Asian Journal of Information Technology 7 (11): 494-497, 2008
ISSN: 1682-3915

Mining Frequent Phrase Patterns of Keywords from Text Data

[1]P.C. Saxena, [2]Asok De and [2]Rajni Jindal
[1]School of Computer and System Sciences, Jawaharlal Nehru University, Delhi, India
[2]Delhi College of Engineering, Delhi, India

Abstract: This study deals with the extraction of associations from unstructured text databases. The terms and phrases are extracted from documents and used as their keywords. The study has extended the concept of frequent patterns to grasp the relationship between phrases or terms in the document set. Our research goal is to devise an efficient algorithm that supports frequent phrase pattern discovery from large text databases.

Key words: Text mining, frequent phrase pattern, keywords, association rules

INTRODUCTION

Most of the classical data mining algorithms and techniques were mainly developed for mining information from transactional or relational data (structured data) used in financial and business market analysis, forecast etc. However, multimedia data is not very structured. Multimedia database is a database containing multimedia data which may include structured data as well as semi structured and unstructured data such as voice, video, text and images. Text and images are still media while, audio and video are continuous media. A considerable amount of data is now in multimedia format. News services provide a lot of video and audio data. Web contains large amounts of text and image data. This data has to be mined to extract useful information.

Text is the natural choice for formal exchange of information through e-mail, world wide web, Internet chat, digital libraries etc (Dijrre *et al*., 1999). There is large amount of information available in text databases in the form of electronic publication of technical and business reports, books and research articles. Such textual data is increasing with the revolution of inter net and information technology. It is therefore, crucial to develop tools for discovery of interesting knowledge from large text databases.

Text mining is currently emerging as an area of interest on its own. Text Mining is the discovery of new, previously unknown information, by automatically extracting information from different written resources. Text mining corresponds to the extension of data mining approach to textual data and deals with various tasks such as extraction of information from large collections of text databases or similarity based structuring. Despite the fact that text databases have been the major focus for many researchers around the world, text mining is still at an experimental stage and is in its infancy. The first proposal of text mining is due to Feldman and Dagan (1995).

A substantial amount of text data is available as unstructured data. One way of mining information from unstructured text databases is by extracting data or metadata from the unstructured database without any loss of key information and storing the extracted data in structured database. For example, consider a text database that has several journal articles. This text database can be converted into relational database by extracting the following attributes: author, date, publisher, title and keywords. The keywords in one article could be America, data mining and in the other article could be India, data mining. A data mining technique can be applied on this relational database to form associations from the keywords that authors from India and America are both doing research and writing articles on data mining. The other possibility is to develop data mining tools to operate directly on unstructured database. The text mining focused is a process to find out useful and meaningful association rules of co-occuring terms or words. To achieve this goal, we first introduce the basic definitions related to text mining.

BASIC DEFINITIONS

Text is considered to be composed of two basic units, the document and the term (Salton and Buckley, 1988). A document can be a traditional document which is a structured or semistructured segment of a text such as a book or journal paper which is structured in the form of number of chapters. Here, each chapter may be composed of a number of sections, subsections and paragraphs. E-mail messages, web pages, source codes, telephone directory etc. are also examples of the document.

Corresponding Author: P.C. Saxena, School of Computer and System Sciences, Jawaharlal Nehru University, Delhi, India

A document set (Singh *et al*., 1999) is defined as a set of documents representing an applications domain.

A term or phrase is defined as any string of letters or words of arbitrary length that has been extracted from unstructured components of the text including blocks of text, abstract headings, paragraphs etc. A phrase pattern Arimura *et al*. (2001) is an expression of the form (<read>, <fire on the wings>: 8). This means that the phrase <read> first appears in a document and then phrase <fire on the wings> follows within 8 words.

Given a set of terms {t1, …, tn} an association rule is of the form B => H.

$$AR: t1, \ldots, tk => tk+1, \ldots, tn$$

meaning that a document containing the terms t1 and t2 and tk also contains the terms tk+1 and tk+2 and tn.

