International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 4, Issue 4, Jun -Jul 2015, 53-64

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

PATTERN DISCOVERY FROM DOCUMENT AND FILTERING NOISE PATTERN

ALEKYA RANI Y

Nishitha College of Engineering & Technology, Lemoor, Kandukur (Man), R.R, Telangana, India

ABSTRACT

Now a days many data mining techniques have been proposed to discover the pattern, but when a pattern is discovered noise patterns are been identified. In the previous research`s many techniques have been used to discover the pattern and implement the discovered pattern, but reducing the noise pattern`s in a document is still an open research. Over the years, people have often held the hypothesis that pattern (or phrase)-based approaches should perform better than the term-based ones, but many experiments do not support this hypothesis. In this present research an innovative technique is developed to discover a pattern in a document and filter the noise pattern`s in a document in order to improve the effectiveness of using discovered pattern for finding relevant and interesting information.

KEYWORDS: Text Mining, Text Classification, Pattern Making, Pattern Evolving, Noise Pattern Detection

INTRODUCTION

The Internet as source of knowledge provides users with access to the abundant information on current research in different areas. To provide users with documents that satisfy their needs is a goal of information retrieval. But the relevance of documents containing that information is a major issue. The relevance must be both efficient and effective since many queries may need to be processed in short time and effectively since the quality of ranking determines whether the search engine accomplishes the goal of finding relevant information. There are several kinds of documents containing the targeted information including academic publications. In order to rapidly respond to the user's query, inverted index as inherent file structure is proposed. In this structure, one term records the identifiers of the documents containing that term, frequencies and sometimes position of the term in the document.

Due to the rapid growth of digital data made available in recent years, knowledge discovery and data mining have attracted a great deal of attention with an imminent need fo turning such data into useful information and knowledge. Many applications, such as market analysis and business management, can benefit by the use of the information and knowledge extracted from a large amount of data. Knowledge discovery can be viewed as the process of nontrivial extraction of information from large databases, information that is implicitly presented in the data, previously unknown and potentially useful for users. Data mining is therefore an essential step in the process of knowledge discovery in databases.

In the past decade, a significant number of data mining techniques have been presented in order to perform different knowledge tasks. These techniques include association rule mining, frequent item set mining, sequential pattern mining, maximum pattern mining, and closed pattern mining. Most of them are proposed for the purpose of developing efficient mining algorithms to find particular patterns within a reasonable and acceptable time frame. With a large number of patterns generated by using data mining approaches, how to effectively use and update these patterns is still an open research issue. In this paper, we focus on the development of a knowledge discovery model to effectively use and update the discovered patterns and filter the noise patterns and apply it to the field of text mining.

Text mining is the discovery of interesting knowledge in text documents. It is a challenging issue to find accurate knowledge (or features) in text documents to help users to find what they want. In the beginning, Information Retrieval (IR) provided many term-based methods to solve this challenge, such as Rocchio and probabilistic models [1], rough set models [2], BM25 and support vector machine (SVM) [3] based filtering models. The advantages of term- based methods include efficient computational performance as well as mature theories for term weighting, which have emerged over the last couple of decades from the IR and machine learning communities. However, term- based methods suffer from the problems of polysemy and synonymy, where polysemy means a word has multiple meanings, and synonymy is multiple words having the same meaning. The semantic meaning of many discovered terms is uncertain for answering what users want.

Over the years, people have often held the hypothesis that phrase-based approaches could perform better than the term- based ones, as phrases may carry more "semantics" like information. This hypothesis has not fared too well in the history of IR [4], [5], [6]. Although phrases are less ambiguous and more discriminative than individual terms, the likely reasons for the discouraging performance include:

1) Phrases have inferior statistical properties to terms, 2) They have low frequency of occurrence, and 3) There are large numbers of redundant and noise phrases among them [7].

In the presence of these setbacks, sequential patterns used in data mining community have turned out to be a promising alternative to phrases [8], [9] because sequential patterns enjoy good statistical properties like terms. To overcome the disadvantages of phrase-based approaches, pattern mining-based approaches have been proposed, which adopted the concept of closed sequential patterns, and pruned nonclosed patterns. These pattern mining-based approaches have shown certain extent improvements on the effectiveness.

However, the paradox is that people think pattern-based approaches could be a significant alternative, but consequently less significant improvements are made for the effectiveness compared with term-based methods.

