CHAPTER 5

QUERY PERSONALIZATION

5.1 INTRODUCTION

Personalization is the combination of technology and customer loyalty to tailor online transactions between providers and individual customers. Advanced personalized E-Commerce applications require comprehensive knowledge about user’s desire in order to customise individual products to the users. An expressive and well formed logical framework for explicit preference learning is discussed in Chapter 4 and it is suitable for other complex domain also. Standard SQL query language is extended with such preferences to deal user wishes efficiently. Besides qualitative, quantitative preferences are learned in Chapter 4 and these simple preferences are composited to deduce multi granular preferences. This chapter summarises the concept of query personalization, its general working principle. In addition, existing personalization algorithms are compared with proposed WETPER algorithm.

Search engines handle the exponential growth of web with sophisticated indexing algorithms. Even though, it is difficult to meet the relevant information with minimum search steps along higher effectiveness, personalization is one of the techniques which applies user modeling component and can affect the retrieval process at three different stages. The personalization process can be implemented in the following sections.
1. user profile utilization
2. query modification
3. result ranking.

Implementation of personalization process in case of user profile utilization is difficult, because the search system needs to generate personalized results satisfying more users with limited time constraint. Hence, the personalization approach is followed mostly in query modification and result ranking process and also it is designed in client side. The involvement of personalization in information system at different levels is represented in Figure 5.1.

![Figure 5.1 Personalization in Information system](image)

Figure 5.1 Personalization in Information system

The query personalization problem can be defined as an optimization problem in which user profile is utilized to generate personalized results and ranking of the personalized results need to be done. This problem
statements sometimes leads to unrealistic solution since utilization of all preferences for personalized query construction and answer generation makes the work more complex. Hence, the personalization problem can be defined as a constrained optimization problem where the upper bound restricts the number of preferences used for personalization process that denoted as k and the lower bound decides the minimum number of preferences need to be satisfied by the personalized answer that denoted as l in personalized answer generation. In addition, the personalization time and distance between the generated personalized answer and actual personalized answers in user perception can be defined as major constraints in constrained personalization problem.

This chapter presents the existing work of (Koutrika & Ioannidis 2010) and the proposed WETPER query personalization algorithm. The role of preference finder is to find related preferences from user profile and personalized answer generation of WETPER is discussed. Preferences are atomic and join conditions in user profile and preference finder finds related composite preferences.

5.2 GENERAL PERSONALIZATION ALGORITHM

The personalization process starts with user profile which is the repository of explicit or implicit preferences which are gathered through questionnaires or from historical data. The personalization algorithm finds most related preferences and enhances them with user query and the enhancement process is denoted as query enrichment process or integration process. These enhanced queries are called preference queries. The personalization logic extracts ranked personalized results from actual results. These steps are given as a general personalization algorithm as shown below.
Input: User profile U, User query Q, Interest Criterion IC

Output: Personalized results

Begin

get user query

If (preferences from U related to Q)

construct top k preferences

organize them based on IC

execute query over datasource

personalized_result= \{results that satisfy 1 out of top k preferences\}

rank personalized result

End

5.3 EXISTING QUERY PERSONALIZATION ALGORITHMS

The existing query PS (Koutrika & Ioannidis 2010) builds user profile by gathering preferences through an interface. The user can enter any number of preferences with doi. Then the preference selection step extracts top k related preferences from the user profile. The system considers preference selection problem as a graph computation problem in which user profile is structured as personalization graph (Koutrika & Ioannidis 2004) and it is given in Figure 5.2 and the query is represented as preference graph and it is given in Figure 5.3. In personalization graph rectangles represent relations and ovals represent attributes and cards represent attribute values. The personalization graph denotes d1, d2 are favourite directors and g1,g2 are favourite genres and a1 and a2 are favourite actors of the user.
The preference graph in Figure 5.3 is a rooted directed graph (Vp,Ep) where the root stands for movies which needs to be retrieved and the Vp stands for relation or attribute name or values, and Ep stands for the connection between relation and attributes. The syntactically equivalent and not conflicting preferences with user query are considered as top k related preferences. These preferences are organized as preference networks.