The support of the rule is defined as the support of the B U H. It is the percentage of documents in D containing all terms in B.

PROBLEM STATEMENT

Given a document set D and a set of terms, phrases or keywords T find all the frequent phrase patterns.

A phrase pattern X is said to be frequent if its support is greater than or equal to minsup i.e., X appears in a number of documents greater than a fixed threshold denoted by minsup. The support of a phrase pattern is the percentage of documents in D containing all terms in that phrase pattern.

Frequent phrase pattern may be regarded as a generalization of frequent pattern in the transaction database such that each item of the transaction database is a phrase of arbitrary length in the text database. A phrase pattern may be ordered or it can be unordered. For every pair of phrases pa and pb the relationship r (pa, pb) measures the association degree between the 2 phrases.

MATHEMATICAL MODELING OF DOCUMENTS

Text mining generally, involves the following 2 phases:

Preparation phase: Text/document representation.

Processing phase: Mining of multiple kinds of knowledge including summarization, classification and association.

A large volume of text data is typically preprocessed into some sort of representation in a high dimension feature space which usually takes the form of a relation or a table. This study first describes an approach for the text representation and then proposes a new scheme for the storage and mining of the text feature space created during the preprocessing phase.

Text representation: Text may be represented as a document-term frequency matrix.
Let,

$$D = (d1, d2, d3 \ldots\ldots dn)$$

represents a set of N documents and

$$T = (t1, t2, t2 \ldots. tN)$$

represents a set of M terms. Each document di may be modeled as a vector

$$Vi = (vi1, vi2, \ldots. ViM)$$

in the M dimensional space RM where, vij represents same information about the occurrence of the jth term in the ith document. The individual vijs are referred to as term weights or term frequencies. The value of vij is zero if terms tj does not appear in document di. The value of vij is non zero, if term tj appears in document di and is equal to the number of times tj appears in di. Thus the entry vij is the measure of association and frequency of occurrence of the term tj with the document di.

Example 1: Let us consider a system with 5 documents and 6 terms as shown in Table 1.

The 3rd row in the matrix is the vector (40, 30, 10, 2, 0, 0) which, represents the document d3. It shows that the term key appears 40 times, SQL appears 30 times, query appears 10 times, processing appears 2 times, memory appears 0 times and Input appears 0 times, respectively in the document d3. It may be clearly noted that first 3 documents certain mainly database terms (key, SQL and query) and the last 2 documents d4-d5 contains mainly operating system terms (processing, memory and input).

Mining of text data: For the mining process the document term frequency matrix can be mapped to a relational matrix R where R(di, tj) is 1 if vij is nonzero and is between k1 and k2 and 0 otherwise. The idea is that if the keyword is occurring too frequently (>k2), it is not of interest as the document may be about this only. Also, if it is too low (k1), it is of no significance. This relation R build on the product D × T is the input to the data mining process. So,

in the 1st phase we filter out the keywords whose frequency in the document is <k1 and >k2.

Dualize and Advance algorithm (Gunopulos *et al*., 2003) was proposed to find the most specific interesting sequences from the database. But it assumes that the data will fit in the memory and does not attempt to reduce the number of times the database is scanned. But that may not be the case in practice. We propose a novel data structure Compressed and Arranged Access Pattern tree (CAAPtree) which is constructed in a single scan of the relational matrix R. The proposed algorithm can mine all the frequent phrase patterns with different support thresholds without reconstructing the tree.

The algorithm scans the document term database W and builds CAAP tree to register all count information for further mining. Then CAAPmine recursively mines the CAAP tree to find all access patterns. Once, the CAAP tree is build, it can be mined repeatedly for access patterns with different support thresholds without the need to rebuild the tree.

