Research Methodology

CHAPTER 3

RESEARCH METHODOLOGY

Navigation pattern is an important web mining problem that has been studied over the years. Applying and analyzing three different algorithms to web documents is the main focus of this research. The working of the three selected models is presented in this Chapter. Initially, a general procedure for web navigation is presented followed by a description of the DBscan algorithm. The rest of the chapter is dedicated to the explanation of FP-growth and Flame algorithm.

3.1 OVERVIEW OF THE METHODOLOGY

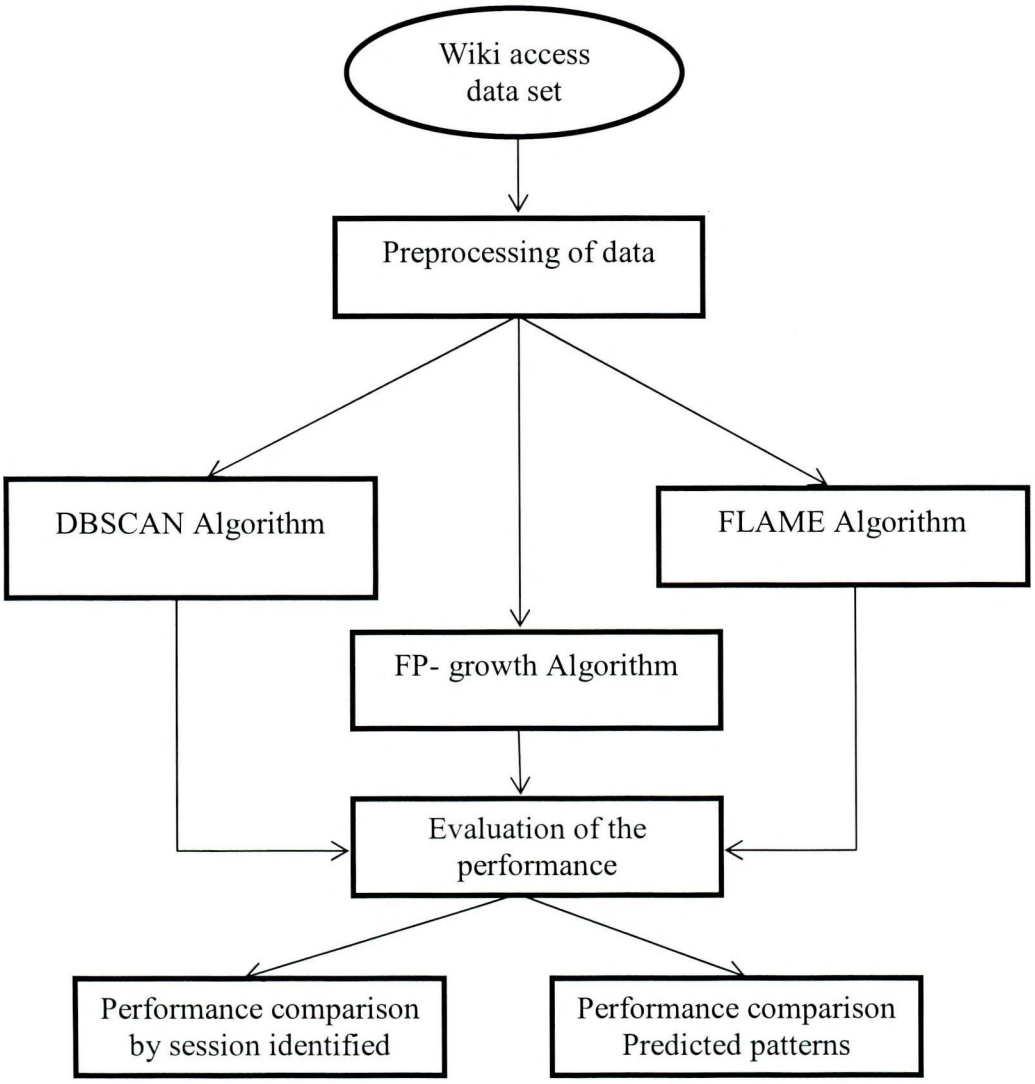

Figure 3.1 Flowchart showing overall research plan

3.2 PREPROCESSING

The wiki access dataset, used in the process is subject to the preprocessing stage. An effective preprocessor represents the web data effectively in terms of both session identification and navigation pattern requirements and maintains good retrieval performance (precision and recall). This phase is the most critical and complex process that leads to the representation of each weblog by selecting user id and time stamp.