5.3.1 Preference Networks

The top k related preferences from user profile are integrated with user query and personalization logic is applied to generate personalized results. Two variables l and k in personalization logic depends on the
closeness of the query with preferences. Hence, these values are restricted by
the system or by the user.

The preference network is constructed based on the principle
behind query containment and preference overriding. The query containment
is defined between two queries qa and qb if qb is subsumed by qa, qb \subseteq qa.
This means every answer to the query condition qb is an answer of qa. This
definition, leads for preference overriding between two preferences pa and pb.
The tightly overriding relationship between two preferences pb and pa defines
there is no other preference that lies between pb and pa. Such overriding
preferences are constructed as preference network for personalized answer
generation in existing system. The preferences in leaf node are not overridden
by any other preferences. The root node does not override any other related
preferences..

The existing system follows five algorithms such as preference
construction (PFC), paths, preference network construction (PNET), find
relationship and match to construct preference network. The path algorithm
stores preference diagram of all preferences through an off line process. The
PFC algorithm finds selection preferences from the user profile by verifying
path algorithm and sends them to PNET algorithm. This algorithm utilizes
find relationship and match algorithm in order to check overriding
relationship among preferences. Such overriding preferences are constructed
as preference networks by PNET algorithm. The steps of each algorithm are
given below.

**PFC and paths algorithm**

The algorithm PFC constructs preference networks from the related
preferences. The join preferences from user profile are composed if necessary
and then inserted into a queue RQ for processing. The paths algorithm
constructs preference diagram for each preference p. During run time the
direction of the edges is decided based on breath first traversal and generates
the set P of the root to leaf paths. This preprocessing saves time. Each path is
represented as a string for easy identification of overriding relationship. Only
selection preferences from the queue are sent to PNET for preference network construction.

Preference network construction (PNET) algorithm

The PNET algorithm constructs preference networks by selection
preferences with tightly overriding relationship. Finding of preferences with
tightly overriding relationship and organising them as a network are done by
PNET algorithm. Initial preference is considered as the root node Prt. At each
round, the preferences from RQ are checked for the overriding relationship
with the preferences in this network. The find relationship and match
algorithms checks for overriding relationship among preferences and sends
these preferences to PNET. The PNET examines the relationship between p
with the root Prt. If p is overridden by Prt, p becomes parent for Prt. Root
property ia transferred to p. Else Prt is overridden by p, p comes as a child of
Prt. If Prt has no edges then p becomes the child. Suppose Prt has child, then
child preferences are added in RQ. This algorithm moves down the network
until it reaches the node (ps,pi). If p is independent with the child of ps, then p
becomes the child of ps and flag becomes true. If p is the child of ps, two case
arises.

- if (ps,pi) is an actual edge, p checked with pi. Two options are there.
  - If p is overridden by pi, the algorithm breaks the edge between pi and its predecessor ps. It creates two edges
one connecting with ps and no outgoing edges from pi are added in RQ. Inserted becomes true.

- Else, pi is overridden by p, the child nodes of pi added to RQ. If there are no outgoing edges then p becomes pi’s child and inserted becomes true.

If p does not override any other preferences, it becomes a separate root. The following steps clearly identify the structure of PNET.

Flow of PNET

1. If $p \subset Prt$ p becomes root

   Else p comes under Prt

   Prt has no child p can come under Prt

   Otherwise childs of Prt are inserted into queue RQ.

2. This procedure continues until it reaches the edge $(ps, pi)$

   If p is independent with the child of ps then p becomes child of ps.

   Otherwise compare p with pi.

3. If $p \subset pi$ then break $(ps, pi)$ edge

   Else pi $\subset p$ add child of pi in RQ queue.

   Repeat step 2.

**Find relationship and match algorithm**

The find relationship algorithm checks for the overriding relationship among two preferences $p$ and $p'$ defined over the relation $R$ and match algorithm finds the relationship between the nodes of preference diagram with the string specified. The sizes of two preferences $p$ and $p'$ are $|P|$ and $|P'|$ respectively. The two paths $s_i$ belongs to $pi$ and $s'_i$ belongs to $p'$
match iff the string representations are the same. The number of pairs of matching paths \((si, si')\) is \(M\). If \(M=|P|\), then \(P\) is overridden by \(p'\). If \(M=|p'|\) then \(p'\) is overridden by \(p\). Otherwise \(p\) and \(p'\) are independent.

