Keyword Extraction for Text Categorization

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

Text categorization (TC) is one of the main applications of machine learning. Many methods have been proposed, such as Rocchio method, Naïve bayes based method, and SVM based text classification method. These methods learn labeled text documents and then construct a classifier. A new coming text document's category can be predicted. However, these methods do not give the description of each category. In the machine learning field, there are many concept learning algorithms, such as, ID3 and CN2. This paper proposes a more robust algorithm to induce concepts from training examples, which is based on enumeration of all possible keywords combinations. Experimental results show that the rules produced by our approach have more precision and simplicity than that of other methods.

Keywords

Document Categorization, Keywords Extraction, Concept learning.

1. Introduction

In the last decade, text categorization (TC) attracted many machine researchers [1][6][7]. Machine learning methods such as Naïve bayes based method [4][9], and SVM based text classification method have been done to classify text documents. These methods learn labeled text documents and then construct a classifier. A new coming text document can be estimated to find category it belongs to. However, the classifier built by these methods does not give any explanations for each category. So end users know that a new coming text document is distributed to a category, but they do not know why. They do not know the types of keywords that played roles in classification. In our method, several keyword combinations are used to describe every category. For example, a category that includes IT documents can be described as "Information and Computer or information and technology". That is, if a text document includes keywords "information" and "Computer", this document should belong to IT category. A text document including "information and technology" belongs to IT category as well. End users can use keyword combinations as a concept to describe each category. There are many concept learning algorithms proposed. ID3 [10] and CN2 [2] are classical algorithms for concept learning. In this paper, we propose a robust algorithm that induces concepts from training examples, which is based on enumeration of all possible keyword combinations. However, the number of keywords are usually very large, the number of their combination increases exponentially in the number of keywords. To reduce the number, we introduce pruning power and propose a robust enumeration-based concept learning algorithm.

Concept learning of text document can be viewed as the problem of acquiring the definition of a general category from given labeled text documents. In general, documents consist of vast number of keywords. If each keyword is represented as an axis in a dimensional space, a text document corresponds to a data point in the dimensional space. The documents of a category are represented in a subspace of the dimensionality. Concept learning tries to find the subspaces and describes them as rules.

English text documents usually consist of more than several thousand words. Except some stop words such as "a", "the" and etc., most words can be keywords in TC. The built keyword-vector space becomes very sparse. But most documents have only a very small keywords subset included. It is because one document usually is written in one topic. Different topic documents use different set of words. For example, chemical words usually do not appear in IT documents in high frequency. In our experiments, after preprocessing, we selected about 600 keywords, but the average number of keywords appeared in one document is only about 30.

To constitute subspace of categories, it is the naïve method that enumerates all possible keyword combinations. One keyword combination represents one subspace. By checking every subspace whether all documents of a category are covered or not, we can find the "best" subspace to describe the category. If the number of keywords is n, the number of keywords in

0-7803-9035-0/05/$20.00 ©2005 IEEE

combination can be from 1-keyword to *n*-keywords. For example, in a 2-keywords combination ("computer = Yes" and "sulfur = No"). It means that the word "computer" appears in the document, but "sulfur" does not appear in the document. We use this kind of simple Boolean combination of keyword tests to describe categories. Each combination of keywords is checked by all documents for its appearance in the documents or not. Then, we select the keyword combination that covers most number of documents in a category, but no any other category's documents are covered. For description of a category, more than one combination may be needed. Since the number of possible keyword combination is a very big number, it is impossible to check each combination with each document. In this paper, we introduce pruning power to reduce the number of possible combinations. In our experiment, the combinations of keywords can be reduced to a reasonable number. The algorithm can be run even on personal computer that has limited memory space.

2. Concept Description and Interpretation

Similar to other concept learning methods, our concept learning method induces a set of decision *rules*, one for each category. Each rule is of the form "if <cover> then predict <category>", where <cover> is a Boolean combination of keyword tests as given below:

A *selector* is the basic test on a keyword. For an instance, *kw = yes (no)* denotes keyword the *kw* (dis)appears in all documents of a category. A conjunction of selectors is called a *cover (or rule)*. We say that a rule ***covers*** an example if the rule is true for the document. **Figure 1** indicates whether three keywords kw1, kw2 and kw3 appear in three documents d1, d2 and d3. If we have a rule: (kw1='Y' ∩ kw2='N'), we say that the rule covers two documents d1 and d3.

	kw_1	kw_2	kw_3
d_1	'Y'	'N'	'Y'
d_2	'Y'	'Y'	'N'
d_3	'Y'	'N'	'N'

Figure 1 Documents and keywords. 'Y' denotes the corresponding keyword appears in a document.

