[IJESAT] [International Journal of Engineering Science & Advanced Technology] Volume-5, Issue-2, 144-148

Image Retrieval Using Frequent Pattern Tree

P.B.Kamdi
III sem Mtech CSE, Vidarbha Istitute of Technology, Nagpur, India.
priyanka.kamdi@gmail.com

Prof. Pravin Kullurkar,
Assistant Professor, Vidarbha Istitute of Technology, Nagpur, India.
pravinkulurkar@gmail.com

Abstract: **Image retrieval is an important topic in the field of pattern recognition and artificial intelligence. Frequent patterns are useful in many data mining problems including query suggestion. Searching or retrieving image based on its content is called content based image retrieval. In CBIR, images are indexed by their visual content, such as color, texture, shapes. Frequent patterns can be mined through frequent pattern tree (FPtree) data structure which is used to store the compact (or compressed) representation of a transaction database. Majority of existing frequent pattern mining approaches require many input parameters to be set by the users.In this paper, we propose an frequent pattern algorithm to retrieve mostly accurate the images from large database. frequent pattern mining research has substantially broadened the scope of data analysis and will have deep impact on data mining methodologies and applications in the long run.**

Keywords: **Image retrieval, FPT, CBIR**

1. INTRODUCTION

In Data Mining the task of finding frequent pattern in large databases is very important and has been studied in large scale in the past few years. Unfortunately, this task is computationally expensive, especially when a large number of patterns exist.

Currently, All of the Web based images search engines rely on textual metadata so we can't capture each and every images. Web based image produces a lot of garbage in the results because users usually enter that metadata manually which is not sufficient, expensive and may not capture every word that describes the image. On the other side, the Content Based Image Retrieval (CBIR) systems can filter images based on their visual contents (colors, shapes, textures or any other information) that can be derived from the image itself which may provide better indexing and return more accurate result. The FP-Growth Algorithm, proposed by Han in , is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the Tree Projection. In some later works it was proved that FP-Growth has better performance than other methods.

1.3 Algorithm FPT: The FP Algorithm is an alternative way to find frequent item sets , thus improving performance. For so much it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

In simple words, this algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP reduces the search costs looking for short patterns recursively and then concatenating them in the long frequent patterns, offering good selectivity.

In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partition the database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases.

The next subsections describe the FP-tree structure and FP-Growth Algorithm, finally an example is presented to make it easier to understand these concepts.

1.4 FP-Tree structure

The frequent-pattern tree (FP-tree) is a compact structure that stores quantitative information about frequent patterns in a database .

Han defines the FP-tree as the tree structure defined below:

One root labelled as "null" with a set of item-prefix sub trees as children, and a frequent-item-header table. Each node in the item-prefix sub tree consists of three fields:

a. Item-name: registers which item is represented by the node;
b. Count: the number of transactions represented by the portion of the path reaching the node;
c. Node-link: links to the next node in the FP-tree carrying the same item-name, or null if there is none.

Each entry in the frequent-item-header table consists of two fields:

1. Item-name: as the same to the node;
2. Head of node-link: a pointer to the first node in the FP-tree carrying the item-name.

1.1 Algorithm 1: FP-tree construction

Input: A transaction database DB and a minimum support threshold ?.

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows.

1. Scan the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Sort F in support-descending order as FList, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as "null". For each transaction Trans in DB do the following:

   - Select the frequent items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert tree([p | P], T).
   - The function insert tree([p | P], T) is performed as follows. If T has a child N such that N item-name = p. item-name, then increment N 's count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree (P, N) recursively.

By using this algorithm, the FP-tree is constructed in two scans of the database. The first scan collects and sort the set of frequent items, and the second constructs the FP-Tree.

1.2 FP-Growth Algorithm

After constructing the FP-Tree it's possible to mine it to find the complete set of frequent patterns. To accomplish this job, Han presents a group of lemmas and properties, and thereafter describes the FP-Growth Algorithm as presented below in Algorithm 2.

Algorithm 2: FP-Growth

Input: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold .

Output: The complete set of frequent patterns.

Method: call FP-growth(FP-tree, null).

Procedure FP-growth(Tree, a) {

(01) if Tree contains a single prefix path then // Mining single prefix-path FP-tree {

(02) let P be the single prefix-path part of Tree;
(03) let Q be the multipath part with the top branching node replaced by a null root;
(04) for each combination (denoted as ß) of the nodes in the path P do
(05) generate pattern ß ∪ a with support = minimum support of nodes in ß;
(06) let freq pattern set(P) be the set of patterns so generated;
}
(07) else let Q be Tree;
(08) for each item ai in Q do { // Mining multipath FP-tree
(09) generate pattern ß = ai ∪ a with support = ai .support;
(10) construct ß's conditional pattern-base and then ß's conditional FP-tree Tree ß;
(11) if Tree ß ≠ Ø then
(12) call FP-growth(Tree ß , ß);
(13) let freq pattern set(Q) be the set of patterns so generated;
}
(14) return(freq pattern set(P) ∪ freq pattern set(Q) ∪ (freq pattern set(P) × freq pattern set(Q)))
}

When the FP-tree contains a single prefix-path, the complete set of frequent patterns can be generated in three parts: the single prefix-path P, the multipath Q, and their combinations (lines 01 to 03 and 14). The resulting patterns for a single prefix path are the enumerations of its sub paths that have the minimum support (lines 04 to 06). Thereafter, the multipath Q is defined (line 03 or 07) and the resulting patterns from it are processed (lines 08 to 13). Finally, in line 14 the combined results are returned as the frequent patterns found.