5.3.2 Exclude_Combine Algorithm

This algorithm executes number of simple queries \(Q\) with the preferences \(pi\) from preference networks. The Results \((pi)\) are the results of the query \(Q\) and \(pi\). \(Q1=Q\) and \(p1\), \(Q2=Q\) and \(p2\), \ldots , \(Qi=Q\) and \(pi\). Result\((p1)\), Result\((p2)\), \ldots , Result\((pi)\) are their results. In this way, this algorithm moves from preference networks to virtual networks of partial results. Now each node represents Result \(p(i)\). The edges in the opposite direction represent the difference operation between between \(pi\) and \(pj\). After this step, tuples that satisfy only \(pi\) and not \(pj\) present in Results \((pi)\). Finally, all the tuples that occur in at least \(I\) sets are ranked using special function and given as personalized results.

5.3.3 Replicate_Diffuse Algorithm

This is an another algorithm for queue personalization based on preference networks. The notion of REPLICATE_DIFFUSE algorithm is tuples at any level of the network satisfies its root. So it is enough to process only the roots then look up to find the particular preferences that give personalized results. REPLICATE_DIFFUSE algorithm has three steps.

- Create
  - In this step, For each root \(Prt\), the algorithm executes the respective query \((Q\) and \(Prt)\) and creates the set of results.
• Replicate
  • For each tuple in this set, the algorithm finds the exact node from its root in order of selectivity.

• Diffuse
  • In this step, at last all the tuples are found only in the corresponding, most specific independent preferences they satisfy.

For each root in the network, the algorithm executes Q and Prt. The resultant tuples are identified with tid. In order to find, which other preferences are satisfied by tid are found by executing Qrt(tid) in the other roots. This step returns 0 or more occurrences of tid depending upon the other root preferences. Then, this algorithm sends the tuples to their respective nodes generates personalized results.

5.4 PROPOSED QUERY PERSONALIZATION ALGORITHMS

Query personalization process enhances user preferences into user query that allows to explore alternatives and to prioritize and filter available choices. Query personalization conceptually involves query enhancement, materialization of personalized queries and aggregation of the result set into a single result set. The proposed system is conceptually designed with preference finder for related preference identification, and the proposed WETPER algorithm for query enhancement process and materialization of personalized queries and aggregation of personalized result. The WETPER algorithm works by finding most weighted composite preference set and process them as a preference network. The preferences that satisfy tuple generating dependency are organised as preference network. The preferences with maximum length in the network is enhanced with user query, then executed in the movie database and the relational schema is as follows
Movie (mid, mname, year, duration) Category (mid, cat)
Acted_by (mid, aid) Actor (aid, name)
Directed_by (mid, did) Director(did, dname)

. The size of the personalized result is set as 5 in default, and the enhancement takes place until the size of the result becomes 5.

5.4.1 Preference Finder

The main purpose of preference finder is to identify related preferences for the user query Q. The preferences for personalization process can be of the following types.

Atomic preferences

The simple preference conditions of a single selection or join condition are known as atomic condition. In this work, the liking of actor “Tom Cruise” is an example single atomic selection preference and liking of the attribute actor with genre is an example of atomic join preference.

Composite preferences

The combinations of atomic preferences are called composite preferences. Hence the composite preference is the combination of atomic selections, or atomic joins, or atomic selection and join. This research work defines this composite preference in a much higher way as defined below.

Multi granular preferences

Multi granular preferences are the combination of composite preferences with at least one selection condition in the sense, these preferences should have at least one selection condition compulsorily. It
means, these preferences are the combinations of join or selection condition with at least one selection condition.

Table 5.1 shows sample of these preferences with their doi for the user whose profile is presented in Table 4.8 in Chapter 4.

