Concept Learning of Text Documents

Jiyuan An[1], Yi-Ping Phoebe Chen[1, 2]

[1] School of Information Technology, Faculty of Science and Technology,
Deakin University, Melbourne, VIC 3125 Australia
[2]Australian Research Council Centre in Bioinformatics
{jiyuan,phoebe}@deakin.edu.au

Abstract

Concept learning of text documents can be viewed as the problem of acquiring the definition of a general category of documents. To definite the category of a text document, the Conjunctive of keywords is usually be used. These keywords should be fewer and comprehensible. A naïve method is enumerating all combinations of keywords to extract suitable ones. However, because of the enormous number of keyword combinations, it is impossible to extract the most relevant keywords to describe the categories of documents by enumerating all possible combinations of keywords. Many heuristic methods are proposed, such as GA-base, immune based algorithm. In this work, we introduce pruning power technique and propose a robust enumeration-based concept learning algorithm. Experimental results show that the rules produce by our approach has more comprehensible and simplicity than by other methods.

1. Introduction

Classification of documents is one of the main problems in web mining [5] [6] [7]. Most methods proposed are based on distance of Vector Space Model [3] [13]. Because of the potential high dimensionality of the vector space, distance-based cluster algorithms can not be applied effectively in text documents. It is called "curse of dimensionality" [1]. To enhance selective of documents, many researchers proposed different kinds of distance, such as *cosine* measure, *Jaccard* measure and phase-based distance [2].

This paper proposes a concept learning algorithm to describe categories of text documents. The produced rules can be used to classify different document categories. The different categories are represented by different rules. To produce the rules, a naïve method can be employed, enumerating all possible keyword combinations and then examining whether they appear all documents of the category or not. However, because of the enormous number of keyword combinations, it is impossible to extract the most relevant keywords to describe the categories of documents by using the naïve method. Many heuristic methods are proposed, such as GA-base [4], immune based algorithm. In this work, we introduce pruning power, and propose an enumeration-based algorithm.

2. Our approach

The documents under the same category have some common features. They usually include some common keywords. For example, the documents in computer category may have "IBM", "Linux" and etc., while documents about sports may have "World Cup", "Olympic games" and etc. keywords. We classify documents using the information of which kinds of keywords appeared in them. The keyword combinations generate rules to describe concepts of text documents.

To describe the concepts of categories of text documents, many different rule sets can be found from very complex rule sets to very simple rule sets. However, since the simplicity is a very important criterion [9], we must use a small rule set to describe the concepts. Moreover, each rule should be short. Another important factor to evaluate the rule sets is explanation, that is, the rules should be interpretable.

According to terminology in concept learning, we call a given category's documents *positive documents*, and the others are called *negative documents*. There are usually more than two categories, so we assume the target category is positive documents, and the other category's documents are negative documents. Our goal is to find the rules which cover largest positive documents but no negative documents are included, so the rules can be viewed as *positive covers.*

In this paper, we enumerate all possible keyword combinations, and then calculate the numbers of *positive documents* and *negative documents* covered by every keyword combination, and then see which keyword combination has the biggest cover for positive documents. This process is repeated until every positive document is covered by at least one rule. We employ the technique used in AQ algorithm [10]: When the first rule is found, we delete all documents covered by this rule. On the

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

remaining positive and negative document set, the algorithm is repeated. The 3rd, 4th,…,rule can be found in the same way. At last, on the remaining documents, there are no positive documents left. By doing so, we guarantee produced rules cover all positive documents and exclude every negative documents.

2.1 A naïve algorithm

To describe a category of text document, more than one rule usually are needed. As shown in Figure 1, the algorithm continues until all the positive documents are covered by one of produced rules. In line 2-4, from all possible keyword combinations, a rule *R* which covers the biggest positive document is found, while no negative documents are covered by *R*. To find the second rule, all documents covered by *R* are deleted (line 5), and then program repeats on the remaining document set. Noted that all deleted documents are positive documents. By using the naïve algorithm, each produced rule covers more positive documents than other keyword combinations. Since finding the set of biggest covers are NP-hard problem, we can not guarantee the set of rules produced by naïve problem is the best one, but the naïve algorithm is principally mountain-climbing method; each rule is the "best" one at each step.

Naïve algorithm
1.While the number of positive documents <> 0
2. Foreach keyword combination
3. Find a rule ***R*** Which is the biggest positive cover.
4. End foreach
5. Delete all documents covered by ***R***
6.End while

Figure 1 A naïve Algorithm to find rules

Although the naïve algorithm can find effective rules to describe the concepts of text documents, it can not find rules effectively. It is because that enumerating all possible keyword combinations costs a lot of CPU time. When the number of keywords becomes large, it is impossible to use naïve algorithm to produce rules. For English text documents, even we delete stop words and rarely appeared words. The number of keywords is usually much more than 100. We show the number of number of keyword combinations with the simplest case, i.e. every keyword has 2 discrete values (0 and 1). For *n* keywords, there are $C_n^1 \times C_2^1$ 1-keyword combinations, and $C_n^2 \times C_2^1 \times C_2^1$ 2-keyword combinations. If we have only two keywords ($n = 2$), 1-keyword combinations are 4: k_0="1", k_0="0" k_1="1", k_1 = "0". 2-keyword combinations are: (k_0, k_1) = {("1", "1"), ("1", "0), ("0", "1"), ("0", "0")}. The algorithm chooses the biggest positive covers from 1-keyword (c_1), 2-keyword (c_2), …, *n*-keyword(c_n) combinations. The number of all possible combinations of keywords is up to $\sum_{i=1}^{n} C_n^i \times (C_2^1)^i$. Consequently, the naïve algorithm is not practicable to examine the enormous number of keyword combinations whether they appear in documents or not. On the next section, we introduce pruning power technique to improve the naïve algorithm, and get the same results as from naïve algorithm.

