= 1 to  j do // for query in every session
8. Start the session
9. if($_SESSION['hasVisited']="yes")
   Y[j][k]=1
10. else
11. Y[j][k]=0
12. end if
13. end for
14. end for
15. end for
16. LF[i] = \( \frac{1}{n} \sum_{i=1}^{n} (y_i - H(x_i))^2 \) as defined in equation (3.2)
   // Loss function of query belong to specific category
17. end for
18. To find minimum Loss function initialize min=LF[1]
19. for i=1 to C do
20. if(LF[i]<min)
21. min=CF[i]
22. end if
23. end for
24. print query belongs to minimum loss function present in min.
25. stop

**Algorithm 3.1:** Loss function calculation

As shown in Algorithm 3.1, the loss function computation algorithm takes 2, 00, 000 sessions, and the number of categories is 3. Each session is examined for each category to know session dynamics concerning URLs visited or not. The loss function is computed as defined in Eq. 3.2. This process is applied iteratively to categorize web queries based on the loss function value. Finally, the categories that exhibit minimum loss function are revealed, and remaining categories are removed.
3.4 NAVIGATIONAL B-TREE MODEL

This section describes the process of constructing B-tree for effective navigation. Here B-tree of order four is used, and then the three key values are information search queries, transactional search queries, and navigational search queries. Once the query is identified belonging to a particular category, it is vital to perform navigation in the B-tree. The general actions involved in the navigation of a B-tree as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand(n)</td>
<td>It is for expanding a current node to view subset concept nodes.</td>
</tr>
<tr>
<td>Display result(n)</td>
<td>It reveals a list of titles associated with a subtree to end users.</td>
</tr>
<tr>
<td>Neglect result(n)</td>
<td>It is for ignoring a title associated with a subtree and move on to the next one.</td>
</tr>
<tr>
<td>Retract(n)</td>
<td>It undoes operation on the last operation related to edge removal.</td>
</tr>
<tr>
<td>Remove edge(n)</td>
<td>It reveals a subset of nodes selectively from the large set of child nodes after removing child nodes that are not matched.</td>
</tr>
</tbody>
</table>

Table 3.1: List of operations associated with B-tree navigation

These actions are part of B-tree navigation to have effective navigation and information retrieval. The navigation process continues until documents are retrieved with maximum relevancy that reflects user intent based on all the keywords present in the query. The construction of B-tree makes the difference in reducing time and space complexity. An algorithm is defined for dynamic navigation of B-tree and performs operations dynamically.

3.4.1 ALGORITHM FOR DYNAMIC NAVIGATION OF B-TREE

This algorithm is used to achieve dynamic navigation of B-tree and to perform various operations dynamically on B-tree. It ensures effective navigation of web queries.
**Algorithm 3.2: Dynamic navigation of B-tree**

As shown in Algorithm 3.2, it is evident that the algorithm takes a node and performs expansion activity. It follows the mechanisms to verify whether the given node is root node or leaf node. If the node is a root node, based on the loss function value, it performs expansion of one of the three key nodes, and remaining nodes are removed as they are non-relevant. If the node is a leaf node, the algorithm performs one of the activities such as display result or neglect results. If the node is an intermediate node it displays all nodes present under it, again it calculates loss function to expand, to remove or to display the resulting nodes. The expansion operation is an iterative process.

When the user issues a query, the query is interpreted. Based on the keywords available in the query loss function is computed and the decision is made to label given query into one of the categories such as navigation, informational and transactional. Afterwards, B-
tree is constructed dynamically. While crawling in the web, all title that matches with query keywords is returned. It is performs based on keyword frequency in the web documents. The first expansion of the root node is performed for reducing non-relevant titles. B-tree is used explicitly for reducing the number of levels when children of the tree are revealed. Thus the navigation cost when a query is issued is $O(\log n)$. The root node of B-tree has three types of queries. Based on the loss function, which kind of category the given query belongs to is determined and expansion process takes place.

Afterwards, by applying remove edge process and based on the value of loss function the child nodes to which query does not belong to are omitted. The remove edge process can efficiently reduce the navigation cost. Afterwards, the matching of keywords presents in the query with the child nodes of the root. If the query keywords are relevant and matched with the child node, then perform displaying the resulting nodes present under a child node or neglect the node or expand the node. If the child nodes are to be displayed, the display result activity is performed. If the child nodes are not required, it shows another operation named neglect children. This process is iteratively carried out until all documents related to the query are retrieved by matching all the keywords.

