International Journal of Advanced Computer Technology (IJACT)
ISSN:2319-7900

Web Page Classification using FP Growth Algorithm

Akansha Garg,*Computer Science Department*
Swami Vivekanad Subharti University,Meerut, India

Abstract **-** The primary goal of the web site is to provide the relevant information to the users. Web mining technique is used to categorize users and pages by analyzing user's behavior, the content of pages and order of URLs accessed. In this paper, proposes an auto-classification algorithm of web pages using data mining techniques. The problem of discovering association rules between terms in a set of web pages belonging to a category in a search engine database, and present an auto – classification algorithm for solving this problem that are fundamentally based on a FP - growth algorithm.

Keywords- *Association Rules, Auto-Classification, FP – growth, Web Mining*

1.1Introduction

A web search engine is designed to search information on the World Wide Web. Web search engine also mines data available in databases. The existing search engines, Google, Bing, and Yahoo. Interacting with the web for the following purposes: Finding Relevant Information, Discovering New Knowledge, Personalized Web Page Synthesis, Learning about Individual Users. The Apriori algorithm has some drawbacks to classify the web pages. In this paper, FP-Growth algorithm is used for categories the different web page. The proposed technique has two phases. The first phase is a training phase where human experts determine the categories of different Web pages, and the supervised Data mining algorithm will combine these categories with appropriate weighted index terms according to the highest supported rules among the most frequent words. The second phase is the categorization phase where a web crawler will crawl through the World Wide Web to build a database categorized according to the result of the data mining approach. The popular FP-growth Association Rule is applied to a particular kind of set enumeration tree, the FP-tree, also developed by Han et al. Both the FP-tree and the FP-growth algorithm are described in the following two sections. A short technique is also provided in this paper. The essential difference between the original **FP-growth algorithm and the this algorithm (Java) implementation is that a second tree structure, the T-tree is used to store the discovered frequent item sets and subsequently generate the desired ARs.** A popular "preprocessing" tree structure is the FP-tree proposed by Han et al. (2000). The FP-tree stores a single item (attribute) at each node, and includes additional links to facilitate processing. These links start from a *header table* and link together all nodes in the FP-tree which store the same "label", i.e. item.

To illustrate the method, considering the simple data set:

{1, 3, 4}
{2, 4, 5}
{2, 4, 6}

The construction process begins with an initial pass to count support for the single items. Those that fail to meet the support threshold are eliminated, and the others ordered by decreasing frequency. For our illustration we will assume that all 1-item sets are adequately supported, so the ordering will be {4,2,1,3,5,6}. We then pass through the dataset a second time and produce an initial FP-tree. We commence by reading the first record in the dataset and place this in the FP-tree (Figure 1(a) --- note the links from the header table). We then add the second record; the first element of this is common with an existing node and so we add the new node to the FP-tree structure as shown in Figure 1(b). We then add the last record (Figure 1(c)) to complete the initial FP-tree.

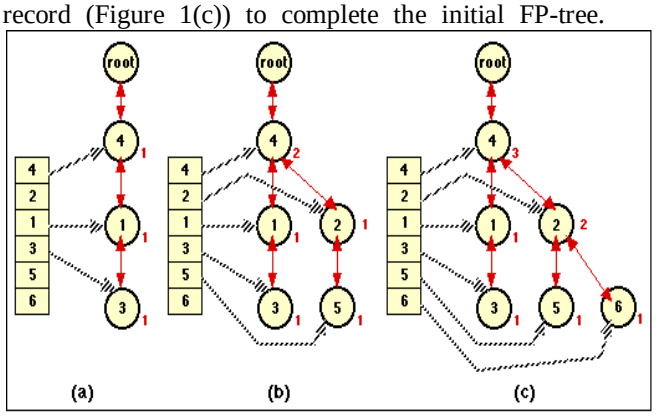

Figure 1:FP growth algo

ISSN:2319-7900

The algorithm, FP-growth, for mining the FP-tree structure is a recursive procedure during which many sub FP-trees and header tables are created. The process commences by examining each item in the header table, starting with the least frequent. For each entry the support value for the item is produced by following the links connecting all occurrences of the current item in the FP-tree. If the item is adequately supported, then for each leaf node a set of *ancestor labels* is produced (stored in a *prefix tree*), each of which has a support equivalent to the sum of the leaf node items from which it is generated. If the set of ancestor labels is not null, a new tree is generated by the set of ancestor labels as the dataset, and the process repeated. In our implementation all frequent itemsets thus discovered were placed in a T-tree, thus providing fast access during the final stage of the ARM process, while at the same time providing for the deletion of FP-subtrees and tables created "on route" as the FP-growth algorithm progressed.