There are two fundamental issues regarding the effectiveness of pattern-based approaches: low frequency and misinterpretation. Given a specified topic, a highly frequent pattern (normally a short pattern with large support) is usually a general pattern, or a specific pattern of low frequency. If we decrease the minimum support, a lot of noisy patterns would be discovered. Misinterpretation means the measures used in pattern mining (e.g., "support" and "confidence") turn out to be not suitable in using discovered patterns to answer what users want. The difficult problem hence is how to use discovered patterns to accurately evaluate the weights of useful features (knowledge) in text documents.

Over the years, IR has developed many mature techniques which demonstrated that terms were important features in text documents. However, many terms with larger weights (e.g., the term frequency and inverse document frequency (tf*idf) weighting scheme) are general terms because they can be frequently used in both relevant and irrelevant information. For example, term "LIB" may have larger weight than "JDK" in a certain of data collection; but we believe that term "JDK" is more specific than term "LIB" for describing "Java Programming Language"; and term "LIB" is more general than term "JDK" because term "LIB" is also frequently used in C and C++. Therefore, it is not adequate for evaluating the weights of the terms based on their distributions in documents for a given topic, although this evaluating method has been frequently used in developing IR models.

In order to solve the above paradox, this paper presents an effective pattern discovery technique, which first

calculates discovered specificities of patterns and then evaluates term weights according to the distribution of terms in the discovered patterns rather than the distribution in documents for solving the misinterpretation problem. It also considers the influence of patterns from the negative training examples to find ambiguous (noisy) patterns and try to reduce their influence for the low-frequency problem. The process of updating ambiguous patterns can be referred as pattern evolution. The proposed approach can improve the accuracy of evaluating term weights because discovered patterns are more specific than whole documents. The rest of this paper is structured as follows: Section 2discusses related work. Section 3 provides some definitions about closed patterns, PTM and closed sequential patterns. Sections 4 and 5 propose the techniques of pattern deploying and detecting the noise pattern respectively. Finally, Section 6 gives concluding remarks

RELATED WORK

Many types of text representations have been proposed in the past. A well-known one is the bag of words that uses keywords (terms) as elements in the vector of the feature space. In [10], the tf*idf weighting scheme is used for text representation in Rocchio classifiers. In addition to TFIDF, the global IDF and entropy weighting scheme is proposed in [11] and improves performance by an average of 30 percent. Various weighting schemes for the bag of words representation approach were given in [12], [13], [14]. The problem of the bag of words approach is how to select a limited number of features among an enormous set of words or terms in order to increase the system's efficiency and avoid over fitting. In order to reduce the number of features, many dimensionality reduction approaches have been conducted by the use of feature selection techniques, such as Information Gain, Mutual Information, Chi-Square, Odds ratio, and so on. A phrase-based text representation for Web document management was also proposed in [15].

In [16], data mining techniques have been used for text analysis by extracting co-occurring terms as descriptive phrases from document collections. However, the effectiveness of the text mining systems using phrases as text representation showed no significant improvement. The likely reason was that a phrase-based method had "lower consistency of assignment and lower document frequency for terms" as mentioned in [17].

Term-based ontology mining methods also provided some thoughts for text representations. For example, hierarchical clustering [18], [19] was used to determine synonymy and hyponymy relations between keywords. Also, the pattern evolution technique was introduced in [20] in order to improve the performance of term-based ontology mining.

Pattern mining has been extensively studied in data mining communities for many years. A variety of efficient algorithms such as Apriority-like algorithms, Prefix Span, FP-tree, SPADE, SLPMiner, have been proposed. These research works have mainly focused on developing efficient mining algorithms for discovering patterns from a large data collection. However, searching for useful and interesting patterns and rules was still an open problem. In the field of text mining, pattern mining techniques can be used to find various text patterns, such as sequential patterns, frequent item sets, co-occurring terms and multiple grams, for building up a representation with these new types of features. Nevertheless, the challenging issue is how to effectively deal with the large amount of discovered patterns.