Algorithm: CAAPmine
Input: Document term database W
Support threshold α
Output: Complete set of frequent phrase patterns in W for α
Procedure:

- Scan W once to construct the CAAP tree using algorithm CAAP tree constructor.
- Recursively mine the CAAP tree using algorithm CAAP tree miner.

Construction of CAAP tree: Initially, the CAAP tree is empty. The root of the tree is a dummy node. Each node contains an event and a count. The event contains an event e_i from the event set E and count contains the number of occurrences of the corresponding prefix with the event e_i as the last event in the web access sequence in database W. The event in the root node is empty and its count is -1. All the nodes in the tree with the same label are linked into a list called event list. The first node of each event list is linked to the header table of the tree.

Pseudo code for CAAP tree construction

```
Algorithm: CAAP tree constructor
Input: A Web access sequence database W
Output: CAAP tree T
Procedure:
for each access sequence S ∈ W
    for each event e ∈ S
     e.count ++;
insert(S);
Procedure insert(S)
  if (children ∩ S ≠ ϕ )
       merge(S);
  else if (descendant ∩ S ≠ ϕ )
    Swap descendant;
    split child;
    merge(S);
else children ← S;
procedure merge(S)
  e.count++;
   S = S – e;
    Insert (S);
```

Table 1: Document-term frequency matrix for 5 documents and 6 terms

Id	t1	t2	t3	t4	t5	t6
d1	20	19	19	0	0	3
d2	22	10	5	0	2	0
d3	40	30	10	2	0	0
d4	0	0	1	30	12	5
d5	1	0	2	17	5	20

t1 = key, t2 = SQL, t3 = query, t4 = processing, t5 = memory, t6 = Input

Table 2: Sample document term database

Term Id	No of docs	Document ID list
T1	8	6, 1, 3, 4, 7, 19, 10, 12
T2	7	1, 2, 3, 6, 9, 10, 11
T3	5	2, 6, 5, 8, 11
T4	5	2, 3, 13, 15, 12
T5	8	1, 6, 3, 14, 9, 12, 10, 16

New access sequences are added at the root level. At each level, events of the access sequence are compared with those of children nodes. If the same events exist in both the new access sequence and that of the children nodes, the access sequence is merged with the node having the highest count. The count of the node is incremented. The remainder of the access sequence is added to the merged nodes. The process is repeated until all common events are found. The remaining events of the transaction, if any, are added as a new branch to the last merged node. It may be noted, that once the count of the new access sequence is added, the count of a descendant node can become larger than that of its ancestor. In such a case, the descendant has to swap in front of its previous ancestor. In the CAAP tree no event of the same kind could appear on the lower right hand side of another event. Here, sub trees are locally optimized to improve compression.

Example 2: Let us consider the frequency term database of Table 2 to illustrate the construction of CAAP tree.

First, the phrase pattern (6, 1, 3, 4, 7, 19, 10, 12) for termid T1 is inserted as it is to the initially empty CAAP tree. Next, phrase pattern (1, 2, 3, 6, 9, 10, 11) for termed T2 is inserted. The common events 1, 3 and 6 are identified and removed from the access sequence and merged with the existing CAAP tree. Event 10 is also common but if it is merged with the existing tree, count of 10 will become higher than its parent node. Therefore, 10 is swapped and put before node 4. The remaining events of web access sequence are inserted at node 10 as there are no more common events.

