Multi-resolution Keyword Mining Algorithm based on Frequent Pattern Technique

Xiaodong Ji

School of Computer Science and Technology, Shandong University, Shandong, China

Abstract

Finding out keywords from massive text document is a hard problem because an amount of synonyms are frequently used in a text. This paper proposes a keyword mining algorithm based on analyzing the "frequent patterns" of the words in a text. We use a new method referring the cluster process to determine the words' relationship by their distances. We firstly extend the FP-tree to a new structure named W3P-Tree which is used to maintain the text information instead of database transactions, then present our mining algorithm following with the growth of W3P-tree.

Keywords: keyword mining, frequent pattern, W3P-Tree

1. Introduction

Keyword mining is usually considered as a subproblem of Text mining which making unstructured text data available for analysis. Unlike common data mining works, text mining programs are often made up by some special text processing language or software package. But in this paper, we still take keyword mining as a common data mining algorithm.

Keyword mining is different to keyword searching [1], the latter one usually used for the search engine that analyzing the density of information on a document. However, the goal of keyword mining is to find out the most valuable information in unstructured text document. In this paper, the text is considered as a special form of data stream, thus we can construct keyword mining algorithm based on classic data mining algorithm. The most important one is frequent pattern mining technique.

Recently, many stream mining algorithms have been implemented to find the frequent patterns based on the structure of FP-tree. J.Han etc.[2] proposed FP-tree to compress and maintain frequent patterns of the data stream. C.Giannella etc.[3] improved this structure to maintain pattern frequency histories under a tilted time window framework for mining the frequent patterns with multi-time granularities. C.K.-S.Leung and Q.I.Khan in [4] proposed DS-tree to find recent frequent patterns from data stream by incrementally updating the tree with a unified single-pass. S.K. Tanbeer ect. [5] modified DS-tree to a compact form named CPS-tree. Hui Chen [6] presented RFP-tree as an extending of FP-tree to adapt on-line data stream in slide windows. While most work on frequent patterns has been concerned with structured databases, there has been little work on handling massive information that is available only in unstructured textual form.

This paper presents a keyword mining algorithm with the help of FP-tree. We need an alphabet list as an assistant input

of the text. The list contain two kinds of information: the weight of a word and the distance between two words. Section2 describes the content of alphabet. Section 3 proposes keyword mining method. Finally, some conclusions are offered.

2. Alphabet content

The alphabet is used to tell our algorithm which words are "important" in mining process. Unlike frequent pattern mining, we need to distinguish characteristic words from the other high frequency ones. We describe this attribute in a field named weight. For example, the word "the" often appears in the text but usually useless, so we give value 0 to its weight. We can also give our focus word a higher value of weight.

Another requisite information is the meaning dependence between two words. This kind of relationship is reflected in the field distance (dist.). If two words have the similar meaning, e.g. "college" and "school", there should be a shorter distance between them.

The alphabet contains two tables storing the above information. Each table could be manually entered or automatically generated by the machine learning algorithm. With the help of the alphabet, the value of characteristic words in the text is counted and the keywords are finally found out.Table1 shows an example of alphabet for words a-f.

3. Keyword Mining Algorithm

3.1. The Structure of W3P-tree

The word weight and width pattern tree (W3P-tree) is constructed with reference to the FP-tree and modified to maintain the text information. Each node in a W3P-tree consists of the following fields:

- word-id, is used to identify a word in the tree as well as in the alphabet.
- weight, is copied from the alphabet to register importance information of a word. It reflects the miner's interest.
- width, is used to incrementally sum up the frequent information of a word. The frequency count of a word in the W3P-tree also includes the emergence of its synonyms. So we call it "width" instead of "frequency". when a node is created, the width is initialized to value 1.
- node-link, is a pointer pointing to the subsequences of the node. Usually it is used to keep a tree structure.

The root node of W3P-tree is special, it has null word-id and its weight is set to the maximum value. The path from root to a node represents a class of the word's synonyms.

id	weight
a	5
b	2
c	1
d	5
e	3
f	4

(a) weight of word

	a	b	c	d	e	f
a	0	1	2	1	4	5
b	1	0	1	5	3	3
c	2	1	0	3	4	5
d	1	5	3	0	3	4
e	4	3	4	3	0	1
f	5	3	5	4	1	0

(b)dist. between words

Table 1. Example of Alphabet.

An emergent word table is used to index the W3P-tree. All words in the W3P-tree are maintained in the table. Each entry in the table has a pointer directed to the first node in W3P-tree.

After the W3P-tree is designed to capture the information of the text, the method based on W3P-tree for keyword mining could be described as follows: We firstly create an empty W3P-tree and start

to scan each word for keyword mining. As we scan the text, the information of the word is inserted into the W3P-tree and promotes its growth. After a paragraph or the whole text is completely managed, the value of each word is calculated by the product of the weight and width in the node carrying the word-id. We take that the keywords of the text carries the maximum of this kind of value.