Example:

Two Steps:

1. Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

   e.g., L={f:4, c:4, a:3, b:3, m:3, p:3}

   In the format of (item-name, support)

2. For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.

Step 1: Scan DB for the first time to generate L

TID	*Items bought*
100	***{f, a, c, d, g, i, m, p}***
200	***{a, b, c, f, l, m, o}***
300	***{b, f, h, j, o}***
400	***{b, c, k, s, p}***

500 **{*a, f, c, e, l, p, m, n*}**

Item frequency

f **4**

c

4

a **3**

b **3**

m **3**

p **3**

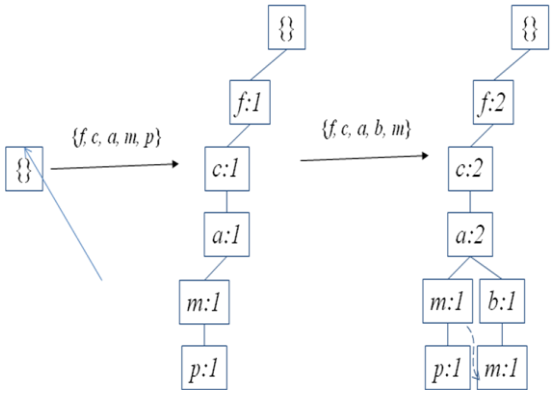
By-Product of First Scan of Database

FP-tree Example: step 2

Step 2: scan the DB for the second time, order frequent items in each transaction

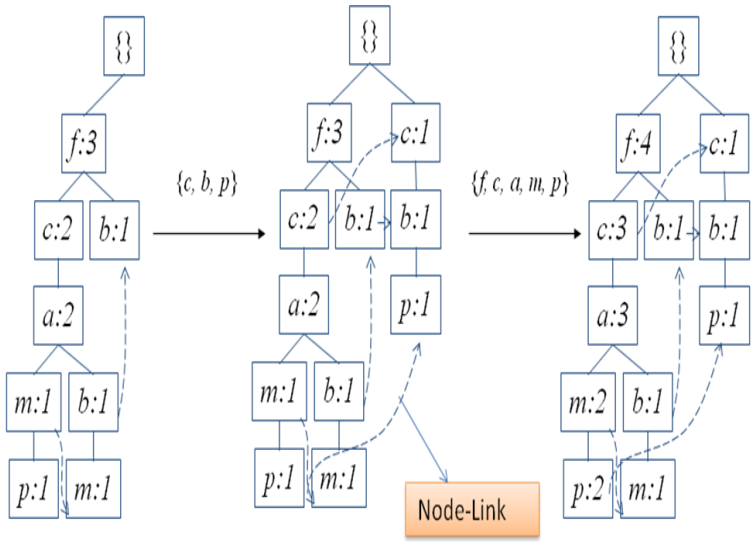
TID	*Items bought*	*(ordered) frequent items*
100	**{*f, a, c, d, g, i, m, p*}**	**{*f, c, a, m, p*}**
200	**{*a, b, c, f, l, m, o*}**	**{*f, c, a, b, m*}**
300	**{*b, f, h, j, o*}**	**{*f, b*}**
400	**{*b, c, k, s, p*}**	**{*c, b, p*}**
500	**{*a, f, c, e, l, p, m, n*}**	**{*f, c, a, m, p*}**

Step 2: construct FP-tree

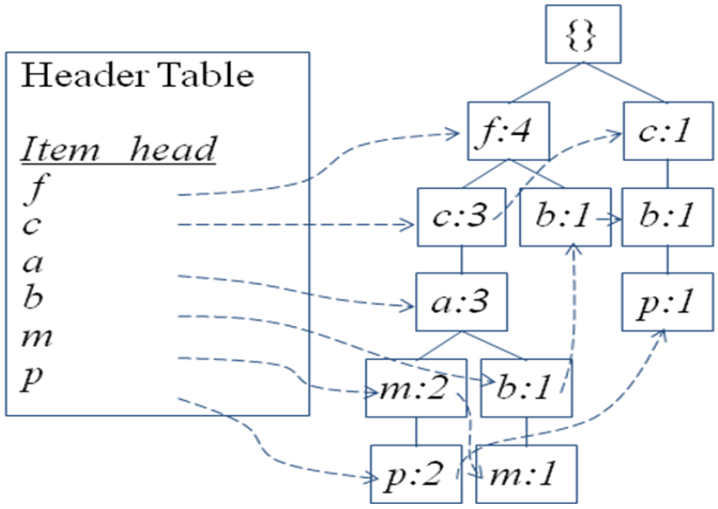


FP-tree Example: step 2

Step 2: construct FP-tree



Final FP-tree



1.5 Advantages of the FP-tree Structure

- **The most significant advantage of the FP-tree**
  - Scan the DB only twice and twice only.
- **Completeness:**
  - the FP-tree contains all the information related to mining frequent patterns (given the min-support threshold). Why?
- **Compactness:**
  - The size of the tree is bounded by the occurrences of frequent items
  - The height of the tree is bounded by the maximum number of items in a transaction