2.2 Pruning power

Firstly, let's observe a relationship between rule and its cover. When we lengthen a rule, its cover definitely becomes smaller. For example, a rule k_1 = "1"covers *t* documents. The rule k_1="1" & k_2="0" covers only a part of *t* documents. That is, the number of documents cover by second rule is smaller than the original rule.

$$Cover\ (k_1 = "1") \supseteq Cover\ (k_1 = "1" \cap k_2 = "0").$$

Based on this observation, the naïve algorithm can be improved. To express our idea concisely, here we explain some notations. The symbol *r* denotes a rule to describe a category of documents. We can use a formula to repress it: $r = (k_1 = "1" \cap k_2 = "1" \cap \cdots)$. k ="1" means that keyword k appears in a document. k = "0" means a document does not include keyword k. A rule (k_1="1" & k_2="0") covers the documents which has keyword k_1 and no keyword k_2 appeared. The number of positive documents covered by the rule *r* is denoted by length(*r*). According to the observation, for rule r_1 and r_2, we have:

$$length(r_1) \geq length(r_2) \Rightarrow length(r_1) \geq length(r_2 \cap k)$$

where *k* is an arbitrary keyword. …..(1)

Improved algorithm
Input:
n: number of keywords
Output:
ans : rule set

1.While the number of positive documents <> 0
2. For nc = 1:n {the number of keywords in a combination}
3. Foreach nc-keyword combination
4. Calculate the NumPos and NumNeg
5. {NumPos, NumNeg: number of positive and negative
6. documents covered by nc-keyword combination.}
7. **if** (NumNeg = 0) & (NumPos -> maximun)
8. ans <- {ans, nc-keyword}
9. **Endif**
10. End foreach
11. End foreach
12. Delete all documents covered by ***R***
13.End while

Figure 2 The improved algorithm

Based on Equation 1, the line2-4 in **Figure 1** can be improved. Enumerating keyword combination start from 1-keyword, 2-keyword,…. If we find the biggest cover, then append the keyword combination to answer set and stop enumerating. The program is shown in **Figure 2**. From our experiment, program stops at early stage. Most keyword combinations are pruned.

2.3 Reduction of keyword combinations

The number of keywords is a very big number, but most documents include just a very little subset of the keywords. For example, the keyword "linux" usually does not appear in mathematics category. We can present the relationship between documents and keywords with a very big matrix. Its columns are keywords, its rows are documents. In the matrix, most of element values are 0. Even introducing the pruning power explained in previous Section, the number of keyword combinations is still very large. It is impossible to apply enumeration-base algorithm to classification of text documents. Fortunately, to describe the concept of document category, we focus on the keywords appeared in the documents. Usually we do not care what kinds of keywords do not appear in the category's documents. So we can enumerate the keyword combinations (k = '1'), and omit the keyword combinations (k = '0'). Consequently, the number of keyword combinations becomes $\sum_{k=1}^{n} C_n^k$, where n is the number of keywords.

2.4 The algorithm

A positive cover is represented by a rule consisting of 1, 2,…, n keywords. Using the pruning power described in previous Section, we propose a method to find biggest positive cover. The method makes it possible to find rules to describe a category with enumeration based algorithm. In our algorithm, the positive covers (or rules) are found step by step until every positive document is covered by at least one rule. Like algorithm AQ15 [11], when a rule found, all positive documents covered by the rule are removed. The process illustrated in Figure 2 is repeated. That is, from the remaining positive documents and negative documents, the next biggest positive cover can be found. In the same way, all the positive documents covered by the rule are removed step by step. The process will continue until all positive documents are removed. Finally, the remaining ones are all negative documents, all the positive documents removed. The concept learning process stops and every positive document is covered by at least one rule; and no negative documents are covered by the found rules. To describe another category of text documents, we just change the positive documents set. The documents under the new category become positive documents. The documents under other categories are set negative documents.

3. Experiment

We tested enumeration-based algorithm on ten categories of web documents. We use keywords to describe the concepts of different categories. In our experiments, firstly, we show the pruning power of keyword combinations. Since our program is based on memory, if there are too many keyword combinations need to be calculated, our approach loses its practicable. Secondly, we compare the results with other methods. We select one of the most effective methods decision tree as a reference. As shown in [9], the rules produced must satisfy completeness, consistency and simplicity. The first two conditions are the essential. The simplicity means that the rules should be short and reflecting the characters of the categories.

