Automatic Pattern-Taxonomy Extraction for Web Mining

Sheng-Tang Wu Yuefeng Li Yue Xu Binh Pham Phoebe Chen*
School of Software Engineering and Data Communications
Queensland University of Technology, QLD 4001, Australia
*School of Information Technology Deakin University, VIC 3125, Australia**
E-mail: st.wu@student.qut.edu.au, {y2.li, yue.xu, b.pham}@qut.edu.au, phoebe@deakin.edu.au*

Abstract

In this paper, we propose a model for discovering frequent sequential patterns, phrases, which can be used as profile descriptors of documents. It is indubitable that we can obtain numerous phrases using data mining algorithms. However, it is difficult to use these phrases effectively for answering what users want. Therefore, we present a pattern taxonomy extraction model which performs the task of extracting descriptive frequent sequential patterns by pruning the meaningless ones. The model then is extended and tested by applying it to the information filtering system. The results of the experiment show that pattern-based methods outperform the keyword-based methods. The results also indicate that removal of meaningless patterns not only reduces the cost of computation but also improves the effectiveness of the system.

1. Introduction

Web mining, when looked upon in data mining terms, is to cluster, associate and analyze the information from the Web data sources. Due to the rapid growth of Web data and the increasing need of a more sensible and rational search system, Web mining has gained an important status in the data mining field.

It is obvious that a Web mining system would be valuable once it is capable of quickly and accurately responding to the users' needs. However, the process of mining useful patterns from a large size of Web database takes a long time. Furthermore, there is only a small amount of input data that are related to a certain user or a group of users. As a result, it is necessary to quickly filter out the most irrelevant data first, further process the filtered data and then return the most relevant documents that fit the users' needs. According to this new methodology, a Web mining system will include two phases: *filtering* and *sophisticated data processing*. '*Filtering'* is to filter out those most irrelevant documents which are retrieved from the search systems, and its aim is to speed up the process of text extraction. Whereas, the '*sophisticated data processing'* phase is to overcome the problem of mismatch by adopting diverse mining techniques in order to achieve the effectiveness of a Web mining system.

Many term-based filtering models have been presented before (see [6] [16] [13] [19]). In this paper, we mainly discuss the phase of sophisticated data processing. We put the emphasis on the improvement of accuracy of a Web mining system, which means the relevance of the retrieved documents to the users' needs is much more concerned. Different from the filtering phase which is based on 'term' or 'keyword', the phase of '*sophisticated data processing*' has utilized a sort of sequential patterns, called phrases, to do the text processing.

Association mining has been used in Web text mining, which refers to the process of searching through unstructured data on the Web and deriving meanings from it [8] [11]. The main purposes of text mining include association discovery, trends discovery, and event discovery [5]. The association between a set of keywords and a predefined category (e.g., a term) can be described as an association rule. The trends discovery means the discovery of phrases, a sort of sequential association rules. The event discovery is the identification of stories in continuous news steams. Usually clustering based mining techniques are used for such purpose. It is also necessary to combine association rule mining with the existing taxonomies in order to determine useful patterns [7] [4].

Another worth-mentioned issue is that comparing to probabilistic models, data mining-based Web mining models do not use the term independent assumption [3] [14]. In addition, Web mining models try to discover some unexpected useful data [15] [5]. It is not very difficult for the discovery of phrases from documents if we view each paragraph as a transaction. The problem, we especially concern in this paper, is how to represent the relationships between phrases in order to use the phrase effectively. One interesting method for this problem is to use a document index graph (DIG) [10], where each node is a unique word, and each edge is a two adjacent nodes which appear successive in a document. The drawback of this method is that a DIG may index some nonsense phrases.

In this research we use sequential patterns [1] to represent phrases. In order to create correct phrase

Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)
0-7695-2100-2/04 $ 20.00 IEEE

taxonomy for representation of discovered knowledge, we introduce a concept for describing the relation between discovered phrases, which is similar to the notion of closed sequential patterns [23] [24]. A frequent sequential pattern mining algorithm is then presented and experiments are also conducted in order to evaluate our results.