3.5 EXPERIMENTAL EVALUATION OF LOSS FUNCTION

To evaluate the proposed methodology with the underlying algorithms, the dataset used is LETOR 4.0 from Microsoft [136]. It uses session logs of users of LETOR dataset. For every given query, the loss function is computed with the help of session log information. Sessions are uniquely identified with an ID; loss function for each session is calculated. Though the dataset has thousands of session logs, 1000 sessions and their loss function dynamics for three categories are presented in Table 3.2. The results are related to user query "information storage and retrieval in web mining."
The evaluation of loss function for a set of 1000 sessions is shown in Table 3.2. The results are captured in a range of sessions starting from S1 to S1000. The first range is S1-S100, second is S101-S200 and so on until it reaches S901-S1000. Loss function C1 for given query is 0.126. C3 exhibits highest loss function that is 0.529. C1 represents informational queries while C2 and C3 represent navigational and transactional queries respectively. Based on the loss function the descendants of C2 and C3 are ignored while the descendants of C1 are expanded. It is repeatedly executed until leaf nodes are reached in the B-tree. The average loss function values for all sessions include 0.281 for C1, 0.338 for C2 and 0.497 for C3. The experimental results show that loss function for C1 is less compared remaining two categories C2 and C3. So the given query belongs to a C1 category.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1-S100</td>
<td>0.126</td>
<td>0.225</td>
<td>0.460</td>
</tr>
<tr>
<td>2</td>
<td>S101-S200</td>
<td>0.242</td>
<td>0.526</td>
<td>0.562</td>
</tr>
<tr>
<td>3</td>
<td>S201-S300</td>
<td>0.295</td>
<td>0.236</td>
<td>0.456</td>
</tr>
<tr>
<td>4</td>
<td>S301-S400</td>
<td>0.258</td>
<td>0.407</td>
<td>0.489</td>
</tr>
<tr>
<td>5</td>
<td>S401-S500</td>
<td>0.139</td>
<td>0.292</td>
<td>0.454</td>
</tr>
<tr>
<td>6</td>
<td>S501-S600</td>
<td>0.257</td>
<td>0.373</td>
<td>0.468</td>
</tr>
<tr>
<td>7</td>
<td>S601-S700</td>
<td>0.353</td>
<td>0.357</td>
<td>0.529</td>
</tr>
<tr>
<td>8</td>
<td>S701-S800</td>
<td>0.193</td>
<td>0.394</td>
<td>0.483</td>
</tr>
<tr>
<td>9</td>
<td>S801-S900</td>
<td>0.156</td>
<td>0.229</td>
<td>0.504</td>
</tr>
<tr>
<td>10</td>
<td>S901-S1000</td>
<td>0.275</td>
<td>0.448</td>
<td>0.399</td>
</tr>
<tr>
<td></td>
<td>Loss Function of Sessions</td>
<td>0.281</td>
<td>0.338</td>
<td>0.497</td>
</tr>
</tbody>
</table>

Table 3.2: Calculation of Loss Function for thousand Sessions
3.6 EXPERIMENTAL EVALUATION OF DYNAMIC NAVIGATION OF B-TREE

A web-based prototype is built using PHP for demonstrating proof of the concept. The application is designed to simulate any number of user queries. It helps us to test the performance of the algorithms by sending any number of user queries. Here is an illustration that shows the execution of a query and the evaluation of results. Assuming that user has issued a query "Information storage and retrieval in web mining," the keywords are identified. They include information, storage, retrieval and web mining. The number of resultant web pages of the query is 145760. Average LF for C1, C2, and C3 are 0.232, 0.356 and 0.984 respectively. When navigated through informational queries, the child nodes related to informational queries are shown while child nodes linked to other queries are ignored. Thus the resultant descendants of informational queries are 12055 nodes that are to be expanded. Figure 3.4 shows the user interface of the web-based application for demonstrating dynamic navigation of queries using B-tree and simple ordinal classification.

