Finding Short Patterns to Classify Text Documents

Jiyuan An and Yi-Ping Phoebe Chen

School of Information Technology, Faculty of Science and Technology,
Deakin University, Melbourne, VIC 3125, Australia
{jiyuan, phoebe}@deakin.edu.au

Abstract

Many classification methods have been proposed to find patterns in text documents. However, according to Occam's razor principle, "the explanation of any phenomenon should make as few assumptions as possible", short patterns usually have more explainable and meaningful for classifying text documents. In this paper, we propose a depth-first pattern generation algorithm, which can find out short patterns from text document more effectively, comparing with breadth-first algorithm.

Availability:

The translated vector data of the tested web text documents and the found patterns can be obtained from http://www.deakin.edu.au/~jiyuan/wi2006.html

Keywords

Document Categorization, rule generation, breadth-first, depth-first.

1. Introduction

To categorize text documents, the classification methods [3][7][8][9][10] are usually be used to generate rules (or patterns) for each category. A rule is combination of keywords. For example, "kw1=computer ∩ kw2=information" may be a pattern for computer document category. Such as Rocchio method [4], Naïve bayes based method [6], and SVM based text classification method are widely used. These methods learn labeled text documents and then construct a classifier. By using the classifier, a new coming text document can be predicted in terms of which category it belongs to. The keywords which appear in documents can be considered as the features of text document domains. If the keywords form a feature space; every document becomes a data point in the feature space. However, the dimensionality for text document domain is very high because every keyword corresponds to a dimension. This results in the ineffective performance of classification methods [2].

In [2], a robust rule generation method was proposed. This method enumerates all possible keyword combinations. The enumeration starts from 1-keyword combination to n-keyword combination. Each keyword combination can be viewed as a subspace in feature space. The subspace covers certain text documents. In an ideal situation, we can find a keyword combination that covers only specific category documents. In most cases, we use several keyword combinations, *C*'s, to cover a specific category document set *S*.

$$C_1 \cup C_2 \cup \cdots \Rightarrow S \qquad (1)$$

If the keyword combination consists of *k* keywords, it is called *k*-keyword combination. For example, Equation 2 is a 2-keyword combination. It covers the documents in which both keyword No. 3 and No. 7 appear. The Equation 2 can be shortened to (kw3 ∩kw7).

$$(kw_3='Y') \cap (kw_7='Y') \qquad (2)$$

If a document is covered by a k-keyword combination, it will also be covered by its sub combination (*i*-keyword combination ($i < k$)). Based on this fact, most of the irrelative keyword combinations can be pruned at the early stages of the rule generation procedure [2]. Firstly, all possible 1-keyword combinations are enumerated. The 1-keyword combinations are sorted into descending order for the "primary key" and in ascending order for the "second key". The primary key is the number of specific category documents that are covered by keyword combination; the second key is the number of other category documents that are covered by the keyword combinations. Secondly, the sorted 1-keyword combinations are put in a queen called **Bin1**. Thirdly, from the top of Bin1, two 1-keyword combinations are combined to form 2-keyword combinations. Like the first step, all 2-keyword combinations are sorted by primary and second keys to form Bin2. When a keyword combination that covers only the specific category documents is found, it becomes a pattern for the specific category. To reach this kind of cover quickly, In this paper, we propose an algorithm that is based on depth-first traversal.

Unlike the breadth-first traversal algorithm [2], the new algorithm finds the most potential keywords efficiently.

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

The irrelative keyword combinations can be pruned at an earlier stage. This avoids dealing with the huge number of keyword combinations whose number increases exponentially in terms of the number of keywords.

2. Preliminaries

Given a specific category, our algorithm induces a set of decision *rules*. Each rule is of the form "if <cover> then predict <category>", where <cover> is Boolean keyword combinations as given below:

A *selector* is the basic test for a keyword. For example, *kw = yes (no)* denotes keyword *kw* (dis)appears in all documents of a category. The selector can be shortened to *kw*, because we do not consider the keyword combinations that disappear in specific documents as a selector.

A conjunction of selectors is called a *cover (or rule)*. We say that a rule ***covers*** a document if the rule is true. For example, *Figure 1* indicates whether or not three keywords kw1, kw2 and kw3 appear in three different documents d1, d2 and d3. If we have a rule: (kw1 ∩ kw3), we say that the rule covers two documents d1 and d3.

	kw1	kw2	kw3
D1	'Y'	'N'	'Y'
D2	'Y'	'N'	'N'
D3	'Y'	'Y'	'Y'

Figure 1 Documents and keywords. 'y'/'n' denotes whether the corresponding keyword (dis)appears in the documents.