Phrase pattern mining from CAAP tree: We can find the frequent events e_i in the CAAPtree by following the event list starting from the header table. CAAP tree miner follows the conditional strategy of the pattern growth

algorithm to mine all web access patterns. But there is no need to construct the CAAP tree again for finding phrase patterns for a different support threshold, once CAAP tree is constructed. For a pattern p, a conditional CAAPtree is a tree constructed from all phrase patterns that contain pattern p. For each frequent event a conditional access sequence base is constructed denoted as S/e_i, which contains all and only all prefix sequences of e_i along with the count from the CAAP tree. The algorithm is given:

Algorithm (a): Pseudocode for CAAP tree miner

```
CAAP tree miner
Input: CAAP tree T and support threshold α
Output: a complete set of phrase patterns
Procedure CAAPtreeminer(α)
    For each event e in CAAP tree T
        Construct etree // e's conditional condensed tree
        Mine(etree, α, null)
```

Algorithm (b): Pseudocode for mine

```
Procedure mine(etree, α , p)
If etree.count > α
  Push e on p
  Pattern ← p
  Pattern.count = etree.count
  If(children ≠ϕ)
     For each event e ∈ etree
      P ← e;
      construct etree;
      Mine(etree, α, p);
      Pop p;
```

RESULTS AND DISCUSSION

We used an English text collection, called Reuters-21578 data (Arimura *et al*., 2001), which consists of news articles 27 MB on international affairs and trade trades from February to August 1997. The sample dataset is collected of unstructured texts of the total size 15.2 MB obtained from Reuters-21578 by removing all but category tags. By experiments on this data, our system could find the best 600 patterns.

Table 3 shows the list of the phrase patterns discovered by our mining algorithm which capture category train relative to other categories of Reuters newswire. Table 3a shows the list of most frequent keywords discovered by traditional frequency finding methods. The user selected the term associations. Table 3b and c shows the list of optimal patterns discovered by our method using the keyword fishes. The patterns of smallest rank (1-10) contain the topic keywords in the major news stories for the period in 1997 and patterns of medium rank 9261-2700 are long phrases.

Table 3: The list of optimal patterns

(a)		(b)		(c)	
Rank	Pattern	Rank	Pattern	Rank	Pattern
56	<platform>	11	<association>	11	<fishes><and><for>
57	<loading>	12	<fishes>	12	<fishes><pay><the>
58	<wagons>	13	<employees>	13	<fishes><employees>
59	<india>	14	<strike>	14	<fishes><on><strike>
60	<coolie>	15	<coolie>	15	<fishes><were><strike>
61	<trains>	16	<labor>	16	<fishes><><were><not><on strike>
62	<wagons>	17	<india>	17	<fishes><said>
63	<stalls>	18	<employers>	18	<fishes><talk>
64	<pantry>	19	<pantry>	19	<fishes><leaders>
65	<association>	20	<pay>	20	<fishes><pay><on>

CONCLUSION

This study presents an algorithm for mining frequent phrase patterns from the text data. First documents are represented in relational tables document x text and then CAAPmine algorithm is applied to extract frequent phrase patterns from the given document set.

REFERENCES

Arimura, Abe J., R. Fujina, H. Sakamoto, S. Shimozono and S. Arikawa, 2001. Text data mining: Discovery of important keywords in the cyberspace. In: Proceedings IEEE Kyoto ICDL, pp: 220-226.

Dijrre, J., P. Gerstl and R. Seiffert, 1999. Text mining: Finding nuggets in mountains of textual data. In: the Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, United States, pp: 398-401. ISBN: 1-58113-143-7. http://doi.acm.org/10.1145/312129.312299.

Feldman, R. and I. Dagan, 1995. Knowledge Discovery in Textual Databases (KDT). In: Proceedings of the 1st Int. Conf. Knowledge Discovery (KDD-95) Montreal, pp: 112-117.

Gunopulos, D., R. Khardm, H. Mannila, S. Saluja and H. Toivouen, 2003. Discovering all most specific sentence. ACM transactions on Database Systems, 28 (2): 140-147.

Salton, G. and C. Buckley, 1988. Term weighting approaches in automatic text retrieval. Inform. Proc. Manage., 24 (5): 513-523.

Singh, L., B. Chen, R. Haight and P. Schenermann, 1999. An algorithm for constrained association rule: Mining in semistructured data. In: Proc. 3rd Pacific-Asia Conf. PAKDD Beijing China, pp: 148-158.