![Dynamic Navigation of Queries using B-tree and categorization factor](image)

**Figure 3.4:** Prototype application showing dynamic tree navigation using B-tree
For child nodes associated with information, web, and storage, the values of LF are 0.197, 0.562 and 0.387 respectively for C1, C2, and C3. Navigation is done through information node, and child nodes such as a web and storage are skipped. Therefore the number of resultant nodes is reduced to 97825 from 120550. At the same time, navigational and transactional nodes for which children are not taken into consideration for expansion. Once information node is expanded, then the child node retrieval with less loss function is expanded. By this time number of resulting nodes is reduced to 85700 from 97825. When storage node is expanded, the number of resulting nodes is further decreased to 65250 from 85700. Final result contains 57650 documents that are related to the given query. For the same, under informative queries, the result was initially 145760. This iterative process is carried out until all keywords that are part of given query are covered, and leaf nodes of B-tree are reached.

**Figure 3.5: Expansion of B-tree nodes**
As shown in Figure 3.5, it is evident that the proposed web-based interface for dynamic navigation of queries using B-tree and simple ordinal classification produces high-quality results by reducing non-relevant documents. By showing only the category to which a query belongs to is based on the loss function computed, space and time complexity besides navigation cost are reduced in the proposed method.

3.7 PERFORMANCE ANALYSIS

This section provides the performance evaluation of dynamic navigation of query results. Primarily, it presents the navigation of B-tree for evaluating the number of citations, average cost needed for retrieving relevant documents of a given query. The results obtained in our experiments are compared with a pairwise RV [15] approach and listwise ES-RANK [132] approach. It is complicated to process all pairs of documents in pairwise [15] approach, so produces more documents. If one or more values are missing in listwise approach [132], then it removes documents even they are relevant. So it provides more non-relevant documents. For existing methods, all documents are to be processed to have relevant documents. Thus the computational complexity is high. In the proposed method it is reduced by using B-tree based approach for processing.

<table>
<thead>
<tr>
<th>SNO</th>
<th>Number of Queries</th>
<th>Average no of Keywords (Ascending order)</th>
<th>Average number of documents retrieved in RV method [15]</th>
<th>Average number of documents retrieved in ListWise ES-RANK [132]</th>
<th>Average number of documents retrieved in our proposed method</th>
<th>Max height of B-tree/No of levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2</td>
<td>99568</td>
<td>90850</td>
<td>87623</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td></td>
<td>175866</td>
<td>165525</td>
<td>145823</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td></td>
<td>325563</td>
<td>290567</td>
<td>272545</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>5</td>
<td>565340</td>
<td>452364</td>
<td>356430</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
<td>79498</td>
<td>70250</td>
<td>67650</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td></td>
<td>150546</td>
<td>140780</td>
<td>128560</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td></td>
<td>302567</td>
<td>275789</td>
<td>250568</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td></td>
<td>495852</td>
<td>395780</td>
<td>306532</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 3.3: Comparison of Query Navigation of B-tree method with pairwise and List Wise Approaches

Table 3.3 shows a portion of experimental results. It shows different queries made by users, the number of keywords present in the query, number of documents retrieved by the pairwise [15] method, number of documents retrieved by the listwise [132] method and the number of documents retrieved by the proposed method. It also presents the number of levels in the B-tree or maximum height of B-tree. The minimum number of keywords present in the queries is two, and the maximum number of keywords is twenty. First 10 queries has two keywords in it. In this case, the maximum height of the B-tree generated is ten. The number of documents retrieved by the pairwise [15] method is 99568, the number of documents by the listwise [132] method is 90850, and the proposed method is 87623. From this, it can be observed that the proposed method exhibits high performance as it reduces the number of irrelevant documents proficiently. The Listwise [132] approach is better than the pairwise [15] while the proposed method is better than both listwise [132] and pairwise [15] approaches.
The last 1000 queries in the Table 3.3 has number of keywords as 20. Its B-tree height is eight to nine. It has got 75685 documents retrieved by the pairwise [15] method, 65368 by the listwise [132] method and 40560 by the proposed method. It also reveals the performance improvement exhibited by the proposed method. From the results, it is observed that the number of keywords in the query has its impact on the number of documents retrieved. The proposed method showed performance improvement over the other two methods for every user query.

The rationale behind this is that the utility of B-tree for dynamic navigation of query results. The listwise [132] and the pairwise [15] learn to rank approaches widely used by many search engines. From the results, it can be observed that B-tree method yields less number of resulting documents when compared with that of listwise [132] and pairwise [15] approaches. Figure 3.6 visualizes the performance of the three approaches in the retrieval of relevant documents against user query by varying number of keywords.