The rules produced by our algorithms can be viewed as finding optimal subspaces covering only one category's documents and not any other documents are included. Throughout the paper, we call this kind of subspace a *positive cover*. The training examples consist of positive samples and negative samples. In our approach, the description of categories is found one by one. If we have *n* categories, we have to learn *n* times to find each description of the category. In each trial, all documents in the specific category are called *positive documents* whereas other documents are called *negative documents*.

3. Our approach

As a pattern corresponds to a cover, we can find out all patterns for a specific category determining coverage of the category documents. A naïve algorithm is immediate.

To cover all text documents for a specific category, more than one positive cover may be needed. *Figure 2* shows a naïve algorithm that generates the patterns to describe a specific category. In the while-loop (line 3-14), we continue finding one by one until all positive samples have been covered. In the for-loop (line 6-10), a keyword combination that has the biggest coverage for positive samples is selected as a pattern. In line 13, all positive samples that are covered by the found pattern are removed. The next pattern is found in the remaining positive and negative samples. As mentioned before, it is not practical to enumerate all possible keyword combinations, because of the high dimensionality of a document's feature space.

```
1. Algorithm ruleGeneration()
2. rules = empty
3. while positiveSample exist
4.     maxPositiveCover = 0
5.     bestCover = null
6.     for each keyword combination Ci
7.         If cover(Ci) > maxPositiveCover
8.             bestCover = Ci
9.             maxPositiveCover = cover(Ci)
10.        end if
11.    endfor
12.    rules = rules ∪ bestCover
13.    delete positiveSample covered by Ci
14. end while
15. return rules
```

Figure 2 A naïve algorithm for rule generation

Similar to most association rule algorithms [1][2], pruning irrelative rule candidates (or keyword combinations) is a crucial problem. In this paper, we introduce a depth-first traversal method to prune irrelative keyword combinations. *Figure 3* illustrates how to filter out irrelative keywords in the process of finding patterns. Firstly, all 1-keyword combinations are listed. The number of positive and negative documents is counted. The 1-keyword combinations are sorted in ascending order of the number of positive documents and in descending order of the number of negative documents, as shown in subfigure (B). Secondly, kw2 and kw3 are selected to be combined because they have biggest positive covers; they are the most potential keywords to cover more positive documents. From subufigure (c), the combination of kw2∩kw3 covers 2 positive documents and 0 negative documents. Consequently, the combination kw2∩kw3 is a pattern for the dataset. Thirdly, the documents D1 and D2 are removed because they are covered by rule kw2∩kw3. Only one pattern is found because no positive documents remain in the dataset.

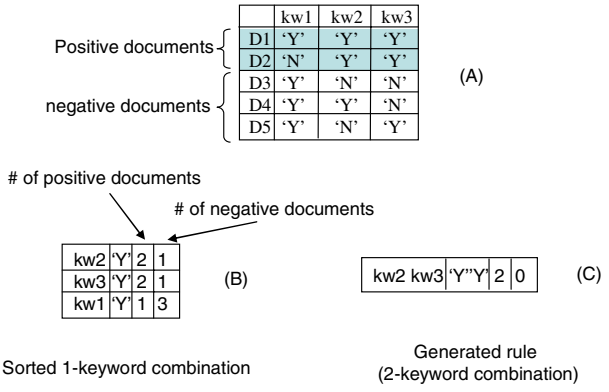

Figure 3 An example for depth first rule generation algorithm

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

The kw1 in 1-keyword combination does not need to combine with other keywords (kw2 and kw3), because after the combining of kw2 and kw3, the positive coverage of kw1 (=1) is smaller than that of kw2∩kw3. In a real dataset, there is a huge number of this kind of keyword combinations that can be pruned away. *Figure 4* shows how to construct bins for keyword combinations.