The rules produced by concept learning algorithms can be viewed as finding optimal subspaces covering only one category's documents and no any other documents are included. We denote this kind of subspace as *positive cover* through this paper. The training examples in concept learning consist of both positive examples and negative examples. 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 all categories. In each time, all documents in the corresponding category are called *positive documents* whereas other documents are called *negative documents*.

3. Our approach

To describe the categories of documents, simplicity of the rules is a very important criterion [9]. That is, the number of rules should be small, and each of the rules must be short. One rule corresponds to one position cover, the categorization of documents becomes to find all positive covers. We enumerate all possible keyword combinations, and calculate the numbers of positive documents and negative documents covered by every keyword combination. Then we delete all the positive documents covered by the biggest positive cover. This process is repeated until no positive document is left. The biggest positive covers of each step are considered as the rules that we want.

Now we analyze the keyword combinations. A keyword is a feature of text document. The value of the feature can be divided into two types, continue-value, and discrete value. Continuous value is represented by the number of keywords appeared in a document. Discrete value is either 'Y' or 'N' which represents whether the keyword appears in a document or not. In this paper, we employ the latter to extract keywords for each category. To avoid the sensitivity of "noise" word, we filter out those words that appear in all documents less than 40 times. Every keyword has 2 discrete values (such as 'Y' and 'N'). 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 *n=2*, 1-keyword combinations has four types: kw_0='Y', kw_0='N', kw_1='Y' and kw_1='N'. The 2-keyword combinations have four types as well: (kw_0, kw_1) = {('Y', 'Y'), ('Y', 'N'), ('N', 'Y'), ('N', 'Y')}. From 1-keyword, 2-keyword, ..., n-keyword, the number of all possible combinations of keywords is up to $\sum_{i=1}^{n} c_n^i \times (c_2^1)^i$.

To describe a category of text document, more than one positive cover may be needed. Figure 2 shows the algorithm to find the rules to describe a category. The first rule is selected from the biggest positive covers. Then the all the positive documents covered by the rule are deleted. The second rule is selected in the remaining documents. The procedure is repeated until no more positive documents remained. Finally, the disjunction of all rules is the description of a category. The line 3-10 gives all possible combinations of keywords. Each combination is checked against every document to find out whether the document is covered by the combination or not. In line 5, the numbers of positive and negative documents are calculated in terms of every combination. The function

pureCover($c_1,\ldots,c_{nc}$) in line 5 returns the number of positive documents covered by the $c_1,\ldots,c_{nc}$, which does not cover any negative documents. In **Figure 1**, pureCover(kw1='Y' ∩ kw2='N') = 2 because it only covers two positive documents. But pureCover(kw1='Y') = 0 because the rule "kw1='Y'" covers positive and negative documents. In line 11, the documents covered by the selected combination are deleted. Notes that the deleted documents must be in category A. From the documents, the next positive cover can be found in the same way. The program stops when all the positive documents are deleted. Finally, every positive document is covered by at least one positive cover.

```
Algorithm find rules
1.While positive data set <> Ø
2.    maxCover = 0
3.    for nc = 1 : n  {the number of combination changed from 1 to n}
4.       for each combination c1,...,cnc {check all keyword combinations}
5.            if (maxCover < pureCover(c1,...,cnc ))
6.                maxcover = pureCover(c1,...,cnc )
7.                 bestCover = c1,...,cnc
8.            endif
9.       endfor
10.   endfor
11.   delete all positive examples covered by bestcover
12.   if no positive cover is found
13.      return all bestCover
14.   endif
15. endwhile
```

Figure 2 Algorithm for finding rules

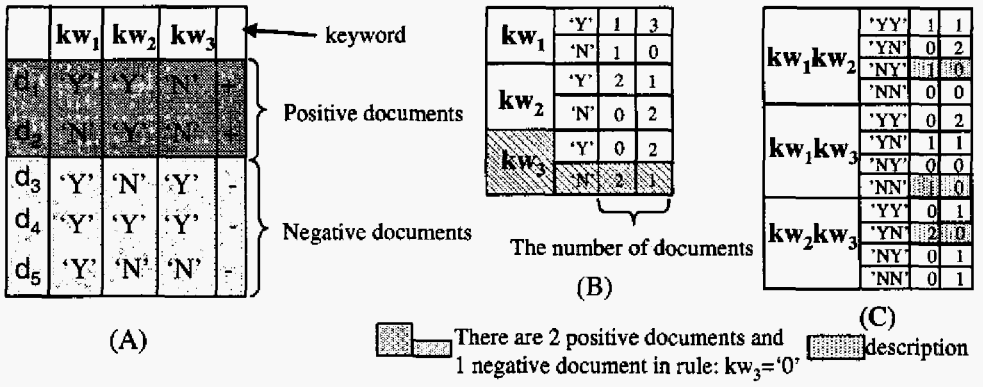

Figure 3 An example of algorithm 1. Subfigure (B) is 1-keyword combinations. Subfigure (C) is 2-keyword combinations.