For example, in Figure 1 (c) we would start with attribute 6. This has a support of 1, and assuming this is above the required support threshold, this would be identified as a frequent set. There are two ancester labels so a new FP-tree is created (Figure 2(a)) using the ancestor labels as the input set /bin/sh: -c: line 1: syntax error near unexpected token `(' /bin/sh: -c: line 1: New FP-tree is created (Figure 2(a)) using the ancestor labels as the input set ' (note that the support values for the label are equivalent to the leaf node). FP growth is applied again and the frequent sets {2,6} discovered. There are still ancestor nodes so another tree is produced (Figure 2(b)) and the process is continued until there are no more ancestor nodes (Figure 2(c)).

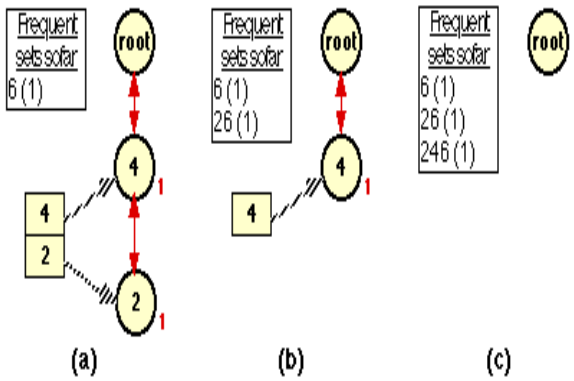

Figure 2: *FP growth algorithm*

1.2.Implemented Algorithm

The implementation of FP growth reorders and prunes the input data. Reorders so that the most common single attributes appear first, and pruned so that unsupported 1- item sets are not included. The effect of this reordering and pruning is to make the running of the code much more computationally efficient (an advantage appropriate to all tree based ARM algorithms). Thus, given an input set of the form:

{1 3 4}
{2 4 5}
{2 4 6}
This would be reordered to give:
{4 1 3}
{4 2 5}
{4 2 6}

However, for the code to operate correctly item set attributes must be numbered sequentially. Thus, some renumbering must take place:

4 becomes 1
2 becomes (stays on) 2
1 becomes 3
3 becomes 5
5 becomes (stays on) 5
6 becomes (stays on) 6
The input data now looks like this:
{1 3 4} {1 2 5} {1 2 6}
Running the code produces a tree of the form:
(1) 1:3 (ref to null)
(1.1) 2:2 (ref to null)
(1.1.1) 5:1 (ref to null)
(1.1.2) 6:1 (ref to null)
(1.2) 3:1 (ref to null)
(1.2.1) 4:1 (ref to null)

Where given X:Y, X is the attribute number and Y is the associated support. Reordering and pruning can be switched off by removing (commenting out) the lines:

newFPtree.idInputDataOrdering();
newFPtree.recastInputDataAndPruneUnsupportedAtts()
;
newFPtree.setNumOneItemSets();
From the FPgrowthApp.java class in which case this would give an FP tree of the form:
(1) 1:1 (ref to null)
(1.1) 3:1 (ref to null)
(1.1.1) 4:1 (ref to null)
(2) 2:2 (ref to null)
(2.1) 4:2 (ref to 4:1)
(2.1.1) 5:1 (ref to null)

ISSN:2319-7900

(2.1.2) 6:1 (ref to null)
Note that this unordered tree contains more nodes than the ordered tree.

1.2.1 Advantage of Implemented Algorithm

The advantages offered by algorithms such as FP-Growth (Han et al. 2000) are partly gained from the ordering process, which reduces the overall size of the input dataset (because unsupported single items are eliminated), and reduces processing time by allowing the most common items to be processed more efficiently. The first advantage can also be realized by other ARM algorithms; anything that reduces the overall size of the input dataset has a positive effect on mining efficiency. However, the advantage will only be significant given sparse datasets and/or high support thresholds, where many 1-item sets are likely to be unsupported and therefore can be eliminated from the input dataset. It should also be noted here that reordering of attributes in datasets according to the support counts for single items has a general benefit with respect to the computational efficiency of set enumeration trees in general.

1.3 Apriori Algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database this has applications in domains such as market basket analysis.

1.3.1 Limitations of Apriori Algorithm

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all $2^{|S|} - 1$ of its proper subsets.

Later algorithms such as Max-Miner[2] try to identify the maximal frequent item sets without enumerating their subsets, and perform "jumps" in the search space rather than a purely bottom-up approach.

1.4 Existing Method

The research work was initiated through a study and analysis phase, where significant study was conducted to understand the existing system. Using Apriori algorithm for web log mining is a novel technique. The explosive growth of the World Wide Web (WWW) in recent years has turned the web into the largest source of available online data.
1. Situations like several unrelated topics in a single web page may lead to satisfy the visitors are too rigid to reach the information.
2. Understand the way user browses the site and find out which is the most frequent used link and pattern of using the features available on the site.
3. It is costly to handle a huge number of candidate sets.
4. It is difficult to repeatedly scan the database and check a large set of candidates with pattern matching, which is especially true for mining long patterns.
In order to overcome the drawback inherited in Apriori, an efficient FP-The growth based mining method is used. FP-growth, which contains two phases, where the first phase constructs an FP tree and the second phase recursively, researches the FP tree and outputs of all frequent patterns. All this information is available on the online, but hidden for the users. Presently, Apriori based approach extract this hidden information for analyzing the visitor browsing behavior.