3.2. The Growth of W3P-tree

Because of the two kinds of information endued with the words in the alphabet list, the growth of a W3P-tree is more complex than the FP-growth algorithm. It starts with setting thresholds for weight and distance. There is only a single threshold for weight named count-threshold (CT). Meanwhile, on each level j of W3P-tree, there is a distance-threshold (DT_j). Those thresholds fix the resolution of our keyword mining algorithm. With different thresholds, the algorithm could give out different mining results.

While a word E is scanned from the text, it is processed by the following steps:

(1) Search out the weight of E from the alphabet. If the weight is less than CT, or the searching process returns null result, directly drop E and proceed with the next scanning.
(2) Set a pointer "current node" to the root of the W3P-tree. The pointer is used to traverse the tree and locate the position of the new word if it is necessary.
(3) If current node is a leaf of the tree, add a new node of word E as its child; else, compare the remaining word E with the words among the children of current node in W3P- tree to find the minimum value of dist. (MD_j). It represents the nearest word to E in the area of their meaning.
(4) If MD_j is greater than DT_j, a new node of word E is inserted as another child of current node; else, set "pivot node" to the child of current node carrying the nearest word to E and execute the following judgment: If the weight of E is greater than that of the pivot node, insert in a new node of E between current node and pivot node; else, set current node to pivot node, then, if the word in pivot node is not same to E, repeat (3) and (4).
(5) Increase the width of current node and all of its ancestors. And then, go on with the next scanned word.

Our growth algorithm keeps all nodes in W3P-tree sorted in their weight descending order, that means, the weight of a node is always greater than any of its children. This kind of growing method is an equivalent to a clustering but incrementally changing the center of current class to a more valuable word from the alphabet. Each branch of the tree stands for a little cluster of words.

After all words of the text are scanned, we have got a W3P-tree containing the critical words. Figure 1 shows a W3P-tree of the following text with the alphabet shown in Table 1 where $DT_1=3$, $DT_j=1$ ($j\geq2$) and CT=2.

3.3. Value Computing and Keyword Mining

When the W3P-tree of the whole text is founded, we compute the value of each word from its index, that is, from the emergent word table. Table 2 shows the index of W3P-tree in Figure 1.The value of a word is calculated by the production of its weight and width. The word with a high value has the priority to be judged as a keyword. For example, the value of word "a" is 40, higher than that of any other words in Table 2. So "a" is taken as the keyword in the text. We can also take plural keywords in order of their values.

Another alternative of the algorithm is dividing a whole text into several paragraphs and for each paragraph

separately mining its sub-keywords using the W3P-tree. Then, all the sub-keywords are processed together to select out the global keywords. We can also give a weight to each paragraph so that to reflect our interest points to different parts of the text. In this case, we set two kinds of weight: the alphabet-weight (a.w.) and the paragraph- weight (p.w.). And then, the value of a word is computed by the following formula :

$$value = a.w. \times \sum_{\substack{\text{word in} \\ \text{each para.}}} \left(p.w. \times \sum_{\substack{\text{each node} \\ \text{of W3P-tree}}} width \right) \quad (1)$$

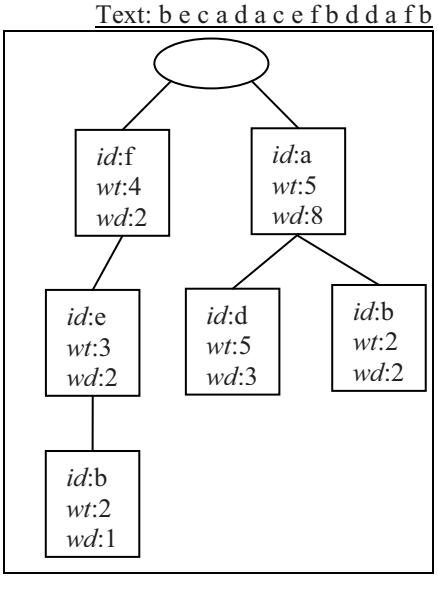

Fig. 1: W3P-tree built based on the example text, wt:short of weight and wd:short of width.

word-id	weight	width
b	2	3
e	3	2
a	5	8
d	5	3
f	4	2

Table 2: Index of the W3P-tree in Figure 1.

The resolution of our algorithm is also changed by the different paragraph-weight. For instance, if we separate the text in Figure1 into three parts and give each paragraph a weight as shown in Table 3, we will get three individual W3P-trees. According to those W3P-trees shown in Figure 2, we find the keyword in Figure 2-a is "a", in Figure 2-b is "d", in Figure 2-c is "d". At last, we find that the keyword of the text is "d".

paragraphs	p.w.
b e c a	5
d a c e f b	2
d d a f b	4

Table 3: Paragraphing of the text.