For the challenging issue, closed sequential patterns have been used for text mining in, which proposed that the concept of closed patterns in text mining was useful and had the potential for improving the performance of text mining

Table 1

Paragraphs	Terms
dp_1	t_1,t_2
dp_2	t_3,t_4,t_5
dp_3	t_3,t_4,t_5,t_6
dp_4	t_3,t_4,t_5,t_6
dp_5	t_1,t_2,t_6,t_7
dp_6	t_1,t_2,t_6,t_7

A Set of paragraphs and to improve the effectiveness by effectively using closed patterns in text mining. In addition, a two-stage model that used both term-based methods and pattern- based methods was introduced in to significantly improve the performance of information filtering.

Natural language processing (NLP) is a modern computational technology that can help people to understand the meaning of text documents. For a long time, NLP was struggling for dealing with uncertainties in human languages. Recently, a new concept-based model was presented to bridge the gap between NLP and text mining, which analyzed terms on the sentence and document levels. This model included three components. The first component analyzed the semantic structure of sentences; the second component constructed a conceptual ontological graph (COG) to describe the sematic structures; and the last component extracted top concepts based on the first two components to build feature vectors using the standard vector space model. The advantage of the concept-based model is that it can effectively discriminate between non important terms and meaningful terms which describe a sentence meaning. Compared with the above methods the concept based model usually relies upon its employed NLP techniques.

PATTERN TAXONOMY MODEL

In this paper, we assume that all documents are split into paragraphs. So a given document yields a set of paragraphs PS(d). Let D be a training set of documents, which consists of a set of positive documents, DP; and a set of negative documents, D. Let T = (t_1, t2, t3,….., tm) be a set of terms (or keywords) which can be extracted from the set of positive documents, DP.

Table 2: Frequent Patterns and Covering Set

Frequent Pattern	Covering Set
$\{t_3,t_4,t_6\}$	$\{dp_2,dp_3,dp_4\}$
$\{t_3,t_4\}$	$\{dp_2,dp_3,dp_4\}$
$\{t_3,t_6\}$	$\{ dp_2,dp_3,dp_4\}$
$\{t_4,t_6\}$	$\{ dp_2,dp_3,dp_4\}$
$\{t_3\}$	$\{ dp_2,dp_3,dp_4\}$
$\{t_4\}$	$\{ dp_2,dp_3,dp_4\}$
$\{t_1,t_2\}$	$\{dp1,dp_5,dp_6\}$
$\{t_1\}$	$\{dp1,dp_5,dp_6\}$
$\{t_2\}$	$\{dp1,dp_5,dp_6\}$
$\{t_6\}$	$\{dp_2,dp_3.dp_4,dp_5,dp_6\}$

Pattern Taxonomy Patterns can be structured into a taxonomy by using the is-a (or subset) relation. For the example of Table 1, where we have illustrated a set of paragraphs of a document, and the discovered 10 frequent patterns in Table 2 if assuming min sup ¼ 50%. There are, however, only three

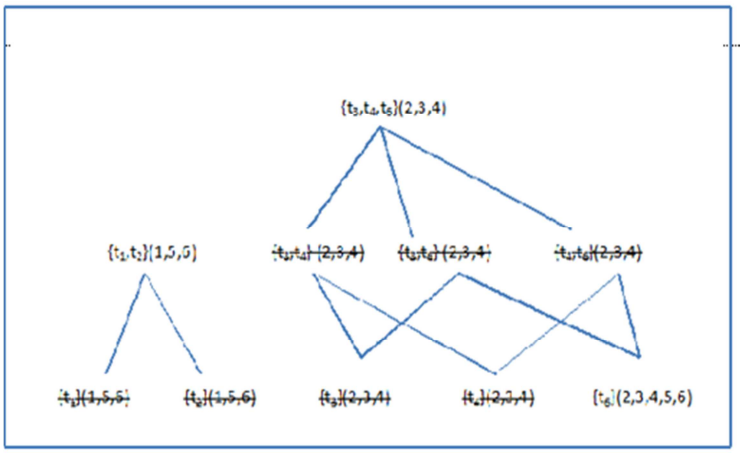

Figure 1: Pattern Taxonomy

Closed patterns in this example. They are <t_3; t4; t6>, <t_1; t2>, and <t_6>. Fig. 1 illustrates an example of the pattern taxonomy for the frequent patterns in Table 2, where the nodes represent frequent patterns and their covering sets; nonclosed patterns can be pruned; the edges are "is-a" relation. After pruning, some direct "is-a" retaliations may be changed, for example, pattern ft6g would become a direct sub pattern of (t3; t4; t6) after pruning nonclosed patterns. Smaller patterns in the taxonomy, for example pattern ft6g, (see Figure 1) are usually more general because they could be used frequently in both positive and negative documents; and larger patterns, for example pattern (t_3;t_4;t_6), in the taxonomy are usually more specific since they may be used only in positive documents. The semantic information will be used in the pattern taxonomy to improve the performance of using closed patterns in text mining.