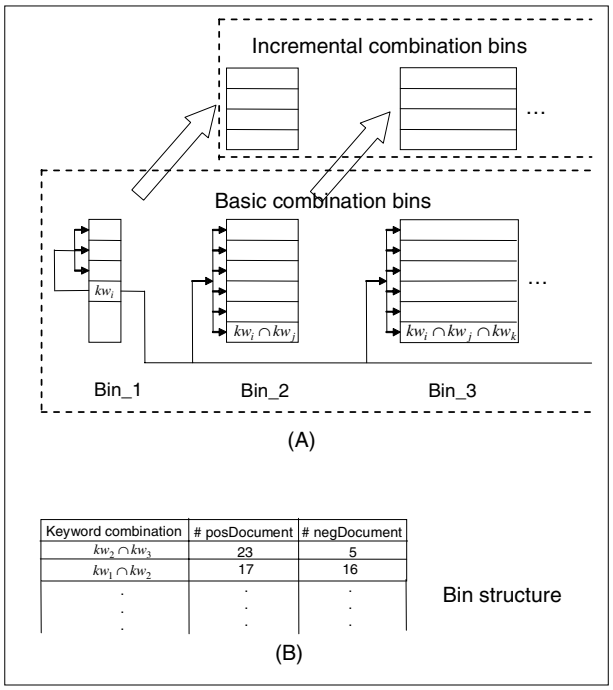

Figure 4 The flow of finding a rule for a specific category

The keyword combinations are stored into Bins as shown in *Figure 4*(A). They are sorted in descending order of the number of positive documents that are covered by the keyword combination and in ascending order of the number of negative documents covered by the keyword combination. If a combination has zero negative document, we keep the combination as the best answer. The best answer is updated in the expanding process. It becomes a pattern at the end of process. They can be viewed as primary and secondary keys. The "good" keyword combinations are always in the top of the Bins because they cover more positive documents and less negative documents. *Figure 4*(B) shows an example of Bin_2. The first column represents the keyword combinations. The second and third columns represent the number of positive and negative samples included in the keyword combination. Here we explain our strategy to expand *n*-key combinations to *n+1*-key combinations. We give an example of expanding from 1-key combinations to 2-key combinations.

Figure 5 illustrates An example of expanding keyword combinations. It has 5th largest keyword "kw8" in Bin_1. We assume the 2-keyword combinations from four largest keyword "kw2", "kw5", kw1" and "kw3" have been calculated and already in basic combination bin_2. There are three rational 2-keyword combinations, i.e. kw2∩kw5, kw2∩kw3 and kw1∩kw5 cover more positive documents than that is covered by the best keyword combination.. Now we want to combine "kw8" with its upper keywords (kw2, kw5, kw1 and kw3). Their combinations are put in incremental combination Bin_2. There are two rational elements "kw2 kw8" and "kw3 kw8" (Note that "kw5∩kw8" and "kw1∩kw8" are assumed to be pruned away). When we expand every element in the basic combination Bin_2 by adding "kw8", we check whether each subset including "kw8" is in incremental combination Bin_2. For example, the first element in the basic combination bin, Bin_2 "kw2∩kw5" is expanded to "kw2∩kw5∩kw8". We then check its 2-keyword subset "kw2∩kw8" and "kw5∩kw8". Due to the fact that "kw5∩kw8" is not in incremental combination Bin_2, the expanded keyword combination "kw2∩kw5∩kw8" can be pruned away. Only "kw2∩kw3∩kw8" cannot be pruned away by checking incremental combination bins because "kw2∩kw8" and "kw3∩kw8" are in the incremental combination bins. Finally we check whether "kw2∩kw5∩kw8"is a rational combination by calculating its covering number of positive documents.

As the incremental combination bins are introduced, a large number of combinations can be pruned away without check their positive document coverage. This is an improvement of the method proposed by [2].

We Summarize depth-first algorithm below: (1).we calculate the number of positive and negative documents for each 1-keyword combination and then proceed to put all 1-keyword combinations in the first Bin (Bin_1), as sorted by the primary and secondary keys. (2) we expand the keyword combinations for bin_1 one by one as shown in *Figure 4*.

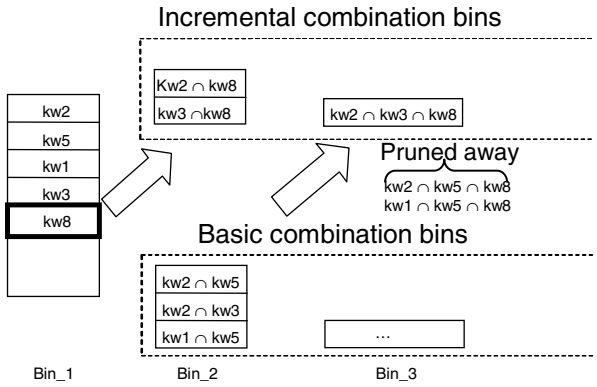

Figure 5 An example of expanding keyword combinations

4. Experiment

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

We test our algorithm to find rules for describing a given category on web documents. In paper [2], the effectiveness in accuracy and interpretability has been shown for the enumeration based algorithm by comparing to ID3 and CN2. The rules generated by this paper are the same as by paper [2]. We demonstrate the priority of our new algorithm in both time and space complexity by comparing with breadth-first rule generation algorithm.