The main objective of preprocessing is to obtain the session identified as monolithic or shared patterns. The preprocessing procedure used in the proposed system is given in Figure 3.2. In general, the preprocessing in this process includes 4 stages as data cleaning, user identification, session identification and the navigation session identification.

- In data cleaning process the raw data taken for the process get analyzed and the request which deviates the user from their navigation pattern is removed. Some logs present in the data are implicit in nature which automatically predict the user navigation page and directs them towards that page. This will distract the user and affect the analysis of the navigation. These data is removed from the log file and helps in accurate prediction of the session interested by the user.
- The logs which are rarely viewed by the user don't help in any prediction to obtain better result they are all removed in preprocessing stage. The log files used to predict the users referred in each page. The address of each user who viewed the page separates each user in individual. The log file which is received after the process contains the log data which is useful for better prediction of the session in accurate manner without any deviation.
- In this process the session used by each user is identified as monolithic session or it's a shared pattern. When the pattern is not having much information they will get deleted. These patterns help in obtaining the navigation process of each user.

```
Input: log file with all sessions
Output: separation of useful session
Begin
{
The session identified depend on user id and time stamp
For
    Divide the sequence depend on the time spend in each page by the user
        If (monolithic sequence)
            S is single session
        Else
            S is shared by various users. Each user separated by column.
        Else
            Skip sequence
End For
}
End
```

Figure 3.2: Preprocessing Procedure

3.3 DBSCAN ALGORITHM

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is one of the most common clustering algorithm proposed by Ester et al, (1996).The preprocessed data set is subjected to the clustering process and this process helps in grouping of the sessions based on the similarity. The DBSCAN algorithm can identify clusters in large spatial data sets by looking at the local density of database elements, using only one input parameter. It is density-based clustering algorithm because it finds a number of clusters starting from the estimated density distribution of corresponding nodes. DBSCAN requires two parameters: (eps) and Minimum number of points required to form a cluster (minPts). This point's neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. If a point is found to be a dense part of a cluster, its - neighborhood is also part of that cluster. Hence, all points that are found within the -neighborhood are added, as is their own-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found.

The preprocessed data which includes the user id, session used, timestamp is processed using the DBscan algorithm. The process is shown in Figure 3.3.