PATTERN DEPLOYING METHOD

In order to use the semantic information in the pattern taxonomy to improve the performance of closed patterns in text mining, we need to interpret discovered patterns by summarizing them as d-patterns (see the definition below) in order to accurately evaluate term weights (supports). The rationale behind this motivation is that d-patterns include more semantic meaning than terms that are selected based on a term-based technique (e.g., tf*idf). As a result, a term with a higher tf*idf value could be meaningless if it has not cited by some d-patterns (some important parts in documents). The evaluation of term weights (supports) is different to the normal term-based approaches. In the term-based approaches, the evaluation of term weights is based on the distribution of terms in documents. In this research, terms are weighted according to their appearances in discovered closed patterns.

Representations of Closed Patterns

It is complicated to derive a method to apply discovered patterns in text documents for information filtering systems. To simplify this process, we first review the composition operation defined in [16]. Let p_1 and p_2 be sets of term-number pairs. p_1 p_2 is called the composition of p_1 and $p_2$which satisfies

$$p_1 \oplus p_2 = \{(t, x_1 + x_2) \mid (t, x_1) \in p_1, (t, x_2) \in p_2)\} \cup \{(t,x) \mid (t,x) \in p_1 \cup p_2, \text{not}((t,\_) \in p_1 \cap p_2)\}.$$

Where is the wild card that matches any number. For the special case we have $p \oplus \phi = p$; and the operands of the composition operation are interchangeable. The result of the composition is still a set of term-number pairs. For example,

$$\{(t_1, 1), (t_2, 2), (t_3, 3)\} \oplus \{t2, 4)\} = \{t1, 1), t_2, 6), t_3, 3)\}$$

$$\overline{d_i} = \{(t_{i1}, n_{i1}).(t_{i2}, n_{i2}), \ldots\ldots\ldots, (t_{im}, n_{im})\}$$

where t_{ij} in pair (t_{ij}, n_{ij}) denotes a single term and n_{ij} is its support in di which is the total absolute supports given by closed patterns that contain t_{ij}; or n_{ij} (simply in this paper) is the total number of closed patterns that contain t_{ij}. For example, using Fig. 1 and Table 1, we have

$sup_a(<t_3, t_4, t_6>) = 3$

$sup_a(<t_1, t_2>) = 3$

$$\overline{d} = \{(t_1, 3), (t_2, 3)(t_3, 3)(t_4, 3)(t_6, 8)\}$$

The process of calculating d-patterns can be easily described by using the operation in Algorithm 1 (PTM) shown in Figure 2 that will be described in the next section, where a term's support is the total number of closed patterns that contain the term. Table 3 illustrates a real example of pattern taxonomy for a set of positive documents. We also can obtain the d-patterns of the five sample documents in Table 3 which are expressed as follows:

ALGORITHM

Algorithm TDM (D^+, min_sup)

Input: positive document D+: minimum support min_sup.

Output: d-patterns dp, and supports of terms

DP=0;

foreach document $d \in D^+$ do

let PS(d) be the set of paragraphs in d;

$\overline{d}$=0;

foreach pattern $p_i \in SP$ do

$P = \{(t, 1) \mid t \in p_{ij}\}$;

$\overline{d} = \overline{d} \oplus p$;

end

DP = DP U { $\overline{d}$ }

end

$T = \{t \mid (t, f) \in p, p \in DP\}$;

foreach term $t \in T$ do

support(t)=0;

end

foreach d-pattern $p \in DP$ do

foreach (t, w) ∈ β(p) do

support (t)= support(t)+w;

end

end

Let DP be a set of d- pattern in D^+, and p ∈ DP be a d-pattern. We call p(t) the absolute support of term t, which is the number of patterns that contain t in the corresponding patterns taxonomies. In order to effectively deploy patterns in different taxonomies from the different positive documents, d-patterns will be normalized using the following assignment sentence:

$$P(t) \leftarrow p(t) * \frac{1}{\sum_{t \in t} p(t)}$$

Actually the relationship between d-patterns and terms can be explicitly described as the following association mapping[21] , a set-value function:

β:DP$\rightarrow 2^{T*[0,1]}$

Such that

β (p_i) = {(t_1, w_1), (t_2,w_2),…….,(t_k,w_k)}

for all p_i ∈ DP, where

p_i =={(t_1,f_1),(t_2,f_2),…….,(t_k,f_k)} ∈ DP , $w_i = \frac{1}{\sum_{j=1}^{k} f_j}$

and T = {t | (t,f) ∈ p,p ∈ DP}

β (p_i)is called the normal form (or normalized d-pattern) of d-pattern p_i in this paper, and termset(p_i) =(t_1;t_2;...;t_k)

FILTERING NOISE PATTERNS

Consider two documents, A and B, that have resemblance ρ. If is close to 1, then almost all the elements of SA and SB will be pairwise equal. The idea of noise filtering is to divide every sketch into k groups of s elements each. The probability that all the elements of a group are pair-wise equal is simply ρs and the probability that two sketches have r or more equal groups is

$$p_{k,s,r} = \sum_{r \le i \le k}^{n} \binom{k}{i} \rho^{s.i} (1 - \rho^s)^{k-i}$$

The remarkable fact is that for suitable choices of [k,s,r] the polynomial $P_{k,s,r}$ behaves as a very sharp high-band pass filter even for small values of k. The sharp drop-off is obvious. To use this fact, we first compute for each document D the sketch SD as before, using k·s independent permutations . (We can now bear literarily generous with the length of the fingerprints used to create shingle uid's; however 64 bits are plenty for our situation.) We then split SD into k groups of s elements and fingerprint each group. (To avoid dependencies, we use a different irreducible polynomial for these

fingerprints.) We can also concatenate to each group a group id number before fingerprinting. Now all we need to store for each document is these k fingerprints, called “features”. Because fingerprints could collide the probability that two features are equal is

$\rho^s + p_f$,

Where p_f is the collision probability. This would indicate that it suffices to use fingerprints long enough to so that pf is less than say 10−6. However, when applying the filtering mechanism to a large collection of documents, we again use the clustering process described above, and hence we must avoid spurious sharing of features. Nevertheless, for our problem 64 bits fingerprints are again sufficient. It is particularly convenient, if possible, to choose the threshold r to be 1 or 2. If r = 2 then the third phase of the merging process becomes much simpler since we don’t need to keep track of how many features are shared by various pairs of documents: we simply keep a list of pairs known to share at least one feature. As soon as we discover that one of these pairs shares a second feature, we know that with high probability the two documents are near-duplicates, and thus one of them can be removed from further consideration. If r = 1 the third phase becomes moot. In general it is possible to avoid the third phase if we again group every r features into a super-feature, but this forces the number of features per document to become (k/r).

Algorithm Description

Let the collection of documents Dn be an n-tuple made of n documents, where n is an integer number. Therefore Dn=<d1, d2, d3 …,dn> , where di | n≥I is the ith document and documents may have different sizes. Hence for each document di | n≥i, we associate its size Si such that we now have the couple (di | i<n, Si|n≥i). The tuple Dn becomes Dn=< (di | i<n,Si|n≥i)>, the total size of Dn being S = n S

Let UQ be the user query made of k strings, where k is an integer number (number of terms of the query). Therefore UQk=UQ1+ UQ2+ UQ3+ UQ4+ UQ5+…+UQk. The proposed algorithm is described as follows: we query the presence of UQh | 1≤h≤k string in every document, the first round of search having started with UQ1. If the string is in a document, then the document is kept for the next round of search of UQh+1 | 1≤h≤k. The search contains k number of rounds at most since each string must be found in each document that had been kept. Hence for each round we may keep j documents where j is an integer number less than n. The algorithm is as follows: Initially we query the presence of UQ1 for each document in Dn. For subsequent searches, let IS j | 1≤j≤n be the tuple of j documents in which the UQh | 1≤h≤k had been found at hth round. Therefore, we keep the tuple IS j | 1≤j≤n, and search UQh+1 | 1≤h≤k in it. Therefore, if the tuple dimension is not zero for all rounds, the search goes on until h=k. The string to search is divided into terms and all documents are represented by their roots in a list. The algorithm searches a particular term in all the documents, if the term is not found then the document is deleted in the list of documents having that term.