You may notice that the mining result is different to the result in Figure1. It means that the change of our focus has effect to the mining result. In this instance, we take more attention to the article structure instead of the meaning of a single word. The flexibility to user's focus is the best advantage of out algorithm.

The paragraphing algorithm has its philological significance because that a verier article usually consists of several paragraphs from the viewpoint of author and his readers, and it is unreasonable to pay same attention to different paragraphs. Generally speaking, the weight of the first and the last paragraphs are usually greater than those of the others. And the title of a paper is also suggested to have a high value of weight since keywords often appear in this area of a document.

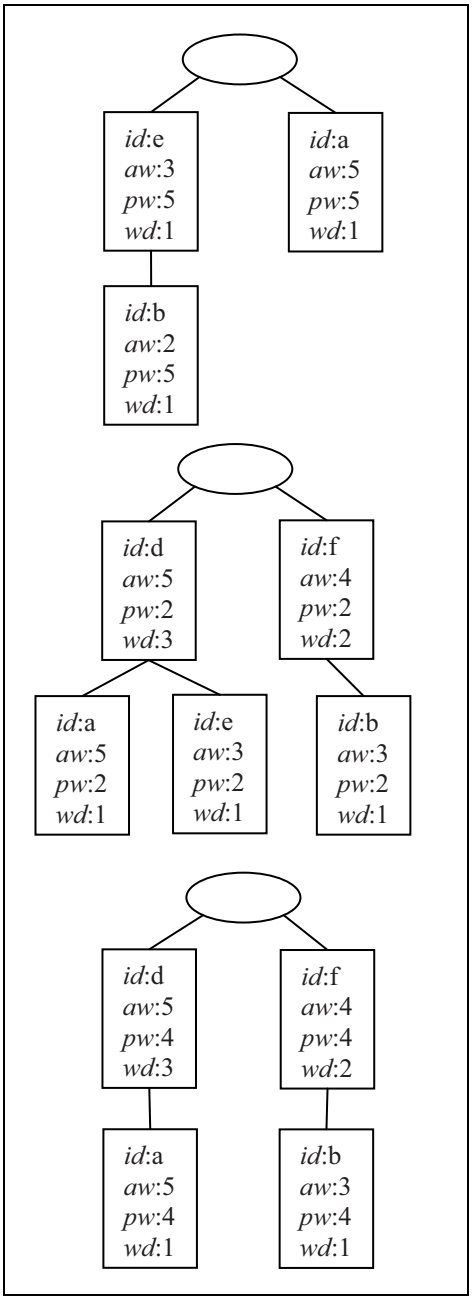

Fig. 2: W3P-trees of three paragraphs.

4. Conclusion

In this paper, we have proposed a new keyword mining algorithm. Our algorithm is extremely like the frequent pattern mining algorithms in data streams but it runs over the unstructured text document. So we have to modify the algorithm and the data structure used in it. The data structure used to maintain text information is a FP-tree like structure named W3P-tree. However, the growth algorithm of W3P-tree is not like the classical FP-growth algorithm.

Two measurements, separately called the weight and the distance (dist.), are used in the construction process of W3P-tree. And the weight of a word also includes two kinds of information, respectively from the meaning of the word and the document structure. With those information, our algorithm catches on to the importance and relationship of the reticula words and causes the growth of the W3P-tree. These kinds of information also help us to adjust the resolution of our keyword mining algorithm.

Our algorithm synthetically considers the frequent information and the interest point of the miner so that the value of a word is the combination of the two sides. In traditional condition, keywords of a text usually have the maximum of this kind of value so we mine them out.

References

[1] Patrick Reynolds , Amin Vahdat, Efficient peer-to-peer keyword searching, the ACM/IFIP/USENIX 2003 International Conference on Middleware, June 16-20, 2003

[2] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. The 2000th ACM SIGMOD International Conference of Management of Data, p.1–p.12. May 2000.

[3] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT, p.191– p.212, 2004.

[4] C. K.-S. Leung and Q. I. Khan. Dstree: A Tree Structure for the Mining of Frequent Sets from Data Streams. The 6th International Conference on Data Mining (ICDM'06), p.928–p.932. IEEE press Dec. 2006.

[5] S. K. Tanbeer, C. F. Ahmed , B.-S. Jeong, Y.-K. Lee. Efficient Frequent Pattern Mining over Data Streams, The 17th ACM conference on Information and knowledge management, Oct. 2008

[6] Hui Chen. Mining Frequent Patterns in the Recent Time Window over Data Streams. The 10th IEEE International Conference on High Performance Computing and Communications, Sept. 2008.