These are the investigational outcome

2. PROJECT OBJECTIVES

The objective of proposed techniques is

- To trace references of images from image database.
- To extract the images using FP-tree algorithm
- By using FPT, we can retrieve image fastley and accurately so we can reduce Complexity and increase response time..

3. INVESTIGATIONAL OUTCOME

To achieve the objective, we have proposed following techniques;

- By using Frequent Pattern Tree we retrieve the images quickly and more accurately
- Following are the snapshot for retrieving images using FPT

4. CONCLUSION

Most of the current frequent pattern mining techniques have taken for granted that users can specify the *min-sup* value easily.In this project, by using FPT retrieve the images that are semantically related by inputting the image

and text keyword . By using these two inputs we can extracted visual features of the query images .Retrieve the images in very efficient way using FPT tree algorithm and get accurate and fastley result.

5. REFERENCES

[1] Raniah A. Alghamdi, Mounira Taileb, Mohammad Ameen, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia – Jeddah, "A New Multimodal Fusion Method Based on Association Rules Mining for Image Retrieval", 17th IEEE Mediterranean Electrotechnical Conference, Beirut, Lebanon, 13-16 April 2014

[2] Goasta Grahne, Jianfei Zhu Grahne, Member, IEEE, and Jianfei Zhu, Student Member, IEEE "Fast Algorithms for Frequent Item set Mining Using FP-Trees" IEEE Transaction On Knowlegde And Data Engineering, VOL. 17, NO. 10, October 2005

[3] S. Wei , Y. Zhao , Z. Zhu , N. Liu, "Multimodal Fusion for Video Search Reranking", IEEE Transactions on Knowledge and Data Engineering, 2010, v.22 n.8, p.1191-1199.

[4] Adrien Depeursinge, Samuel Duc, Ivan Eggel and Henning M¨uller " Mobile Medical Visual Information Retrieval" IEEE Transactions on Knowledge and Information Technoloy In Biomedicine, VOL. 17, NO. 11, October 2011

[5] Agarwal R., Aggarwal, C.C., & Prasad, V.V.V. (2000). Depth first generation of long patterns.
In: *ACM Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pp 108-118

[6]. Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of
items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data
(SIGMOD'93)*, pages 207–216, Washington, DC

[7]. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In:
Proceedings of the 20th International Conference on Very Large Databases (VLDB'94),
Santiago, Chile. Morgan Kaufmann, pp 487-499

[6] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACMSIGMOD Int'l Conf. Management of Data, pp. 207-216, May 993.

[7]. S. Wu and S. McClean, "Performance prediction of data fusion for information retrieval". Information Processing, Management, 2006. 42(4): p. 899-915.

[8] Ernst, R.: "Co design of embedded systems: status and trends", Proceedings of IEEE Design and Test, April–June 1998, pp.45–54

[9]H. Müller, P. Clough, Th. Deselaers, B. Caputo, "ImageCLEF" (ser. The Springer International Series on Information Retrieval), vol. 32, pp.95 -114, 2010, Springer-Verlag.

[10] M. Ferecatu and H. Sahbi, "TELECOM ParisTech at ImageClefphoto 2008: Bi–modal text and image retrieval with diversity enhancement". In: Working Notes of CLEF 2008, Aarhus, Denmark.

[11] T. Gass, T. Weyand, T. Deselaers, and H. Ney. "FIRE in ImageCLEF 2007: Support vector machines and logistic models to fuse image descriptors for photo retrieval". In: CLEF 2007 Proceedings. Lecture Notes in Computer Science (LNCS), 2007, vol 5152. Springer, pp 492–499.

[12] R. Bellman, "Adaptive Control Process: A Guided Tour". Princeton University Press, 1961.

[13] P. K. Atrey , M. A. Hossain , A. E. Saddik and M. S. Kankanhall "Multimodal fusion for multimedia analysis: A survey", Multimedia Syst., vol. 16, no. 3, pp.1432 -1882, 2010.

[14] C. Lau, D. Tjondronegoro, J. Zhang, S. Geva, and Y. Liu, "Fusing visual and textual retrieval techniques to effectively search large collections of wikipedia images" 2007, p. 345-357.

[15] H. Frigui, J. Caudill, and A. Ben Abdallah. "Fusion of multimodal features for effiecient content-based image retrieval.", IEEE World Congress on Computational Intelligence, pp. 1992-1998, 2008.