3.3.1 Flowchart

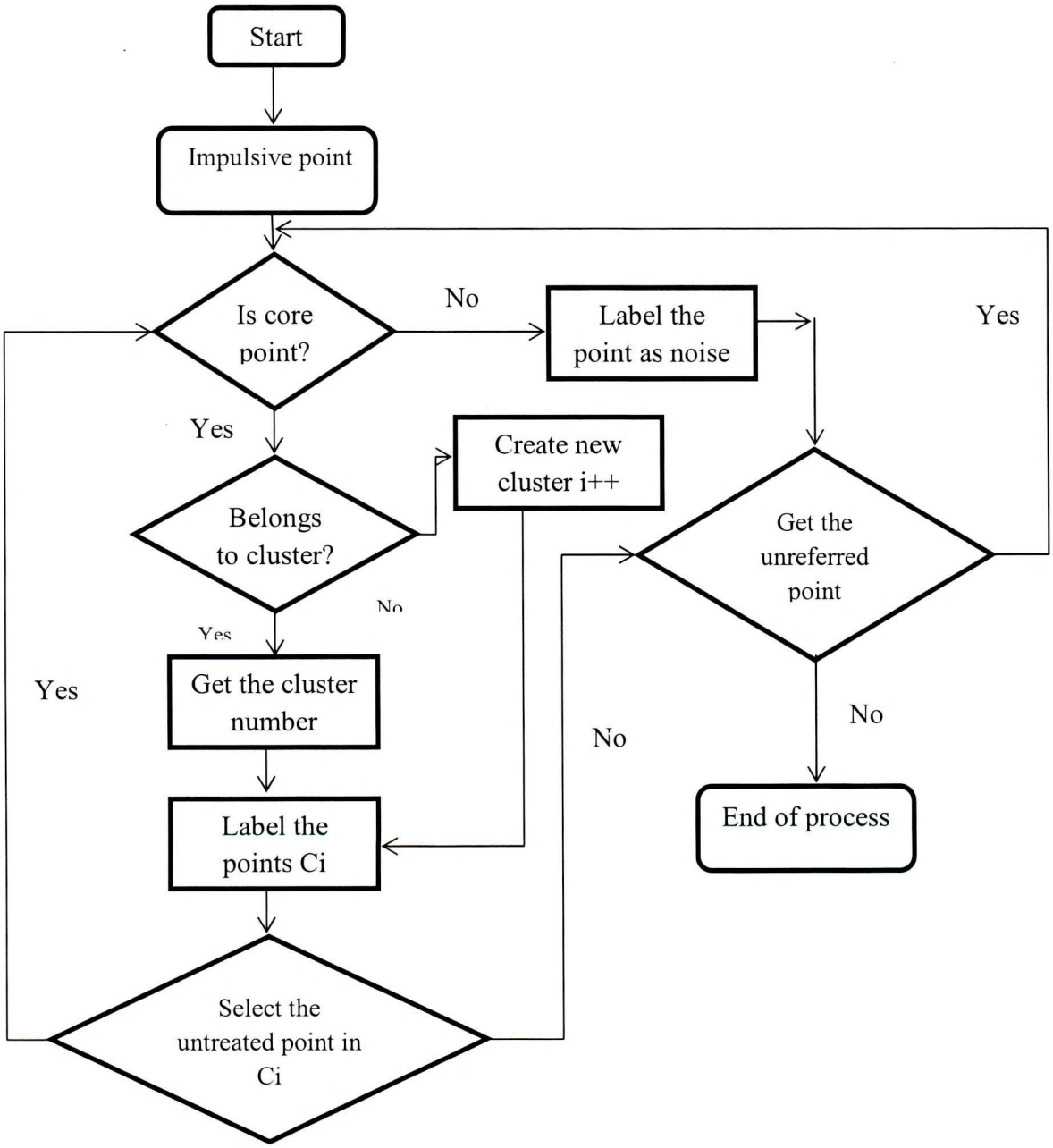

Figure 3.3 Flowchart for DBscan algorithm

3.3.2 Steps

m and n are data points present in a data set. The cluster is C with specified value of epsilon and minpoints so that it satisfies the condition specified in the data set. The session are founded by the iterative process where the epsilon value of the each session get varied depend on the variation in the number if user refers each page.

- If m is the session present in the cluster and n is in the density specified region of m. Then n also belongs to the cluster C.
- The iterative process starts from the core points and covers all the data points.

```
Begin

For each d belongs to the data set do
{
   If d is not assigned to any cluster then
      If d is a core object then
        The d collects the entire object present in the object reachable
depend on epsilon and
        Minpoints
Form a new cluster
}
End for
End
```

Figure 3.4: Pseudocode for DBScan Algorithm

3.4 FP-GROWTH ALGORITHM

Frequent pattern is one of the association rule mining algorithm. FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques: (1) a large database is compressed into a condensed, smaller data structure, FP-tree which avoids costly, repeated database scans, (2) The FP-tree-based mining adopts a pattern-fragment growth method to avoid the costly generation of a large number of candidate sets, and (3) a partitioning-based, divide-and-conquer method is used to decompose the mining task into a set of smaller tasks for mining confined patterns in

conditional databases, which dramatically reduces the search space. The FP-growth method is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent-pattern mining methods. (Jiawei Han et al,2004)

3.4.1 Steps

- Frequent pattern tree is constructed from the large data set helps in relating the sessions which are referred by the user.
- The sessions which are related get in to the tree structure helps in analyzing the link between each page. This also helps in finding the user navigation pattern.
- The complete data set is connected to tree structure on a single scan.
- The patterns which are more commonly used to get high count value.
- Divide and conquer method is used, where the subsets are avoided. The high weightage in the count includes that the most used link by the user.
- The header table is constructed using the conditional pattern.

```
Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null).
Procedure FP-growth (Tree, α)
{
1) if Tree contains a single path P then
2) for each combination do generate pattern β
  α with support = minimum support
of nodes in β.
3) Else For each header ai in the header of Tree
do {
4) Generate pattern β = ai α with support = ai.support;
5) Construct β.s conditional pattern base and
then β.s conditional FP-tree Tree β
6) If Tree β= null
7) Then call FP-growth (Tree β, β)}
}
```

Figure: 3.5 Pseudocode for FP-Growth Algorithm

3.4.2 Process

FP-tree is a frequent pattern tree consists of one root labeled as "null". It has a set of item prefix sub trees as the children of the root, and a frequent-item header table. Each node in the item prefix sub trees has three fields:

- item-name to register which item this node represents, the page which is seen by the user
- Count, the number of transactions represented by the portion of the path reaching this node, and
- Node-link that links to the next node in the FP-tree carrying the same page, or null if there is none.

Each entry in the frequent-item header table has two fields

- Page-name and head of node-link that point to the first node in the FP-tree carrying the item-name that is the page name.