The rest of this paper is structured as follows: the problem formulation is stated in Section 2. Followed is the description of the pattern taxonomy model in Section 3. Then in Section 4, the results of experiment are evaluated. We finally conclude and summarize this study in Section 5.

2. Problem formulation

Instead of the keyword-based concept used in the traditional document representation model, the pattern-based model containing frequent sequential patterns (single term or multiple terms) is used to perform the same concept of task. This section will define the basic problem of mining sequential pattern in text documents.

2.1. Basic definition

The basic definition of sequences used in this study is described as follows. Let $T = \{t_1, t_2, \ldots, t_k\}$ be a set of all terms, which can be viewed as keywords in text datasets. A sequence $S = \langle s_1, s_2, \ldots, s_n \rangle$ ($s_i \in T$) is an ordered list of terms. A sequence $\alpha = \langle a_1, a_2, \ldots, a_n \rangle$ is a sub-sequence of another sequence $\beta = \langle b_1, b_2, \ldots, b_m \rangle$, denoted by $\alpha \sqsubseteq \beta$, if there exist integers $1 \le i_1 < i_2 < \ldots < i_n \le m$, such that $a_1 = b_{i1}$, $a_2 = b_{i2}, \ldots, a_n = b_{in}$. The sequence α is a *proper* sub-sequence of β if $\alpha \sqsubseteq \beta$ but $\alpha \neq \beta$, denoted by $\alpha \sqsubset \beta$. For instance, sequence ⟨A, C⟩ is a sub-sequence of sequences ⟨A, B, C⟩. However, ⟨B, A⟩ is not a sub-sequence of ⟨A, B, C⟩ since the order of terms is considered. In addition, we also can say sequence ⟨A, B, C⟩ is a super-sequence of ⟨A, C⟩. The problem of mining sequential patterns is to find the complete set of sub-sequences from a set of sequences whose support is greater than a user predefined threshold, *min_sup*.

Definition 2.1 (absolute and relative support) Given a document $d = \{S_1, S_2, \ldots, S_n\}$, where S_i is a sequence representing a paragraph in d. Let P be a sequence. We call P a sequential pattern of d if there is a $S_i \in d$ such that $P \sqsubseteq S_i$. The **absolute support** of P, denoted as $supp_a(P) = |\{ S \mid S \in d \wedge P \sqsubseteq S \}|$, is the number of occurrences of P in d. The **relative support** of P is the fraction of paragraphs that contain P in document d, denoted as $supp_r(P) = supp_a(P) / |d|$.

For example, the sequential pattern $P = \langle A, B, C \rangle$ in the sample database (Table 1) has $supp_a(P) = 2$ and $supp_r(P) = 0.5$ for the document in Table 1.

Table 1. Sequences in a document

Paragraph ID	Sequence
1	< A, B, C, D >
2	< B, D, E, C >
3	< C, F, A >
4	< E, A, B, G, C >

Definition 2.2 (frequent sequential patterns) A sequential pattern P is called **frequent sequential pattern** if $supp_r(p)$ is greater than or equal to a minimum support (*min_sup*) ξ.

For example, let *min_sup* $\xi = 0.75$ for the document shown in Table 1; we can obtain four frequent sequential patterns: ⟨B, C⟩, ⟨A⟩, ⟨B⟩, and ⟨C⟩ since their relative supports are not less than ξ.

The purpose of using *min_sup* in our model is to reduce the number of patterns discovered in a large document. Otherwise these patterns with lower relative support will increase the burden of the training. Removing less significant patterns will save much computation time without affecting the performance very much. An example is given in Section 5 (see Table 2) to show that only a small amount of frequent sequential patterns left after using *min_sup*.

Definition 2.3 (maximum sequential patterns) A frequent sequential pattern P is a **maximal sequential pattern** if there exists no frequent sequential pattern P' such that $P \sqsubset P'$ and $supp_a(P) = supp_a(P')$.

For instance, the nodes in Figure 1 represent sequential patterns extracted from Table 1. Only the patterns within the dash-line borders are maximum sequential patterns if *min_sup* $\xi = 0.50$.