### Table 5.1 Preferences and their types

<table>
<thead>
<tr>
<th>S.No</th>
<th>Preference</th>
<th>Preference Type</th>
<th>doi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Movie.mid=Cast.mid and Cast.aid=Actor.aid</td>
<td>Atomic</td>
<td>0.074</td>
</tr>
<tr>
<td>2</td>
<td>Movie.mid=Cast.mid and Cast.aid=Actor.aid and Actor.aname=“Hendry jones”</td>
<td>Composite</td>
<td>0.452</td>
</tr>
<tr>
<td>3</td>
<td>Movie.mid=Cast.mid and Cast.aid=Actor.aid and Actor.aname=“Hendry jones” and Movie.mid=Genre.mid and Genre.genre=”action”</td>
<td>Multi granular</td>
<td>0.204</td>
</tr>
</tbody>
</table>

The preference finder defines the condition for composing two preference pa and pb such as

**Two preferences pa and pb are composable**

Iff:

*pa is a join preference that refers relation Ra with Rb and pa=(qa, da)*

*where Ra and Rb are relations in database*

*pa is a preference, qa is the query condition and da is the doi of the preference pa.*

*pb is a join or selection preference on relation Rb where pb=(qb,db)*

*where pb is the preference, qb is the query condition db is the doi of preference pb.*
The composite preference of \( p_a \) and \( p_b \), represents the conjuction of \( p_a \) and \( p_b \) which actually denotes \((q_a \text{ and } q_b, F(da,db))\) where \( F(da,db) \) is derived from Equation (4.1) in Chapter 4.

The preferences in user profile consist of simple and composite preferences. The preference finder finds top \( k \) related preferences from the user profile. The preference finder considers the preference \( p^* \) as a possible conjunctive query \( \phi(p^*) \) that gives tuples \((RS^*)\) from the database satisfying \( p^* \). The two preferences \( p^*, p^{**} \) with the corresponding query conditions \( q^*, q^{**} \) are said to be related if they satisfy any one of the conditions as follows.

\[
\forall \; q^* \; \text{of} \; p^* \rightarrow \exists \; t^* \; \text{which satisfies} \; q^{**} \; \text{of} \; p^{**} \; \text{refers tuple generating dependency}
\]

\[
\forall \; q^* \; \text{of} \; p^* \rightarrow \exists \; t^* \; \text{which equals to} \; t^{**} \; \text{of} \; q^{**} \; \text{of} \; p^{**} \; \text{refers equality generating dependency}
\]

This research work considers tuple generating dependency alone for its process. The preference finder defines the preference overriding relationship between preferences \((p^*, p^{**})\) if \( q^* \) is subsumed by \( q^{**}, q^* \subseteq q^{**} \).

\[
q^* \; \text{for each} \; p^* \; \text{in a set of user preferences} \; P \rightarrow \exists \; p^{**} \; \text{with} \; q^{**}.
\]

Hence, preference finder defines the overriding relationship among two preferences \( p^*, p^{**} \) if there exists a tuple generating dependency among two preferences. The algorithm for finding top \( k \) related preferences is as follows.
**Algorithm for top k preferences**

*Input*: User profile U, User query Q  
*Output*: Set of top k related preferences

*Begin*

*Step 1*: RP={}, S={}, P={}

*Step 2*: For each p ∈ U

\[ S = \text{atomic condition of } p \text{ related } Q \]

End For

*Step 3*: While \( S \neq \emptyset \) {

For each \( p_i \in S \{ \)

RP=Execute \((p_i \text{ and } Q)\) // for tuple generating dependency

If RP not null

add p to P

\}  \}  

End

After, this step the related top k preferences for the query are given to WETPER algorithm to find personalized results.

**5.4.2 WETPER Algorithm**

The proposed WETPER denotes weighted personalization algorithm which gets input from preference finder. The related preferences from preference finder have overriding relationship. The preference with maximum weight is treated as root node. The following nodes are organized as a preference network with the overriding relationship as a graph. If the node does not override any other node, then it is constructed as another root node. Root node does not overridden by any other node. The set of nodes with the maximum length is the most favourite preference since it denotes the maximum weighted preferences in the user profile. This composite preference is enhanced with user query, and its results are given as personalized results. If the result size is less than personalized result size, the next lengthiest composite preference is enhanced with query making it as personalized query.
In this way, WETPER algorithm presents more personalized results to the user by calculating doi of preferences quantitatively. The number of personalized queries constructed in WETPER algorithm is very less compared to other two existing algorithms REPLICATE_DIFFUSE and EXCLUDE_COMBINE. The proposed WETPER algorithm is as follows.

