Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 3 Issue 1, January - 2017

A Study on Naive Bayes Text Classification Algorithm Based on Position Weighting

[1]Yonghe Lu
School of Information Management
Sun Yat-sen University
Guangzhou, China
luyonghe@mail.sysu.edu.cn

[2]Jinghuang Chen
School of Information Management
Sun Yat-sen University
Guangzhou, China

[3]Jianhua Chen
School of Information Management
Sun Yat-sen University
Guangzhou, China

***Abstract*—The Naive Bayes algorithm in text classification has the limitation of the hypothesis: each attribute is independent in the algorithm. Attribute weighting is an effective method to improve Naive Bayes algorithm, and different positions of feature words in a document have different impacts on text classification. Therefore, in this paper, we introduce the position weighting values into Naive Bayes algorithm and we propose Position-Bayes algorithm. In order to get a higher text classification precision, Particle swarm optimization (PSO) algorithm is used to optimize the position weighting values. The experimental results show that compared with Naive Bayes algorithm, Position-Bayes algorithm has better performance in the text classification precision in different dimensions.**

Keywords—Naive Bayes algorithm; text classification; position weighting; particle swarm optimization algorithm

I. INTRODUCTION

Text classification is the process of automatically classifying specific text according to its content into one or more predefined text categories [1]. These classification algorithms include Naive Bayes, KNN, Support Vector Machine (SVM), Decision Tree, Neural Network and so on. Among them, Naive Bayes algorithm has the advantages of faster classifying and higher classification precision, and has been mainly used in spam filtering [2], text classification [3][4], credit evaluation of financial industry [5][6], face recognition [7], traffic modeling [8] and other fields.

However, Naive Bayes algorithm is based on the assumption of “conditional independence”. In fact, there are correlations among feature attributes. Therefore, many scholars have improved Naive Bayes algorithm.

Firstly, improving the methods of feature selection makes the feature attributes more independent. For example, Y. Zhang et al. improved Naive Bayes classification algorithm by merging the related feature words to make the feature attributes more independent [9]. Y. Zhang and L. Zhang removed redundant correlated feature words by using associated feature algorithm, which improved the performance of Naive Bayes classification algorithm [10].

Secondly, improving Naive Bayes classification algorithm makes it reflect the correlations among the attributes or weaken the impact of the “conditional independence” hypothesis on text classification. For instance, Friedman et al. proposed TAN (Tree Augmented Naive Bayes) tree-model to make Naive Bayes contain the dependencies among attributes [11]. Bai et al. clustered the word clusters according to the probability distributions of the feature words in the training set, and then established the ordered subsequences by calculating the mutual information among the words and the clusters to improve the accuracy of Naive Bayes classification algorithm [12]. Lee extracted strongly correlated keywords and calculated the prior probability of keywords to improve the performance of Naive Bayes classification algorithm in the field of text classification [13]. Y. Chen et al. constructed the three weighting values including discrimination, representativeness and word frequency, which improved the Naive Bayes algorithm, and the experiments on Uyghur corpus show that the improved algorithm can reach a higher classification precision [14]. L. Jiang et al. proposed a new model LWNBTC to reduce the effect of the attributes dependency on text classification by locally weighting method [15].

Because the existing methods do not consider the impacts of the positon of feature attributes in documents on text classification, we propose a position-weighted Naive Bayes classification algorithm, also called “Position-Bayes”, to reduce the impact of the "conditional independence" hypothesis of Naive Bayes algorithm, and the position weighting values of each feature attribute are calculated by Particle Swarm Optimization (PSO) algorithm. Finally, our experimental results show that the performance of Position-Bayes algorithm is better than Naive Bayes algorithm.

II. POSITION-BAYES ALGORITHM

Generally, the title of a document, the sentence in the front of a document and the sentence at the end of a document contain more information about the

document. Some scholars have introduced the positions of sentences into information extraction and have improved the performance of information extraction [16]. Considering the different posteriori probability density of each category which is caused by the difference of the position of the feature words in documents, we propose a position weighting method to improve the Naive Bayes text classification algorithm, and the details are as follows.

Let $W = \{W_1, W_2, \ldots\ldots, W_n\}$ represent n different characteristic attributes, n represent the number of feature dimensions in the text classification, $C=\{C_1, C_2, \ldots\ldots, C_m\}$ represent a categories set, m represent the number of categories, w_i represent a specific value of W_i, and $X=\{w_1, w_2, \ldots\ldots, w_n\}$ represent an instance of a document. And the procedures are as follows.