Definition 2.4 (*nTerms* pattern) The length of sequential pattern P, denoted as $len(P)$, indicates the number of words (or terms) contained in P. A sequential pattern which contains n terms can be denoted in short as ***nTerms* pattern**.

For instance, given pattern $P = \langle B, C \rangle$, we have $len(P) = 2$, and P is a *2Terms* pattern. Although a sequential pattern consists of several terms (words), *1Term* pattern is a sort of special *nTerms* pattern in this study.

In this research, we concentrate on applying data mining techniques on the area of Web mining. We discover frequent sequential patterns from a text document collection (a training set) and generate a pattern taxonomy model (PTM), which illustrates the relationship between patterns. The details of PTM are described in Section 3. We then utilize the knowledge extracted from pattern taxonomy to replace the concept used in keyword-based document representation model.

3. Pattern taxonomy model

In this paper, we present a new pattern-based model PTM (Pattern Taxonomy Model) for the representation of text documents. Pattern taxonomy is a tree-like structure that illustrates the relationship between patterns extracted from a text collection. An example of pattern taxonomy is shown in Figure 1. The arrow in this figure indicates the sub-sequence relation between patterns. For example, pattern ⟨A, B⟩ is a sub-sequence of pattern ⟨A, B, C⟩, and pattern ⟨B⟩ is a sub-sequence of pattern ⟨B, C⟩. The root of the tree in the bottom level represents one of the longest patterns (i.e., maximum sequential patterns). Once the tree is constructed, we can easily find the relationship between patterns. The next step is to prune the meaningless patterns in the pattern taxonomy.

Definition 3.1 (closed relation) A frequent sequential pattern P_1 is a ***closed*** *pattern* of P_2 if P_2 is a frequent sequential pattern, $P_1 \sqsubseteq P_2$, and $supp_a(P_1) - supp_a(P_2) = 0$.

The objective of pruning phase is to eliminate the meaningless patterns. As can be seen in the Figure 1, pattern ⟨A, B⟩ is a closed pattern of ⟨A, B, C⟩. That means they always appear in the same paragraph. Therefore, the shorter one (i.e., pattern <A, B>) is negligible and is considered as a meaningless pattern. We keep the longer one since it is more meaningful and carry more information than the shorter one. Thus, after the pruning phase, only the significant patterns remain in the pattern taxonomy. An example is given in Section 5 (see Table 2, the last column) to show the number of frequent sequential patterns left after applying pruning scheme.

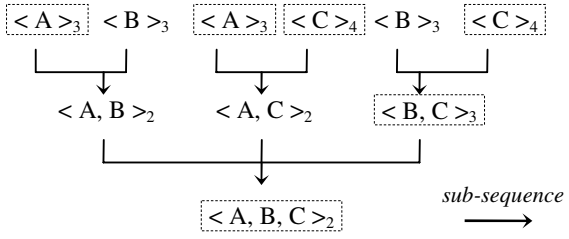

Figure 1. Partial pattern taxonomy for the sample database, those patterns within the dash-line borders are representative

The "pattern" used as a word or phrase in this paper is extracted from the text documents. The algorithm that performs the extraction of frequent sequential patterns is shown in Algorithm 3.1. Two parameters are needed for the method 'SPMining'. The first parameter *PL* is a list of *nTerms* frequent sequential patterns, which are generated from its previous recursions. The second parameter is *min_sup*, the predefined minimum relative support. Since the algorithm is a recursive function, the initial value of *PL* is the *1Term* frequent patterns. For example, {⟨A⟩, ⟨B⟩, ⟨C⟩, ⟨D⟩, ⟨E⟩} is the initial value of the parameter *PL* for the set of sequences in Table 1.

Once obtaining the parameter *PL* from either the initial value or the previous recursions, we then get into the pruning step in the algorithm. The aim of this step is to eliminate the meaningless patterns using the closed relation. As shown in Figure 2, the patterns ⟨B⟩, ⟨D⟩, ⟨E⟩, ⟨A, B⟩, and ⟨A, C⟩ are pruned since all of them are closed patterns of their super-patterns (fathers). For instance, pattern ⟨B⟩ has four of super-patterns and there exists pattern ⟨B, C⟩ whose support is the same as pattern ⟨B⟩'s. The dash-line arrows in Figure 2 indicate that the linked patterns have the closed relation.