3.1 Dataset

Our experimental data set has 314 web documents collected from University of Waterloo various web sites. It was downloaded from http://pami.uwaterloo.ca/~hammouda/webdata/ [2]. Ten categories and the number of documents in the categories are list below:

1. Black bear attach (30)
2. Campus network (33)
3. Canada transportation roads (22)
4. Career services (52)
5. Co-operative education (55)
6. Health services (23)
7. River fishing (23)
8. River rafting (29)
9. Snowboarding skiing (24)
10. Winter Canada (23)

We deleted the stop words, such as "a", "an", "on", and change all words into their root, for example "fishing" → "fish". If a word appears in documents rarely, we think it is a noise in classification of text document. So we delete low frequent words. In the experiment, we delete words that appear in documents below 40 times. Finally, we got 619 keywords as features to represent documents.

3.2 Pruning power

To find a biggest positive cover, we have to create 1 to n-keyword queues. We call them bins. The size of k-keyword bins is an important criterion of our algorithm. If the size is too big, it will takes long CPU time to select the large positive cover, and can not run program based on memory. By introducing pruning power, the size of bin for constitute first rules is small. In our experiment, 5/9

categories stop before 3-keyword combinations. The other categories stop before 6-keyword combinations. The number of candidates kept in bins is very small (not over 600). The algorithm can be run based on memory in a usual computer.

3.3 Comparison with decision tree C4.5

The rules produced by decision tree C4.5 conclude NOT literal. That make rules lack of interpretability. For example, the first category "Black bear attach" is represented as "~art"& ~snowfall & bear & ~package &~raft" ("~" denotes "NOT"). Form this description, the category is explained: there is no relation with "art", "snowfall", "package" and "raft", but "bear". We believe that no one can understand the concept description well. The experiment was implemented base on the source downloaded from [14].

Our concept leaning algorithm extracts keywords that exist in the documents. So in the rules produced by our algorithm, there is no "NOT". From the rules shown in Table 1, we can outline easily the characters of the category of documents. For example, the category 0 can be described with two words "bear" and "attack".

The category 8 and 9 (snowboarding skiing and winter Canada) are two similar categories. Snowboarding skiing is occurred in winter. In winter Canada, many people play snowboarding skinning. From the rules produced by our concept learning algorithm, the word "snowboard" appears in every documents of category 8, while the word "snowfall" exists in every documents of category 9. "inform", "function" or "time" intersect with "snowboard" can describe category 8.

Table 1 The rules produced by enumeration based algorithm

	Category 0	Category 2	Category 8	Category 9
Rule 1	Bear & attach	transport & road & program	Inform & snowboard	Snowfall & rain
Rule 2		Transport & traffic	Function & snowboard	Snowfall & rang & amp
rule3			time & snowboard	Snowfall & effect

4. Conclusions

In this paper, we proposed a robust concept learning algorithm to describe the category documents. The algorithm is based on the enumeration of keywords. To reduce the vast number of keyword combinations, we introduced pruning power to create *k*-keyword bins. Since the bin size is very small, our algorithm becomes practicable in a usual personal computer. From our experiment, the rules produced by our algorithm have more interpretable and concise than by other methods.

5. Acknowledgments

The work in this paper was supported by Grant DP0344488 from the Australian Research Council.

Reference

1. Beyer, K. S., Goldstein, k. Ramakrishnan, R. Shaft, U.: When Is "Nearest Neighbor" Meaningful? ICDT 1999: 217-235
2. Hammouda, K. M., Kamel, M. S.: Phrase-based Document Similarity Based on an Index Graph Model. ICDM 2002: 203-210
3. Jain, A. K., Dubes, R. C.: Algorithms for Clustering Data. Prentice-Hall 1988
4. Jong, K., Spears, W., Diana F. Gordon: Using Genetic Algorithms for Concept Learning. Machine Learning 13: 161-188 (1993)
5. Li, Y. and Zhong, N., Interpretations of association rules by granular computing, ICDM 2003: 593-596
6. Li, Y and Zhong, N. Ontology-based Web mining model: representations of user profiles, WI 2003: 96-103.
7. Li, Y. Zhang, C. and Zhang, S. Cooperative strategy for Web data mining and cleaning, Applied Artificial Intelligence, an International Journal, 2003, Vol 17 No 5-6, pp. 443-460.
8. Michalski, R. S.: "On the quasi-minimal solution of the general covering problem," in Proceedings of the Fifth International Symposium on Information Processing, vol. A3, 1969, pp. 125–128
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
11. Michalski, R. S., Mozetic, I., Hong, J, Lavrac, N., "The AQ15 inductive learning system: An overview and experiments," Technical Report UIUCDCS-R-86-1260, University of Illinois, Urbana-Champaign. IL, 1986
12. Quinlan, J. R.: Induction of Decision Trees. Machine Learning 1(1): 81-106 (1986)
13. Salton, G., Wong, A., Yang, C. S.: A Vector Space Model for Automatic Indexing. Commun. ACM 18(11): 613-620 (1975).
14. Winston, P. "C4.5 Decision Tree", http://www2.cs.uregina.ca/~hamilton/ 2002