Algorithm: Input: D a set of d-patterns to search in

User query: phrase containing terms

Output: Patterns that match the query presented by order of relevance

List of D-patterns D = {d_1, d_2, d_3, d_4, ……, d_n}

List of Query terms T = {t_1, t_2, t_3, t_4, ……..,t_n}

FOR each node t_i in T

```
 Get the term t_i
  FOR each node d_i in D
   FOUND = search Term (t_i in d_i)
    IF Not FOUND THEN
      DELETE d_i in D
     ELSE
          Pos ← get the position of d_i
IF Pos is not 0 and Pos is different to d_i.position + 1
THEN
 DELETE d_i in D
         ELSE
 d_i.position ← Pos
       END IF
     END IF
   NEXT node in D
NEXT Node in T
FUNCTION search Term Parameters root: pointer to the root, term: string
BEGIN
  IF root.info == term THEN
    Return (root)
  ELSE
     IF root.info > term THEN
     Return search Term (Right Son of root, term)
    ELSE
       Return search Term (Left Son of root, term)
    END IF
  END IF
END
```

The proposed algorithm follows different steps and the number of those steps depends on the number of terms of query. We have used Vector Space Model to represent the text document. In that model a document is represented by a vector of keywords extracted from the document with associated weights representing the importance of keywords in the document within the whole documents set.

Time and Space Complexity

The time complexity is measured by the number of elementary operations carried out during execution of program and space complexity is the computer memory used by our algorithm. In our algorithm, the operations considered are the ones done by binary search on the input size n used to search term in a document. The amount of work done during a single execution before and after loop is constant. The time of our algorithm is proportional to the number of time the loop executed. It is possible to reduce running time of our algorithm by reducing the number of candidate size to search in. That leads to the increasing speed of algorithm.

Using binary search in the proposed algorithm after each iteration, the input size to search in is decreased and it is less than the one for previous iteration. That it is why the time complexity of our algorithm is logarithmic and is O (log2 n). Since each comparison binary search uses halves of the search space, search process will never use more than O (log N) comparisons to find the target term in a document. It is the same calculation for its space complexity. The more space we allocate for the algorithm, the faster it runs. The work space cannot exceed the running time. We know that writing in each memory cell requires at least a constant amount of time.

Thus if we let T(n), time complexity and S(n) space complexity of our algorithm, then S(n) = O(T(n)).The space complexity of our algorithm is then O (1).The function of our algorithm is logarithmic. As the number of documents to search in increases the time used to search in is decreased. The algorithm for filtering noise pattern is listed below.

The following graph shows the comparison of PTM and TFIDF

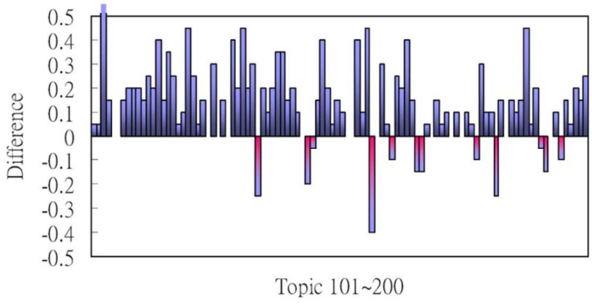

Figure 2: Comparison of PTM (IPE) and TFIDF in Top-20 Precision

RESULTS AND DISCUSSIONS

The most important information revealed is that our proposed PTM (IPE) outperforms not only the pattern mining-based methods, but also the term-based methods including the state-of-the-art methods BM25 and SVM. PTM (IPE) also outperforms CBM Pattern Matching and CBM in the five measures. For the time complexity in the testing phase, all models take O(| T | * | d |) for all incoming documents d. In our experiments, all models used 702 terms for each topic in average. Therefore, there is no significant difference between these models on time complexity in the testing phase.

The following are the experimental results

Figure 3: Uploading Text File Figure 4: Uploading a Document

Figure 5: Finding Frequent Patterns Figure 6: Discovered D-Patterns

CONCLUSIONS

We have presented a method that can discover the pattern and eliminate noise pattern from a collection of hundreds of millions of documents by computing independently for each document. Although research on detecting noise patterns is going on until now, efficiency and effectiveness for document relevance is still an issue and needs improvement. In this paper we proposed an efficient algorithm for discovering pattern detecting noise pattern by exploiting position and order of term in documents. The results show that the proposed method provides effectiveness and high efficiency by reducing the documents size to search in up to 12% and that leads to the decreased computation time in partial document detection. In future we intend to investigate compression methods in our method for the query efficiency.