The followings are the definitions of two operations used in the algorithm: sequence extension and p-projected database.

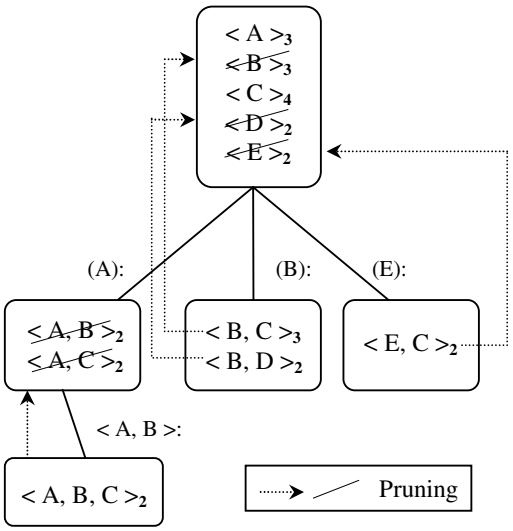

Figure 2. Illustration of pruning meaningless patterns

Definition 3.2 (sequence extension) Given a term *t* and a sequence *S*, the **sequence extension** of *S* with term *t* simply appends *t* to *S* and generates a sequence *S′*, denoted as $S' = S \bowtie t$.

For instance, the sequence extension of ⟨A, C⟩ with term B is sequence ⟨A, C, B⟩.

Definition 3.3 (*p*-projected database) Given a pattern *p*, a ***p*-projected database** contains a set of sequences made of postfixes of *p*.

For example, in the sample database in Table 1, let *p* be ⟨A⟩, the *p*-projected database will be {⟨B, C, D⟩, ⟨⟩, ⟨⟩, ⟨B, G, C⟩}. If *p* does not appear in a paragraph or the last term of *p* locates at the end of a paragraph, an empty sequence will be generated in the database.

Algorithm 3.1: SPMining(*PL*, *min_sup*)
Input: the list of *nTerms* frequent sequential patterns *PL*; The minimum support threshold *min_sup*. (Notice: in the beginning, *SP* is the set of *1Terms* frequent sequential patterns.)
Output: a set of frequent sequential patterns *SP*.
Method:

1) $SP = SP - \{P_a \in SP \mid \exists P_b \in PL$ such that
   $len(P_a) = len(P_b) - 1 \wedge P_a \sqsubset P_b \wedge$
   $supp_a(P_a) = supp_a(P_b)\}$ /* pruning */
2) $SP \leftarrow SP \cup PL$ /* add found patterns */
3) $PL' \leftarrow \{\varnothing\}$ /* PL': set of *(n+1)Terms* frequent sequential patterns */
4) **foreach** pattern *p* in *PL* **do begin**
5) generate *p*-projected database *PD*
6) **foreach** frequent term *t* in *PD* **do begin**
7) $P' \leftarrow p \bowtie t$ /* P': set of *(n+1)Terms* sequential candidates */
8) **if** $supp_r(P') \geq min\_sup$ **then**
9) $PL' \leftarrow PL' \cup P'$
10) **end if**
11) **end for**
12) **end for**
13) **if** $|PL'| = 0$ **then**
14) **return** /* no more patterns found */
15) **else**
16) **call** SPMining(PL', *min_sup*)
17) **end if**
18) output frequent sequential patterns in *SP*

After pruning, the *nTerms* frequent sequential patterns found from the previous recursion are stored (step 2 in Algorithm 3.1) and then the algorithm starts to mine for the *(n+1)Terms* patterns (step 3-12) from the projected database. If the relative supports of the *(n+1)Terms* patterns are greater than or equal to *min_sup*, we will store those patterns (i.e., frequent *(n+1)Terms* patterns). We repeat SPMining recursively if there exists at least one frequent *(n+1)Terms* pattern and pass them as the value of the first parameter; otherwise, the program returns. As a result, the output of the algorithm is the set of the frequent maximum sequential patterns. We can then simply skip the step 1 in the algorithm to find all the frequent sequential patterns without pruning.