Figure 3 (A) shows five training documents (d_1-d_5) that consisting of 3 keywords (kw_1, kw_2, kw_3). Each keyword has two discrete values 'Y' and 'N'. There are 2 positive documents and 3 negative documents. We enumerate all keyword combinations. Since there are only 3 keywords in documents, the combination of keywords has only 3 types: i.e. 1-keyword, 2-keyword and 3-keyword. 1-keyword combinations, such as kw_1='Y' or kw_1='N' as shown in **Figure 3** (B). In **Figure 3** (B), it can be seen that the rule of kw_1='Y' covers 1 positive document and 3 negative documents. The rule of kw_1 = 'N' covers 1 positive document but no negative documents, i.e. the rule of kw_1 = 'N' is the positive cover in 1-keyword rules. pureCover(kw_1='N')=1. Then we extend 1-keyword rules to 2-keyword rules. From the **Figure 3** (C), we find 3 rules cover only positive documents and no negative documents are covered. Since *pureCover*(kw_2='Y' ∩ kw_3='N')=2, it becomes the biggest positive cover, which describes the concept of the 5 training documents.

However, the enumeration based algorithm is only applicable in small keyword vector space. Most of the web documents involve much larger keyword vector spaces. The possible keyword combination set becomes very big. It costs very much CPU time and space to find biggest positive cover. In the next section, we will explain the mechanism of pruning irrelevant keyword combinations and getting small set of candidates.

3.1 Pruning power

Positive cover is represented with a conjunctive literal. For example, (kw_1='Y' ∩ kw_2 = 'N'). The description of concept can be viewed as a disjunction of positive covers or rules. In the enormous possible combinations, we observed that most of them can be pruned without affecting to find the biggest position cover. For example, in **Figure 3** (B), rule kw_2='N' does not cover any positive documents. Its any conjunctive rules will not cover any position documents as well. So the rule kw_2='N' can be pruned. To explain pruning power formally, we denote a rule as R, and *cover*(R) as a function that calculates the

number of positive documents covered by rule R. Unlike the function *pureCover*(R) which appeared in the **Figure 1**, *cover*(R) does not consider whether R covers negative documents or not. For example, in **Figure 3**, *pureCover*(kw_1 = 'Y') = 0, but *cover*(kw_1='Y') = 1.

LEMMA 1. The number of positive documents covered by rule R is represented as *cover*(R). Then we have *cover*(R) $\supset$ *cover*(r $\cap$ R), where r is denoted as an arbitrary rule.

Proof: if a positive document $e \in$ *cover*(r $\cap$ R), then e is covered by R and r. So $e \in$ *cover*(r $\cap$ R_i). □

We calculate all positive covers of 1-keyword combinations as illustrated in **Figure 3** (B). In all 1-keyword rules, we can get the best cover which covers the most number of positive documents but no negative documents are covered. In the example, *pureCover*(kw_1='N') = *cover*(kw_1='N') = 1. So kw_1='N' is the best rule among 1-keyword rules. This means that we can omit all combinations that cover the number of positive documents less than 1, because they can not be a better rule than kw_1='N'. That is, {kw_1='Y', kw_1='N', kw_2='N' and kw_3='Y'} and their combinations can be pruned. Only two 1-keyword rules (kw_2='Y' and kw_3='N') and their combinations become the candidates of concept description. We can easily calculate that the candidate (kw_2='Y' $\cap$ kw_3='N') becomes the final answer. In the 2-keyword rules, 11/12 combinations are pruned. Below we formally explain the pruning power in our algorithm by using **Figure 4.**