1.5. Proposed Method

The exponential increase in the amount of Web data, automatic classification of Web pages into predefined categories is essential to support the Web directories.
Web pages can be classified by two methods: syntactic and semantic. This proposed work emphasizes syntactic classification, which uses a set pattern on a web page to classify it. This article suggests a method to classify a Web page with only minimum number of representative features or terms extracted from it without using the entire Web page. The optimum number of features is selected using a three step procedure, by filtering the features in each subsequent step. Finally, Complete FP-tree structures are illustrated in figure 4.

TABLE 1

TID	ITEMS
1	A, B, E
2	B, D
3	B, C
4	A, B, D

ISSN:2319-7900

5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

TABLE 2

ITEM	FREQUENCY	PRIORITY
B	7	1
A	6	2
C	6	3
D	2	4
E	2	5

1.5.1 FP Growth Algorithm

The FP - Growth algorithm allows mining frequent item set without candidate generation. The algorithm consists of two steps:
1. Compress a large database into a compact, for mining frequent set to build a FP-tree.
2. Develop an efficient, FP-tree-structure,extracts the frequent itemsets directly. It divides into smaller ones to avoid candidate generation [7].

The example illustrates how to find frequent items from the FP-tree. The following databases have the following transaction to find all frequent item sets using FP-Growth algorithm. Table 2 consists transactions and items, in table 3, the frequency of occurrence of each item in the databases is shown and in figure 4, all the items are ordered according to its priority.

TABLE 3

TID	ITEMS	ORDERED ITEMS
1	A, B, E	B, A, E
2	B, D	B, D
3	B, C	B, C
4	A, B, D	B, A, D
5	A, C	A, C
6	B, C	B, C
7	A, C	A, C
8	A, B, C, E	B, A, C, E
9	A, B, C	B, A, C

1.5.2. Working of FP Growth Algorithm

The proposed technique uses FP-growth to categorize the web pages. For instance, the university website may contain the web pages for student, staff, and others.
Step 1: In the First step, the support items are scanned for the transaction. If the items are minimum support they are eliminated and it will be considered as inadequate for transactions.
Step 2: Then the items are sorted in descending order of the frequency to find the frequent items. The sorted items are indexed for the transaction purpose.
These sorted items are stored in an array, which contains a pointer to the head of the list. It is easy to read the frequent items.
The sorted items are arranged in a tree format to fetch the relevant web pages. In this proposed technique the FP Growth algorithm operates in the following
Four steps to categorize the web sites.
1. Preprocessing
2. FP Tree an FP Growth
3. Association Rule Generation
4. Results
The preprocessing techniques include data selection, cleaning and transformation which of them general steps to clean, correct and complete the input data for web mining requirements.
The second step is performed in two steps.
1.FP Tree generation
2.Applying FP Growth to generate association rules. FP tree is a compact data structure that stores important, information about frequent patterns.

1.5.3 ADVANTAGES OF FP-GROWTH ALGORITHM

The major advantages of the FP - growth algorithm are,
1. Uses compact data structure.
2. Eliminates repeated database scans.
FP-growth is an order of magnitude faster than other association rule mining algorithms and is also faster than FP-tree [5, 6].The algorithm reduces the total number of candidate item sets by producing a compressed version of the database in terms of an FP-tree. The FP-tree stores relevant information and allows for the efficient discovery of frequent item sets [5, 6].

1.6. CONCLUSION

Web mining is the Data Mining type that automatically discovers or extracts the information from web documents. In this paper, An Enhanced Novel approach

ISSN:2319-7900

on Web page Categorization is discussed. Our goal of research is finding the frequent web pages as efficient as the other frequent mining rules.

REFERENCES

[1]. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal,"Web Search Using Automatic Classification," Proceedings of the 6th International World Wide Web Conference, April 1997.

[2]. J. Hou and Y. Zhang, Effectively Finding Relevant Web Pages from Linkage Information, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 4, 2003.

[3]. R. Kosala, and H. Blockeel, Web Mining Research: A Survey, SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining Vol. 2, No. 1 pp 1-15, 2000.

[4]. Arun K Pujari, "Data Mining Techniques".

[5]. Pramod S. and O.P Vyas, Survey on frequent item set mining algorithms, IJCA, volume 1.

[6]. C. Borgelt. An Implementation of the FP-growth Algorithm. Proc. Workshop Open Software for Data Mining(OSDM'05 at KDD'05, Chicago, IL), 1–5.ACMPress,New York, NY, USA 2005.

[7]. Prateek Gupta and Surendra Mishra,"Improved FP Tree algorithm with customized web log preprocessing" IJCST Vol.2.

[8]. http://en.wikipedia.org/wiki/Web_search_engine.