4.1 Dataset

The dataset consists of 314 web text documents that are collected from various University of Waterloo web sites. It was downloaded from http://pami.uwaterloo.ca/~hammouda/webdata/ [5]. The cleaned up data can be found in *http://www.deakin.edu.au/~jiyuan/wi2006.html*. Ten categories and the number of documents in the categories are listed below:

Black bear attach (30)
Campus network (33)
Canada transportation roads (22)
Career services (52)
Co-operative education (55)
Health services (23)
River fishing (23)
River rafting (29)
Snowboarding skiing (24)
Winter Canada (23)

All stop words, such as "a", "an", "on" are removed, and we have changed all words into their root, for example "fishing" → "fish". If a word rarely appears in the documents, we treat it as a noise in the classification of the text document. So we removed all low frequent words. In the experiment, we have removed words that appear in documents below 40 times. Finally, 619 keywords are obtained as the features to represent documents.

Compared to the Breath First rule generation algorithm, the depth first algorithm can find the largest positive coverage faster. *Figure 6* shows the CPU cost for generating rules to describe documents in 10 categories. The horizontal axis represents 10 categories. The vertical axis represents the seconds to generate rules for corresponding categories. Note that the log scale is used in the vertical axis. In the figure, the time complexity has been reduced by adopting the depth-first technique. Since the page limitation, we omit other experiment results.

5. Conclusion

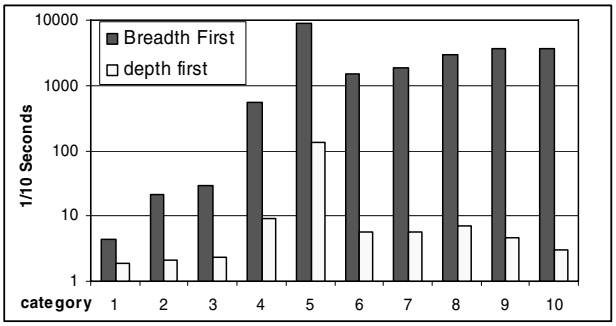

Figure 6 Comparison of CPU time

Document classification is a fundamental topic in the web world. Many learning algorithms such as AQ15, ID4 and CN2 have been applied to classify categories of documents. In this paper, we proposed a depth-first rule generation algorithm. Comparing to breadth-first algorithm, the new algorithm demonstrates its priority in time and space complexities, especially in finding short patterns. Since the algorithm needs only limited space, the algorithm can be executed on normal personal computers. Besides, the rules produced by our algorithm are more accurate and interpretable.

6. Acknowledgments

The work reported in this paper was partially supported by the Australian Research Council's Discovery Project grants DP0344488 and DP0559251.

Reference

1. Agrawal, R Srikant, R.: "Fast Algorithms for Mining Association Rules", VLDB94, 487-499. 1994.
2. An, J. and Chen, Y.: Concept Learning of Text Documents. Web Intelligence 2004: 698-701
3. Clark, P and Niblett, T: The CN2 Induction Algorithm. Machine Learning 3: 261-283 (1989)
4. Joachims, T.: A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. ICML97. 143-151 (1997).
5. Hammouda, K. M., Kamel, M. S.: Phrase-based Document Similarity Based on an Index Graph Model. ICDM 2002: 203-210
6. Lewis, D. D.: Naïve (Bayes) at forty: The independence assumption in information retrieval. ICML 1998, 148-156
7. Li, Y. and Zhong, N., Interpretations of association rules by granular computing, ICDM 2003: 593-596
8. Li, Y and Zhong, N. Ontology-based Web mining model: representations of user profiles, WI 2003: 96-103.
9. Michalski. R. S. Carbonell, J. G. and Mitchell, T. M.: "Machine learning an artificial intelligence approach", Morgan Kaufmann Publishers, INC., 1983
10. Mitchell, T. "Machine learning", McGraw Hill, 1997