The session of the each user gets identified from the mentioned time criteria and each session is made with a link of related session. This process helps in predicting the navigation pattern of the user. The transaction between the each session for the user gets identified by the tree structure. The links between the each transaction of the user get identified by the tree structure. The number of times the patterns get moved is identified by the frequency of the page analyzed. The sessions prediction by the frequency of the user analyzed, the accuracy of the prediction is improved from the pattern frequency.

The weblogs present for the analysis having the record of each user pattern of pages visited. The frequent pattern seen by each user helps in better prediction of the page request provided by the user. The minimum support threshold and minimum confidence threshold helps to predict whether the pattern is in frequent process or not. The link between the pages provides the transaction details which is mined on basis of frequent usage helps in constructing the tree structure. The noisy data are removed in the removed in the tree construction stage helps in quicker prediction of the process. The pages present as p1, p2,...pn and the user represented as u1, u2, ... , un. The transaction between the pages is analyzed and the frequency count helps to analyze the pattern occur in frequent manner and the prediction can be done.

The flowchart for the process is shown in the figure 3.6. The initial stage includes by setting the MST value which helps to eliminate the page at the initial stage and the prediction can be achieved in the better rate.

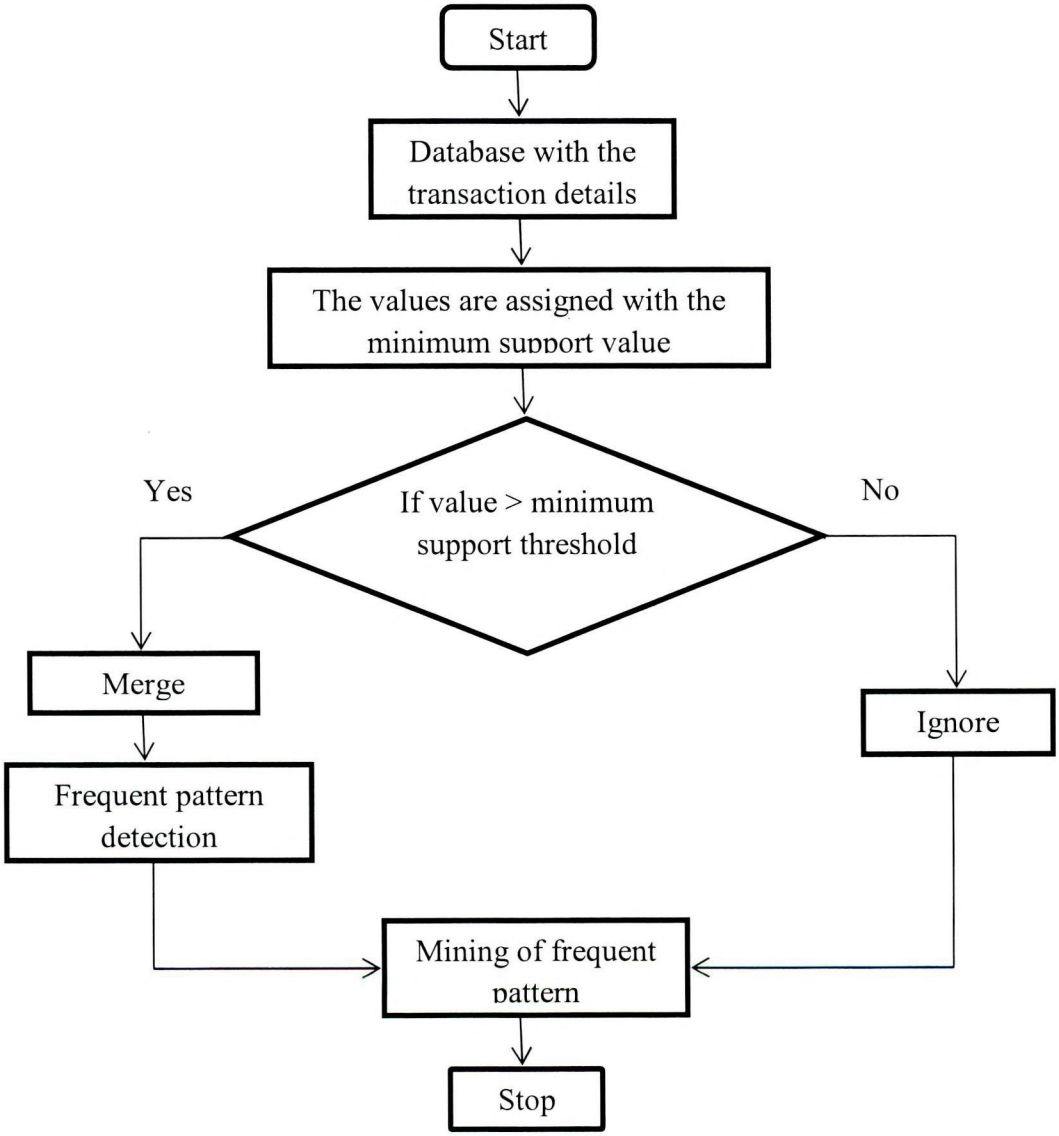

Figure 3.6 Flow chart for FP-growth algorithm

3.5 FLAME ALGORITHM

The flame algorithm used for sequential pattern mining has been applied to webpage navigation prediction. It provides better session prediction and suggestion by mining with the sub strings so more relevant data get mined for accurate results (AvriliaFloratou,et al, 2011). The flexible and accurate motif detector is suffix based tree structure where the data pattern

can be predicted in more flexible manner. This algorithm is capable to support the similarity pattern without the noise effect.

The matrix model is constructed depend on the pages visited by the user, the time spent on each page, the string used by the search and the minimum support for each page is taken in to account to support the prediction. These are taken in to account to build the matrix to support more number of suggestions. This process is powerful method in determining the better pages in suggestion and in a flexible manner.