SPMining adopts the concept of projected database method for extracting frequent sequential patterns from a document. The main difference between SPMining and others [17] [23], which adopt the same concept, is that SPMining deals with several sequences at a time, whereas others only handle one sequence at a time.

4. Application

For the testing purpose, we apply the PTM on the user profile filtering task, which has mentioned in Section 1. For each topic, the system aims to filter out non-relevant incoming documents according to the user profiles. The system extracts knowledge from a training set for the user profiles, which consist of both relevant documents and irrelevant documents. Before starting to mine for patterns, data preprocessing is necessary in order to improve the efficiency. Removing stopwords and term stemming are adopted according to a given list of stop words and the Porter Stemming algorithm [18]. Feature selection is based on the term's *tf*idf* (please see details in Section 5) value. There are two phases in the application, i.e. training and testing phases. The task of training phase is to find all frequent sequential patterns from entire training set and prune meaningless patterns using PTM for a certain topic. Once the patterns, which represent the user profiles (or topic), are obtained using PTM from the training set, a feature vector (i.e., centroid) is used to hold the representation of the context of the topic, which consists of representative patterns extracted from the training set. Given a discovered pattern *P*, the value of pattern *P* in the centroid is computed by the following weight function:

$$W(P) = \frac{\left|\left\{d_a \middle| d_a \in D_{rel}, P \ in \ d_a\right\}\right|}{\left|\left\{d_b \middle| d_b \in D, P \ in \ d_b\right\}\right|}$$

where d_a and d_b denote the documents, D denotes the training set, and $D_{rel} \subseteq D$ denotes the set of relevant documents in D. Once the centroid for a topic is obtained, we can test the system in the testing phase to find the most relevant documents related to the topic from the test set by ranking each of them. The higher the positions of the ranked documents, the more relevant they are with respect to the topic. We use a simple way to estimate the similarity between a test document and a centroid by summing the weights of patterns which appear in the document.

5. Experimental evaluation

In order to evaluate the effectiveness of the pattern-based method proposed in this paper, we present the experimental results of using the pattern-based method in comparison with the keyword-based method. For keyword-based method, we elected to use two popular information retrieval techniques for calculation of term weights: *tf*idf* (Term Frequency Inverse Document Frequency) and *Pr* (Probabilistic) schema.

The definition of *tf*idf* is described as follows. Let D_{rel} be a set of relevant documents, $D_{rel} = \{d_1, d_2, \ldots, d_n\}$ ($d_i \in D$), which contains documents with positive judgment from a training set of a certain topic (i.e., these documents are relevant to the topic). The term frequency $TF(d, t)$ is the number of times term (word) *t* occurs in document *d* ($d \in D_{rel}$) and the document frequency $DF(t)$ is the number of documents in which term *t* occurs at least once. The inverse document

frequency $IDF(t)$ is denoted by $log\ (\ |\ D\ |\ /\ DF(t)\ )$, which scores low if term t occurs in many documents and scores high if it occurs in few documents. The weight of a term t then can be represented by *tf*idf* value which is calculated as $W(t) = TF(d, t) \cdot IDF(t)$.

Pr [9] is another scheme we chose for keyword-based method. Given a term t, the weight of t is calculated by the following formula:

$$W(t) = \log \frac{\frac{r}{R-r}}{\frac{n-r}{(N-n)-(R-r)}}$$

where N is the total number of documents in the training set, R is the number of relevant documents, n is the number of documents which contain t, and r is the number of relevant documents which contain t (see [9]).