ACKNOWLEDGMENTS

I wish to thank my Husband who supported me in finding an efficient algorithm for discovering a pattern and filtering noise pattern some essential ideas behind the resemblance definition and computation were developed in conversations with my professors.

REFERENCES

1. R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval.Addison Wesley, 1999.
2. Y. Li, C. Zhang, and J.R. Swan, “An Information Filtering Model on the Web and Its Application in Jobagent,” Knowledge-Based Systems, vol. 13, no. 5, pp. 285-296, 2000.
3. S. Robertson and I. Soboroff, “The Trec 2002 Filtering Track Report,” TREC, 2002, trec.nist.gov/pubs/trec11/papers/OVER. FILTERING.ps.gz.
4. D.D. Lewis, “Feature Selection and Feature Extraction for Text Categorization,” Proc. Workshop Speech and Natural Language, pp. 212-217, 1992.
5. S. Scott and S. Matwin, “Feature Engineering for Text Classification,” Proc. 16th Int’l Conf. Machine Learning
6. (ICML ’99), pp. 379-388, 1999. 1, pp. 1-47, 2002.
7. N. Jindal and B. Liu, “Identifying Comparative Sentences in Text Documents,” Proc. 29th Ann. Int’l ACM SIGIR

Conf. Research and Development in Information Retrieval (SIGIR ’06), pp. 244-251, 2006.

8. S.-T. Wu, Y. Li, and Y. Xu, “Deploying Approaches for PatternRefinement in Text Mining,” Proc. IEEE Sixth Int’l Conf. DataMining (ICDM ’06), pp. 1157-1161, 2006.
9. S.-T. Wu, Y. Li, Y. Xu, B. Pham, and P. Chen, “Automatic Pattern- Taxonomy Extraction for Web Mining,” Proc. IEEE/WIC/ACM Int’l Conf. Web Intelligence (WI ’04), pp. 242-248, 2004.
10. X. Li and B. Liu, “Learning to Classify Texts Using Positive and Unlabeled Data,” Proc. Int’l Joint Conf. Artificial Intelligence (IJCAI ’03), pp. 587-594, 2003.
11. S.T. Dumais, “Improving the Retrieval of Information from External Sources,” Behavior Research Methods, Instruments, and Computers, vol. 23, no. 2, pp. 229-236, 1991.
12. K. Aas and L. Eikvil, “Text Categorisation: A Survey,” Technical Report Raport NR 941, Norwegian Computing Center, 1999
13. T. Joachims, “A Probabilistic Analysis of the Rocchio Algorithm with tfidf for Text Categorization,” Proc. 14th Int’l Conf. MachineLearning (ICML ’97), pp. 143-151, 1997.
14. G. Salton and C. Buckley, “Term-Weighting Approaches in Automatic Text Retrieval,” Information Processing and Management: An Int’l J., vol. 24, no. 5, pp. 513-523, 1988.
15. R. Sharma and S. Raman, “Phrase-Based Text Representation for Managing the Web Document,” Proc. Int’l Conf. InformationTechnology: Computers and Comm. (ITCC), pp. 165-169, 2003.
16. H. Ahonen, O. Heinonen, M. Klemettinen, and A.I. Verkamo, “Applying Data Mining Techniques for Descriptive Phrase Extraction in Digital Document Collections,” Proc. IEEE Int’l Forum on Research and Technology Advances in Digital Libraries (ADL ’98), pp. 2-11, 1998.
17. D.D. Lewis, “An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task,” Proc. 15th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR ’92), pp. 37-50, 1992.
18. A. Maedche, Ontology Learning for the Semantic Web. Kluwer Academic, 2003.
19. C. Manning and H. Schu¨ tze, Foundations of Statistical Natural Language Processing. MIT Press, 1999.
20. J.S. Park, M.S. Chen, and P.S. Yu, “An Effective Hash-Based Algorithm for Mining Association Rules,” Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’95), pp. 175-186, 1995.
21. R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” Proc. 20th Int’l Conf. Very Large Data Bases (VLDB ’94), pp. 478-499, 1994.