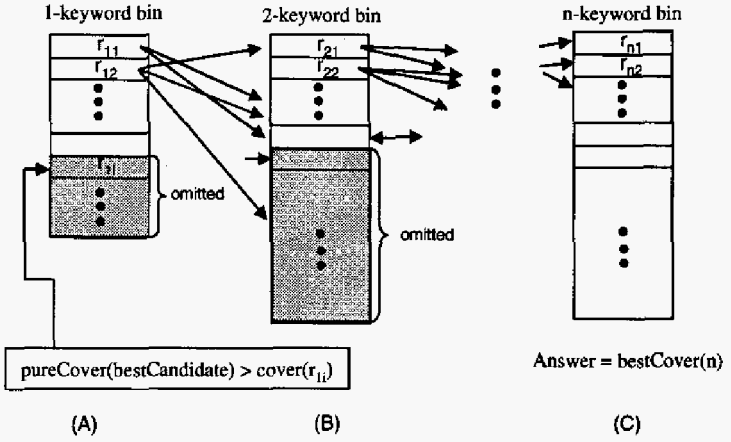

Figure 4 pruning power for finding candidates

First, we calculate the covers of n 1-keyword rules: *cover*(kw_1), *cover*(kw_2), ..., *cover*(kw_n). Then we sort them into a queue on descend. *cover*(r_{11}) $\geq$ *cover*(r_{12}) $\geq$ ... $\geq$ *cover*(r_{1n}) as shown in Figure 4 (A). From the queue, we find out a rule that has the biggest positive cover which is denoted as *bestCandidate*. In Figure 4 (B), *bestCandidate* is rule kw_1='N'. In Figure 4 (A), rule r_{11} usually cover more number of positive documents than *bestcandidate*, i.e. *cover*(r_{11}) > *pureCover*(*bestcandidate*). It is because r_{11} usually covers negative documents. Second, we constitute 2-keyword rules. Since the rule r_{11} covers the most number of positive documents in 1-keyword rules, it is selected to extend to 2-keyword rules. By combining with each keyword except itself, 2-keyword rules are constituted. At the same time, *bestcandidate* is updated when a bigger *pureCover*(R) appeares. Following r_{11}, the second best candidate 1-keyword rule r_{12} will be expanded to 2-keyword rules. However, we must confirm whether r_{12} can be pruned or not. If r_{12} does not satisfy *cover*(r_{12}) > *pureCover*(*bestCandidate*), r_{12} is pruned to expend into 2-keyword. The process continues until *cover*(r_{1i}) $\leq$ *pureCover*(*bestCandidate*). If we can not find a rule that covers more positive documents than the rule *bestcandidate*, the program stops. In most cases, the program will stop before the candidate combinations are expanded to n-keyword.

3.2 Reduction of keyword combinations

The number of keywords is usually very large, but most documents include just a small subset of the keywords. Even introducing the pruning power described in the previous section, the number of keyword combinations is still very large. It is costly to enumerate all keyword combinations. However, to describe the concept of document category, we only focus on the keywords appeared in the documents. And we do not need to consider the keywords that do not appear in the documents. Half combinations are reduced. The number of keyword combinations becomes $\sum_{i=1}^{n} C_n^i$, where n is the

number of keywords. For an example, the descriptions of concept can be (kw_i='Y' & kw_j ='Y').

3.3 The algorithm

A positive cover is represented by a rule that consists of 1, 2,..., *n* keywords. Using the pruning power described in the previous section, we propose a method to find the biggest positive cover. This 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. Similar to algorithm AQ15 [8], all positive documents covered by the rule are removed when a rule is found. The process illustrated in Figure 4 is repeated. That is, from the remaining positive and negative documents, the next biggest positive cover can be found. The process will continue until all positive documents are removed. Finally, the remaining ones will contain all negative documents and no positive documents. Every positive document is covered by at least one rule; no negative document is covered by the found rules.

Usually we have more than two categories of documents. To find each concept for all categories, we have to change positive documents to run program. For example, we have three categories documents, *A*, *B* and *C*. To find the concept of category *A*, we set all documents in category *A* to positive documents. All documents in category *B* and *C* is set to negative documents. Similarly, if we set all documents in *B* to positive documents and other documents to negative documents, we can find the concept to describe category *B*.

4. Experiment

To confirm the utility of concept learning algorithm, we test a set of web documents, which is divided into ten categories. We use keywords to describe the concepts of different categories. To evaluate our approach, we show the pruning power of keyword combinations. Since our program is based on memory, if there are too many combinations to be calculated, our approach loses its practicality. As shown in [8], the rules produced must satisfy completeness, consistency and simplicity. The first two conditions are essential. The simplicity means that the rules should be short and reflect the characters of the categories.

4.1 Dataset

The dataset has 314 web documents collected from various University of Waterloo web sites. It was downloaded from http://pami.uwaterloo.ca/~hamm ouda/webdata/ [3]. Ten categories and the number of documents in the categories are listed 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 delete all stop words, such as "a", "an", "on", and change all words into their root, for example "fishing" → "fish". If a word appears in the documents rarely, we treat it as a noise in the classification of text document. So we delete less 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.