The following methods are used in our experiment:

- ***tfidf*** : the keyword-based model which using *tf*idf* scheme to calculate term weights
- ***Pr***: another keyword-based model which using keywords to represent the profile of documents
- ***PTM-1***: PTM model with *min_sup* = 0.2
- ***PTM-2***: PTM model with *min_sup* = 0.2 and with pruning

5.1. Real world datasets

In an attempt to evaluate our algorithm, TREC (Text Retrieval Conference) data collection[1] was used as the benchmarks. The version of this data collection we chose is Reuters Corpus Volume 1 (RCV1)[2], which includes 806,791 news stories. The English language stories are produced by Rueters journalists for the period between 20 August 1996 and 19 August 1997. These documents are formatted using a consistent XML schema.

TREC has developed and provided 100 topics for the filtering track aiming at building a robust filtering system [20]. The first 50 were composed by human researchers and the rest by intersecting two Reuters topic categories. Each topic is divided into two sets - training and test, and the relevance judgments have also been given for each topic. The training set has a total amount of 5,127 news stories with dates up to and including 30 September 1996 and the test set contains 37,556 news stories from the rest of news of the collection. As mentioned above, stories in both sets are assigned to either positive or negative. 'Positive' means the story is relevant to the assigned topic; otherwise 'negative' will be shown. We chose ten topics (topic 110, 120,…, 200) for our evaluation. Table 2 shows the number of documents of each topic.

[1] http://trec.nist.gov/

[2] http://about.reuters.com/researchandstandards/corpus/

5.2. Results evaluation

The measures used for evaluating experimental results are *precision*/*recall* (*P/R*) breakeven points and the precision of *top-20* returned documents. The *precision* is the fraction of retrieved documents that are relevant to the topic, and the *recall* is the fraction of relevant documents that have been retrieved. These two measures are denoted by the following formulas:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

where *TP* (true positives) is the number of documents the system correctly identifies as positives; *FP* (false positives) is the number of documents the system falsely identifies as positives; *FN* (false negatives) is the number of relevant documents the system fails to identify.

Table 2. Number of patterns for ten topics with different constraints applied in the test set

Topic	# docs	# frequent sequential patterns		
		$\xi = 0$	$\xi = 0.2$	$\xi = 0.2$ & *pruning*
110	491	9,977	5,784	5,252
120	415	5,395	3,933	2,959
130	307	4,128	1,948	1,845
140	432	16,688	4,007	3,227
150	371	8,492	5,022	3,646
160	199	4,032	2,060	1,929
170	507	12,239	6,649	4,745
180	426	26,098	2,023	1,794
190	337	4,382	2,780	2,085
200	277	3,227	1,996	1,251
Total	3,762	94,658	36,202	28,733
Avg. P/R		0.409	0.406	0.443

(where ξ is a *min_sup*)

Therefore, the higher the figures of both precision and recall curves, the more effective the system is. The *P/R* breakeven point indicates the value at which precision equals recall. The larger the measure scores, the better the system performs. The precision of *top-K* returned documents is another important measure, which refers to the relative value of relevant documents in the first *K* returned document.

The results of *P/R* breakeven point of ten topics are listed in Table 2, which shows that the improvement is achieved by using PTM with pruning. As the *min_sup* ξ increases, the average value of *P/R* breakeven point of ten topics reduces slightly from 0.409 to 0.406. This

means that the effects of these pruned patterns are not significant since their supports are relatively smaller than the remained patterns. The performance is obviously enhanced by applying the pruning scheme as we can see the measure of *P/R* breakeven value increased from 0.406 to 0.443. This is caused by the fact that the noises of meaningless patterns are reduced as they are pruned. By using combination of *min_sup* and pruning scheme as constraints, we can not only save computation time but also improve the accuracy.

Table 3 shows the precision of *top-20* returned documents on each topic. It is obvious that the *PTM-2* outperforms other three models as far as the average *P/R* breakeven point concerned. However, the measure of *PTM-1* is even lower than the result of the keyword-based model *Pr*. The reason is that the '*overfitting'*, caused by the meaningless patterns of *PTM-1,* influences the effectiveness of the system. Thus, in order to enhance the performance, the pruning scheme is required to be a part of the pattern-based models.