**Input: User profile U, Query Q, related preferences P**

**Output: Personalized Results PRS**

**Begin**

\[ VH=\{\}, \; EH=\{\}, \; Preference\_Network \; G=\{\} \]

For each \( p \) from \( P \)

\[
Q_p=\{p, \; P\} \text{ where } P \text{ has set of related preferences for } Q
\]

While \( Q_p \neq \{\} \)

\[
\{ \text{ Remove } (p, \; P) \text{ add to } G
\]

If (root = \{\}) {make pi as root}

else

Find dependency between nodes of \( G \) and \( p \)

\[
\{ \text{ If } (p \text{ not related to nodes in } G) \{\text{ set as individual node}\}
\]

Else {add \( p \) in respective place with overriding property}

\[
\}
\]

Find maximum connected node in \( G \)

\[ RS=\text{ Execute( query with maximum weighted preference) } \]

For each \( ti \) in \( RS \)

\[
\{ \text{ PRS= tuples satisfying maximum node in } G
\]

If \( PRS < 5 \) {move to next maximum connected node in \( G \)}

Else exit

\[
\}
\]

**End**

The steps of preference network construction and it’s breaking up steps are given in Figure 5.4. In step 1, \( pa \) is the root node and \( pb \) is overridden by \( pa \). In step 2, \( pc \) is overridden by \( pb \) and \( pa \). In step 3, \( pd \) is also overridden by \( pb \) and \( pa \). Step 4, connects \( pe \) with \( pd \) with overriding...
principle. Hence $pa$ and $pc$ are detached from $pb$ and it is connected with $pd$ and $pe$ with maximum length. The step 4 and 5 are proceed further with top k related preferences. Finally, the multi granular preference with maximum length is enhanced with user query $Q$ to find personalized results and this process continues until the size becomes 5.

Figure 5.4 Steps of WETPER
Table 5.2 shows the personalized results of the query select movies of year>2000. The constructed sample personalized query is given in Figure. 5.5 According to user profile, the users favourite actor is Johnny Depp and based on doi genre thriller and adventure comes in order. From this personalized query itself, 5 personalized answers have been generated.

<table>
<thead>
<tr>
<th>Year</th>
<th>Movie Title</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>the tourist</td>
<td>Johnny Depp</td>
</tr>
<tr>
<td>2012</td>
<td>dark shadows</td>
<td>Johnny Depp</td>
</tr>
<tr>
<td>2004</td>
<td>secret window</td>
<td>Johnny Depp</td>
</tr>
<tr>
<td>2004</td>
<td>finding neverland</td>
<td>Johnny Depp</td>
</tr>
<tr>
<td>2000</td>
<td>the man who cried</td>
<td>Johnny Depp</td>
</tr>
</tbody>
</table>

Select distinct Movie.name from Movie, Cast, Actor, Directedby, Director where Movie.year > 2000 and ((Movie.mid=Cast.mid and Cast.aid=Actor.aid and Actor.aname='Johnny Depp') and (Movie.mid=Genre.mid and Genre.genre='thriller') or (Movie.mid=Genre.mid and Genre.genre='adventure'))

Figure 5.5 Sample personalized query

5.5 RESULTS AND DISCUSSION

The novelty of the proposed system stems from allowing user to enter their favourite actor, director and genre and the system finds doi of them. Earlier approaches allow user to enter preferences with doi. Practically it is not possible to assign exact doi freely. Inconsistent input decreases the quality of personalized result. The proposed personalization algorithm shows better result since it works on valid preferences dynamically and it is proved in experiments also. Incorporation of multiple criteria that affects user interest leads for accurate result.
The empirical evaluation of the system is done with 400 users. The users are allowed to enter single or conjunction of preferences, initially 87 users entered single preferences, the remaining 313 users entered conjunction of preferences. This observation indicates users who gave fine grained choice have more knowledge about the domain. But the proposed system restricts them to give single conditions because, from analysis it is proved that the system can make more number of composite preferences as shown in Equation (5.1) that the people can do.

\[
\text{Number of combinations for n preference} = \sum_{r=1}^{n} C_r^n
\]

(5.1)