4.2 Discussion of the experiment

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 take 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 to constitute first rule is shown on the Table 1. The number of candidates kept in bins is very small. The algorithm can be run based on memory in a usual computer.

Category#	Bin0	Bin1	Bin2	Bin3	Bin4	Bin5	Bin6	Bin7
Category 0	550	33	0	0	0	0	0	0
Category 1	465	154	1	0	0	0	0	0
Category 2	544	78	5	0	0	0	0	0
Category 3	560	259	236	60	0	0	0	0
Category 4	485	2172	510	309	40	9	1	0
Category 5	421	529	290	245	81	31	5	0
Category 6	470	2002	373	42	10	0	0	0
Category 7	444	157	206	231	144	49	9	1
Category 8	498	219	15	3	0	0	0	0
Category 9	516	46	5	1	0	0	0	0

Table 1 The sizes of all first rules of 10 categories

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 Figure 5, 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". Moreover, unlike the rules produced by C4.5 [10], every rule produced by our algorithm does not cover any negative examples. The numbers of documents covered by rules are shown in the right column. Since the cover may be overlapped, the total number of covered documents of

one category is more than the number of positive examples in this category.

The categories 8 and 9 (snowboarding skiing and winter Canada) are two similar categories. Snowboarding skiing occurs in winter. During winter in Canada, many people play snowboarding and 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. Category 6 and category 7 are also similar categories. The conjunction with one or two words can describe the two categories.

5. Conclusion

Concept learning is a fundamental topic in the field of machine learning field. Many learning algorithms such as AQ15, ID3 and CN2 have been applied in real datasets. In this paper, we proposed a new robust concept learning algorithm to find rules to represent document categories. The algorithm is based on enumeration of combination of features. To avoid the vast number of keyword combinations, which is usually impossible to implement base on memory of usual personal computer, we have introduced pruning power to create k-keyword bins, the size of bins is reduced drastically and the algorithm can be run in normal personal computers. 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. An, J. and Chen, Y.: Concept Learning of Text Documents. Web Intelligence 2004: 698-701
2. Clark, P and Niblett, T: The CN2 Induction Algorithm. Machine Learning 3: 261-283 (1989)
3. Hammouda, K. M., Kamel, M. S.: Phrase-based Document Similarity Based on an Index Graph Model. ICDM 2002: 203-210
4. Lewis, D. D.: Naïve (Bayes) at forty: The independence assumption in information retrieval. ICML 1998, 148-156
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. Carbonell, J. G. and Mitchell, T. M.: "Machine learning an artificial intelligence approach", Morgan Kaufmann Publishers, INC., 1983
9. Mitchell, T. "Machine learning", McGraw Hill, 1997
10. Quinlan, J. R.: Induction of Decision Trees. Machine Learning 1(1): 81-106 (1986)

Rules produced by our algorithm

Category 0	bear & attack	(30|0)
Category 1	network & switch ||	(24|0)
	network & ist ||	(23|0)
	address & ip ||	(19|0)
	source & ist	(5|0)
Category 2	transport & road & program ||	(19|0)
	transport & traffic	(17|0)
Category 3	work & career & interest & person ||	(24|0)
	career & email ||	(16|0)
	volunteer & employ ||	(20|0)
	skill & assist ||	(12|0)
	interest & experiment & type ||	(17|0)
	public & career & update	(10|0)
Category 4	student & length & return ||	(13|0)
	ordinary & report & field & write ||	(12|0)
	concern & employer & write ||	(5|0)
	return & ordin & write ||	(3|0)
	follow & space ||	(3|0)
	office & college & write ||	(2|0)
	art & term & faculty & engine ||	(6|0)
	allow & student ||	(5|0)
	average & op	(2|0)
Category 5	page & update & campus & university ||	(9|0)
	cause & health ||	(8|0)
	waterloo & page & November ||	(8|0)
	accommodation & charge	(1|0)
Category 6	home & fly & river ||	(12|0)
	fish & experience & package & trip ||	(9|0)
	fish & update & link & river ||	(3|0)
	fish & strong & river ||	(3|0)
	choose & February	(1|0)
Category 7	raft & tour ||	(12|0)
	reserve & profession & raft ||	(9|0)
	join & whitewater ||	(8|0)
	country & raft ||	(7|0)
	thing & whitewater & adventure	(4|0)
Category 8	inform & snowboard ||	(13|0)
	function & snowboard ||	(9|0)
	time & snowboard	(7|0)
Category 9	snowfall & rain ||	(17|0)
	snowfall & rang & amp ||	(5|0)
	snowfall & effect	(9|0)

Figure 5 The rules produced by enumeration based algorithm