Table 3. Precision/Recall breakeven point for the ten topics

Topic	*tfidf*	*Pr*	*PTM-1*	*PTM-2*
110	0.161	0.226	0.419	0.548
120	0.519	0.494	0.570	0.551
130	0.063	0.063	0.063	0.313
140	0.328	0.388	0.373	0.284
150	0.130	0.222	0.111	0.167
160	0.796	0.815	0.759	0.778
170	0.411	0.384	0.493	0.397
180	0.569	0.611	0.472	0.486
190	0.494	0.553	0.529	0.588
200	0.314	0.372	0.267	0.314
Avg.	0.379	0.413	0.406	0.443

The comparison of average precision of *top-20* on ten topics is listed in Table 4. The measures of keyword-based models (*tfidf* and *pr*) are around the number of 0.4. Whereas the figures of pattern-based models (*PTM-1* and *PTM-2*) are obviously increased to be over 0.5. This implies that both PTM models improve the precision of the top returned documents.

Table 4. Average precision of top20 ranked documents on the ten topics

	tfidf	*Pr*	*PTM-1*	*PTM-2*
Avg.	0.400	0.406	0.505	0.515

Figure 3 illustrates the effect of introducing the pattern-based methods on the *P/R* curves on the topic 110, which can represent the trend of all topics. In Figure 3, as the PTM models are applied, their *P/R* curves lift up and indicate the improvement of performance made by the PTM models.

6. Related work and conclusion

Mining sequential patterns has been extensively studied in data mining community since the first research work in [1]. The earlier studies which focused on the large size of retail datasets have developed several Apriori-like algorithms in order to solve the problem of discovering sequential patterns from such databases. However, the Apriori-like algorithms only perform well in databases consisting of short frequent sequences [17]. This is caused by the fact that it is quite time-consuming to generate *nTerms* sequences candidates from *(n-1)Terms* sequences. As a result, in order to solve the disadvantage of Apriori-like algorithms [1], a variety of algorithms such as PrefixSpan [17], SPADE [25], SLPMiner [22] and GST [12] have been proposed. In order to improve the efficiency, each algorithm pursues a different method of discovering frequent sequential patterns, which makes them featured by the capability of mining such patterns without even generating any candidates.

With respect to the representation of the content of documents, some research works have used phrases rather than individual words [21]. However, the effectiveness of the text mining systems was not improved very much. The likely reason is that, a phrase-based method has "lower consistency of assignment and lower document frequency for terms" [21]. Hence, in this paper, we present a novel concept for mining text documents for sequential patterns. Instead of using single words, we use pattern-based taxonomy to represent documents. By pruning meaningless patterns, which have been proven to be the source of the 'noise' in this study, the problem of 'overfitting' is solved and the experimental results which show the encouraging outcomes are achieved.

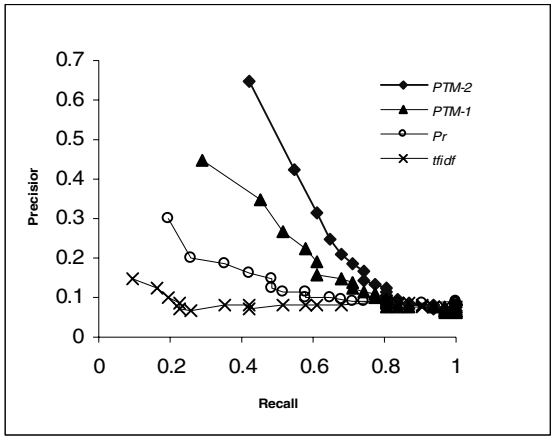

Figure 3. Precision/Recall curves on Topic110

Some future research works can be conducted based on the results of this research. To improve the accuracy, one major direction is to extract and use the interesting information from negative or unlabeled documents. The weighting scheme of discovered patterns can then be optimized.

References

[1] R. Agrawal, and R. Srikant, "Mining sequential patterns," *Proceedings of Int. Conf. on Data engineering (ICDE'95),* Taipei, Taiwan, 1995, pp. 3-14.

[2] H. Ahonen, O. Heinonen, M. Klemettinen, and A. I. Verkamo, "Applying data mining techniques for descriptive phrase extraction in digital document collections," *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL'98)*, Santa Barbara, CA, 1998, pp. 2-11.