The user interface is designed to search movies based on actor, director, genre and year of the movie. The system allows user to make a pool of possible searches. All the users are asked to give the five compulsory queries with their own search queries. The compulsory queries are

1. Movies based on actor with year.
2. Movies with actor, actress combinations
3. actor with director combinations
4. actor with category combinations
5. category with year

User can set personalized result set size. The personalized result set size ranges from 5 to 20 in the proposed system, it can be further extended or restricted. The experimental result demonstrates the efficiency and effectiveness of the proposed system.
Evaluating PS is a trivial task because different algorithms show good and bad performance depending on the application domain. In the early days, performance accuracy is considered as the basic factor for personalization. Nowadays, the decision support system measures quality of personalization with correct conclusion and the performance accuracy increases if it does not lead of wrong conclusion. Yang et al. (2005) presents different methods for evaluating PS by the following two methods.

1. The controlled experiments
2. knowledge driven based experiments

The first method is the classical evaluation scheme of personalization where the standard measures such as customer satisfactory, number of movies clicked are used to decide the goodness of the system. In the knowledge driven based experiments, data mining techniques are used to evaluate the proposed system.

5.5.1 Controlled Experiment Method

This method defines answer score as a user satisfaction score. Answer score is a numeric score that specifies how satisfactory the answer is to the user. Degree of difficulty is also a numeric score that defines the difficulty associated with selecting their favourite result from personalized result. The user rates the personalized result ranging from 0 to 10 as answer score. Degree of difficulty stands for the difficulty to find something interesting. Both scores ranges from 0 to 10. It is found that the use of composite preferences substantially reduces the difficulty to find interesting tuples within answer and attracts distinctively higher answer scores. The result of answer score and degree of difficulty with number of users for WETPER algorithm is given in Figure 5.6.
Figure 5.6 Answer score and degree of difficulty of WETPER

The proposed system follows answer score rating scheme as the existing movie PS and allows user to rate personalize movies. The answer score of personalized results is compared with unpersonalized results. In addition, the proposed WETPER algorithm results are compared with existing REPLICATE_DIFFUSE and EXCLUDE_COMBINE algorithm. The three algorithms are randomly chosen during the execution time. The users are unaware about which algorithm is in process and they provide answer score for the personalized answers. Answer score comparison of personalized and non personalized results are shown in Figure 5.7.
The answer score of WETPER algorithm is improved 2.1 times better than without personalization method. The answer score of WETPER algorithm is improved 1.4 times better than EXCLUDE_COMBINE and 1.2 times better than REPLICATE_DIFFUSE. The answer score of WETPER algorithm shows 47% improvement than without personalization method, 26% improvement than EXCLUDE_COMBINE and 12% than REPLICATE_DIFFUSE.

The comparison of without personaliation, EXCLUDE_COMBINE, REPLICATE_DIFFUSE and WETPER algorithm for degree of difficulty is presented in Figure. 5.8. The answer score of WETPER algorithm is higher than REPLICATE_DIFFUSE algorithm and the degree of difficulty is sometimes lesser than WETPER algorithm. This happens because, the personalized result satisfies many of the preferences in preferences network. But WETPER forms personalized queries one by one. So, the difficulty to find interesting tuples from personalized results is slightly higher in WETPER algorithm.

Figure 5.7 Answer score of personalization algorithms
Figure 5.8 Degree of difficulty of personalization algorithms

The personalization time of the proposed approach and existing approaches is shown in Figure 5.9. The time of user preference gathering is not considered in personalization time. The processing of user preferences for personalized query construction and their execution and result ranking are measured as personalization time. The approach, without personalization is processed by executing user query directly and personalized results are randomly chosen from the result.

Figure 5.9 Personalization time analysis
In the first iteration the profile size is 18. When the iteration goes on, the learned preferences are updated in the profile. Hence the size of the profile increases in case of EXCLUDE_COMBINE and REPLICATE_DIFFUSE. In case of WETPER, recent preferences are replaced with least used preferences.

The process time of without personalization technique does not show any improvement when the number of preferences increased. In case of EXCLUDE_COMBINE algorithm, the process time is reduced to 17% at 6th iteration. In REPLICATE_DIFFUSE algorithm, the process time is reduced to 41% at the 6th iteration. The proposed WETPER algorithm shows 71% improvement in process time at 6th iteration. The personalization time analysis of a single user is given in Figure 5.9. The user with maximum time difference for proposed and existing algorithms is shown graphically to prove the efficiency of the system in terms of personalization time.