![Comparison of Dynamic Navigation Method with Pairwise and Listwise Approaches](image1)

![Comparison of Dynamic Navigation Method with Pairwise and Listwise Approaches](image2)

**Figure 3.6:** Comparison of number of keywords, queries executed Vs documents retrieved
Figure 3.6 shows the performance evaluation of our proposed method with listwise and pairwise approaches. The horizontal axis shows queries ranging from 1 to Q1000 while the vertical axis indicates the number of documents retrieved. Two data series are plotted in the graph. Against maximum number of keywords least number of documents is retrieved that is 20582 documents for ten queries. In this case, a maximum number of documents is shown in results that are 55560 for 1000 queries. Dynamic navigation of B-tree resulted in the reduction of the number of documents (removes documents that are not related). The usage of B-tree reduced space and time complexity. Thus the navigation cost is reduced as the number of levels in the tree is reduced. It also has its effect on the navigation cost and the time taken for producing such results as well.

![Comparison of Dynamic Navigation Method with Pairwise and Listwise Approaches](image1)

**Figure 3.7:** Comparison of number of keywords, queries executed Vs documents retrieved

Figure 3.7 shows the performance of the proposed method when compared with the listwise [132] and pairwise [15] method. From the results of experiments, an excerpt with thousand user queries ranging from 1 to 1000 is presented by varying number of keywords from 2 to 20. It is observed that for all queries list wise method shows better performance. The least performance is exhibited by the pairwise [15] method for all queries. The
proposed dynamic navigation with B-tree shows superior performance over its two counterparts such as listwise and pairwise methods. The proposed method produced the least number of documents when compared with other two methods. The proposed method employs query classification method known as SOC besides using B-tree for dynamic navigation of query results.

![Comparison of Dynamic Navigation Method with Pairwise and Listwise Approaches](image)

**Figure 3.8**: Performance comparison of pairwise, list wise and proposed methods

The Figure 3.8 represents 1000 queries were executed with 20 keywords. The proposed algorithms and the utility of B-tree improve the performance in retrieving the documents. The rationale behind this is that ability of the methods to reduce the number of irrelevant documents. Nevertheless, the proposed method with dynamic tree navigation generated the least number of documents and reduced more irrelevant records than the other two methods. The proposed simple ordinal classification method and regression show better performance over other two approaches in document retrieval as part of information retrieval process.

**3.8 SUMMARY**

Web document retrieval is a process which mainly involves two steps i) Retrieval of relevant documents to the user query and ii) Retrieved documents are sorted using Page Rank or learn to rank algorithms.
For retrieval relevant documents to the user query, this chapter has covered the proposed B-tree based approach for dynamic query navigation with underlying simple ordinal classification method for web query classification. It is understood that web queries made by the user can have ambiguity, conciseness and a kind of inconsistency in query formation. It is not possible to expect similar expertise from users to form queries. Therefore, it is the responsibility of applications or search engines that take and process such queries to have mechanisms to understand user intent. Web query classification is the method used to solve the problem. Towards this end, the proposed method considered three categories of queries. They are known as informational queries, navigational queries, and transactional queries.

When a query is given by end user with specific keywords, the query is analyzed by the proposed system, and loss function is computed. Based on the value of loss function, the category to which the query belongs is determined. The loss function is very important in reducing the number of documents retrieved in the results. If the query belongs to informational, the navigational and transactional results are discarded. User satisfaction is thus achieved as the query results reflect user intent as much as possible. The existing algorithms that are the listwise and the pairwise methods showed less performance. LETOR dataset [136] is used to demonstrate the performance of the proposed method. LETOR [136] session logs are used for experimental evaluation. From results, it is understood that the web query classification method along with B-tree used in the proposed method is capable of reducing irrelevant documents. Thus the objective of effective query results is achieved with dynamic navigation of query results with B-tree. Besides the B-tree based approach reduced navigation cost and improved the performance of query results. The next chapter throws light into improved distance Page Rank algorithm and evaluation Page Rank for web pages based on user feedback.

Once the relevant documents are retrieved using the proposed method, and to arrange these documents from highly relevant to least relevant, Page Rank approaches are used. The next chapter proposes an improved distance Page Rank algorithm to rank the retrieved documents.