[3] M. L. Antonie and O. R. Zaiane, "Text document categorization by term association," *ICDM02*, Maebashi City, Japan, 2002, pp. 19-26.

[4] Y. Bi, T. Anderson, S. McClean, "A rough set model with ontologies for discovering maximal association rules in document collection," *Knowledge-Based Systems vol.16*, 2003, pp. 243-251.

[5] G. Chang, M.J. Healey, J. A. M. McHugh, and J. T. L. Wang, *Mining the World Wide Web: an information search approach*, Kluwer Academic Publishers, 2001, pp. 192.

[6] D. A. Evans, et al., CLARIT experiments in batch filtering: term selection and threshold optimization in IR and SVM Filters, TREC02, 2002.

[7] R. Feldman, et al., "Text mining at the term level," *Lecture Notes in AI 1510: Principle of data mining and knowledge discovery*, Springer-Verlag, 1998, pp. 65-73.

[8] R. Feldman and H. Hirsh, "Mining associations in text in presence of background knowledge," *KDD96*, 1996, pp. 343-346.

[9] D. A. Grossman and O. Frieder, *Information retrieval algorithms and heuristics*, Kluwer Academic publishers, Boston, 1998.

[10] K. M. Hammouda and M. S. Kamel, "Phrase-based document similarity based on an index graph model," *Proceedings of IEEE 2002 Int. Conf. on Data Mining (ICDM'02),* Dec. 2002, pp. 203-210.

[11] J. D. Holt and S. M. Chung, "Multipass algorithms for mining association rules in text databases," *Knowledge and Information Systems vol.3*, 2001, pp. 168-183.

[12] Y. Huang and S. Lin, "Mining sequential patterns using graph search techniques," *Proceedings of the 27th Annual Int. Computer Software and Applications Conf. (COMPSAC'03)*, Dallas, 2003, pp. 4-9.

[13] Y. Li, C. Zhang, and J. R. Swan, "An information filtering model on the Web and its application in JobAgent," *Knowledge-based Systems 13(5)*, 2000, pp. 285-296.

[14] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," *ICDM03*, 2003, pp. 179-186.

[15] B. Liu, Y. Ma, and Philip S. Yu, Discovering business intelligence information by comparing company Web sites, in: N. Zhong, J. Liu and Y. Y. Yao (eds.), "*Web Intelligence*", Springer-Verlag, 2003, pp. 105-127.

[16] J. Mostafa, W. Lam, and M. Palakal, "A multilevel approach to intelligent information filtering: model, system, and evaluation," *ACM Transactions on Information Systems 15(4*), 1997, pp. 368-399.

[17] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," *Proceedings of Int. Conf. on Data Engineering (ICDE'02),* Heidelberg, Germany, 2001, pp. 215-224.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program* 14(3), 1980, pp. 130-137.

[19] S. Robertson and I. Soboroff, The TREC 2002 Filtering Track Report, *NIST Special Publication: SP 500-251 (TREC-2002)*, 2002.

[20] T. Rose, M. Stevenson, and M. Whitehead, "The Reuters Corpus Volume1 - From yesterday's news to today's language resources," *Proceedings of the 3rd Inter. Conf. on Language Resources and Evaluation*, Las Palmas, Spain, 2002.

[21] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys, Vol. 34, No.1,* 2002, pp. 1-47.

[22] M. Seno and G. Karypis, "SLPMiner: An algorithm for finding frequent sequential patterns using length-decreasing support constraint," *Proceedings of IEEE 2002 Int. Conf. on Data Mining (ICDM'02),* 2002, pp. 418-425.

[23] P. Tzvetkov, X. Yan, and J. Han, "TSP: Mining top-K closed sequential patterns," *ICDM03*, 2003, pp. 347-354.

[24] X. Yan, J. Han, and R. Afshar, "CloSpan: mining closed sequential patterns in large datasets," *Proceedings of SIAM Int. Conf. on Data Mining (SDM 03)*, San Francisco, CA, 2003, pp. 166-177.

[25] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning vol. 40*, 2001, pp. 31-60.