5.5.2 Knowledge Driven Evaluation Method

This method evaluates the system from knowledge about expected outcomes under various circumstances. This method follows three different steps such as gathering knowledge about various situations, determining the measures to prove the efficiency of the system and decides the best situations from the measurement. In this aspect, the proposed system is evaluated by precision and recall which are the basic measures in search evaluation. Precision is the ratio of the number of personalized results retrieved to the total number of personalized and non personalized records retrieved. Recall is the ratio of the number of personalized results retrieved to the total number of personalized results in the database. Both precision and recall are expressed as percentage. The proposed system finds precision by Equation (5.2), and recall by Equation (5.3).
\[
\text{precision} = \frac{\text{Number of correctly detected personalized results of the user}}{\text{Number of all detected personalized results of the user}}
\]  

(5.2)

\[
\text{recall} = \frac{\text{Number of correctly detected personalized results of the user}}{\text{Number of all personalized results of the user}}
\]  

(5.3)

Personalized results may be relevant or irrelevant and relevancy of personalized result is a perception of individual. Measuring recall is difficult since it is hard to know how many relevant records exist in the database satisfying user preferences. The precision and recall values of the WETPER and REPLICATE_DIFFUSE algorithm for 5 users are given in Table 5.3.

**Table 5.3  Precision and Recall of WETPER and REPLICATE_DIFFUSE algorithms**

<table>
<thead>
<tr>
<th>User</th>
<th>Measure</th>
<th>WETPER</th>
<th>REPLICATE_DIFFUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>I</td>
<td>Precision</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>19</td>
<td>18.7</td>
</tr>
<tr>
<td>II</td>
<td>Precision</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>22.2</td>
<td>12.9</td>
</tr>
<tr>
<td>III</td>
<td>Precision</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>16.6</td>
<td>21.4</td>
</tr>
<tr>
<td>IV</td>
<td>Precision</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>25</td>
<td>23.8</td>
</tr>
<tr>
<td>V</td>
<td>Precision</td>
<td>60</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>20</td>
<td>28.5</td>
</tr>
</tbody>
</table>

The precision of WETPER algorithm is comparatively better for almost all users. The size of personalized result is fixed during the experimentation of DQPS and further this size can be extended in future. Hence, the denominator of precision formula is 5 for all the users. The numerator specifies how many personalized results are accepted by the users as their more personalized results. The numerator of precision and recall are same. The denominator of recall specifies the count of personalized movies satisfying user preferences in movie database. It is observed that, this
The denominator value ranges from 8 to 32 in this experiment. The recall of Q3 is high for user III and V because the number of personalized results of the user in database is low in this case. Experimental evaluation indicates WETPER algorithm shows 35% improvement than EXCLUDE_COMBINE algorithm and 19% improvement than REPLICATE_DIFFUSE algorithm in terms of personalization time. The precision and recall measures of the personalized results exhibits the significance of WETPER algorithm.

The controlled and knowledge driven based experiments and their significant results shows the effectiveness and efficiency of the proposed system. The overall personalization time of WETPER algorithm shows 35% improvement than EXCLUDE_COMBINE algorithm and 19% improvement than REPLICATE_DIFFUSE algorithm.

5.6 SUMMARY

The presence of personalization technique in E-commerce is common today. This chapter presents the functionality of existing algorithms and WETPER with personalized answer generation. The comparison of graph computation problem and weight computation problem is detailed. The time for constructing preference network and personalized queries are comparatively high in REPLICATE_DIFFUSE and EXCLUDE_COMBINE where as in WETPER the number of personalized queries constructed is very less due to quantitative maximum weighted personalization logic. The time for executing WETPER algorithm also reduces when tuple generating dependency increases among preferences. Personalization time is considered as a performance factor and its results are presented. The proposed personalization logic with maximum scored personalized query is completely different from 1 out of top k logic. Finally, this chapter presents the experimental results WETPER providing an insight for the appropriateness and effectiveness of DQPS highlighting reduced personalization time.