When the user enter the web page to obtain some information, the string length get computed and the strings present in the database is compared. The distance between the strings gets compared using hamming distance. The instances conditioned as present in the database not the string is conditioned to present. The structure is constructed depend on the suffix structure. The search in the matrix on the string doesn't match only the perfect match but also allows some deviation accepted by the normalization value. They provide all the suggestions matched with the search process, the tree is constructed depend on the suffix value in the descending order. The string search help in constructing the tree in an effective manner and the patterns related can be easily extracted. The data tree is constructed with the strings present in the data set. The model tree is constructed depend on the pattern of searching with the related string match pattern. This model work helps to avoid the redundant of data search in string. The search of the string gets mismatched then they get pruned from further process. The process mainly includes matches of the string and the support provided by them (AvriliaFloratou,et al, 2011).

The Hamming distance (Hamming 1950) is a metric expressing the distance between two objects by the number of mismatches among their pairs of variables. It is mainly used for string and bitwise analyses, but can also be useful for numerical variables.

$$d(i,j) = \sum_{k=0}^{n-1} [y_{i,k} \neq y_{j,k}]$$

where,
d (i,j) is the Hamming distance between the objects *i* and *j*,

k is the index of the respective variable reading *y* out of the total number of variables *n*.

The Hamming distance itself gives the number of mismatches between the variables paired by *k*.

Begin

Input:

1. Input sequence is composed of symbols from a discrete alphabet sets.

Set of items/alphabets I= {a1, a2, ..., am}

Input Sequential database D = { x | ∀x ϵ I}

E.g. I = {A, B, C, D}

D = ACABBDACCAB

2. Values of L, d and k.

Where

L: Length of substring

d: Distance/number of mismatches allowed

k: Minimum Support

Output:

Set of (L, d, k) model Strings.

Op = {s1, s2, ..., sn}

Where

Si = {si1, si2, . . .,sim} is set of instances for si.

Op = {si| d(si, sij) ≤d, |sij|=L and m ≥k}

Procedure:

FreqPattern(mTree, root, L, d, k)

1. process();
2. getmodelTree();
3. For i=0 to i<mTree.size()
4. computeSupport();
5. End For

process()

Construct count suffix tree on input dataset.

getmodelTree()

Construct suffix tree of depth L on strings of length L that occurs in the count suffix tree.

computeSupport()

1. oldMatches=model.parent.matches;
2. if(model.parent.id=1)
3. newmatches=expandMatches()
4. else
5. for k=0 to k<oldMatches.size()
6. newMatches.addAll(expandMatches())
7. End for
8. setMatches(newMatches)
9. for i=0 to modelTree.size()
10. x=modelTree.get(i)
11. if(distance()<=d)
12. model.matches.add(x)
13. model.modelSupport +=x.support;
14. End for
15. End

3.7 Pseudocode for Flame Algorithm

3.6 PERFORMANCE EVALUATION METRICS

The overall performance of the DBscan, FP-growth and Flame algorithm is analyzed in terms of the Average Patterns, Session Identification, Precision, Recall, Accuracy and F-measure.

3.7 CHAPTER SUMMARY

This chapter presents the methodology used to obtain patterns and identifying sessions. The process of session identification, prediction of pattern and the steps involved in different strategies is discussed.

The experimental set up and the environment to support the work along with results is discussed in the next chapter.