Step 1: Divide the words of an article into k segments according their positions, and initialize the weighting value of each segment. The value of k is based on the specific length of an article or specific scene. The corpus used in this paper is Reuters Corpus and we set k as 10. According to (1) and (2), we divide the words of a document into k segments:

$$num(position) = F\left(\frac{position-1}{length} \times 100/k\right) \quad (1)$$

$$F_w(position) = pos[num(position)] \quad (2)$$

Where $position$ represents the position of the feature word. For example, if the feature word t is the 5th word in the document, $position$ will be set as 5. $length$ represents the length of the document. In this paper, we use the number of words to measure the length of the document. For example, if there are 150 words after word segmentation in the document, $length$ will be set as 150.

And $F(\,)$ is a rounded down function, such as $F(1.6) = 1$, $F(0.1) = 0$. Equation (1) is used to calculate the value of segment where the feature word appears in the document. It can be deduced that when k = 10, if the feature word appears at the top 10% in the document, the value of position is 10% of $length$, so $F(\frac{position-1}{length} \times 100/k)$ is 0; if position is between top 10% and top 20% (including 20%), $F(\frac{position-1}{length} \times 100/k)$ is 1; and so on. Therefore, the values corresponding to 1st segment to k^{th} segment are 0, 1, 2, ..., $k-1$, respectively.

Where $pos[\,]$ is a k-dimension array containing position weighting adjusted values in the document which is divided into k segments, and its indices are from 0 to $k-1$. It can be deduced that $pos[0]$ represents the position weighting adjusted value of the 1st segment in the document, $pos[1]$ represents the position weighting adjusted value of the 2nd segment, ..., and pos[k-1] represents the position weighting adjusted value of the k^{th} segment. And the values of the array $pos[\,]$ need to be optimized by Particle Swarm Optimization algorithm.

Step 2: Calculate the adjusted value of position weighting of each feature word corresponding to each document. The adjusted value of position weighting of the feature word W_i in a document d_k is calculated according to (3):

$$Doc\_Pw(d_k, W_i) = \sum_{j=1}^{m(W_i,d_k)} F_w\left(Position_{(j,i)}\right) \quad (3)$$

Where $m(W_i, d_k)$ is the frequency of the feature attribute W_i appearing in the document d_k. $Position_{(j,i)}$ represents the value of the position where the feature attribute W_i appears for the j[th] time in the document d_k. For example, the position where W_i appears for the 2nd time in the document d_k is 8th, then $Position_{(2,i)}$= 8.

Step 3: Calculate the adjusted value of position weighting of each feature attribute in the training set and the testing set. The adjusted values of position weighting of feature attribute W_i in the category C_j of the training set and in the testing document X are calculated respectively as (4) and (5):

$$Train\_Pw\left(C_j, W_i\right) = \sum_{k=0}^{M(W_i,C_j)} (Doc\_Pw(d_k, W_i)) \quad (4)$$

$$Test\_Pw(W_i) = Doc\_Pw(X, W_i) \quad (5)$$

Where $M\left(W_i, C_j\right)$ is the number of documents with feature attribute W_i in the category C_j of the training set, and the category of d_k is C_j. $Train\_Pw\left(C_j, W_i\right)$ is the adjusted value of position weighting of feature attribute W_i in the category C_j of the training set. $Test\_Pw(W_i)$ is the adjusted value of position weighting of feature attribute W_i in the test document X.

Step 4: Add the values of position weighting of each feature attribute in the training set and the testing set to the Naive Bayes classification algorithm, and classify the documents in the testing set. The classification result of the document X can be calculated by the improved Naive Bayes classification algorithm according to (6):

$$C* = \underset{C_j \in C}{arg\,max} \left(\prod_{i=1}^{n} (P(w_i|C_j) * P(C_j) * Train\_Pw\,(C_j, W_i) * Test\_Pw(W_i))\right) \quad (6)$$

III. The Optimization of The Position Weighting Values

In order to find the optimal position weighting values, we use Particle Swarm Optimization algorithm to make an iterative calculation, and the details are as follows.

The velocity of the particle i is denoted as $V_i=(v_{i1}, v_{i2}, \ldots, v_{iD})$, the position of the particle i is denoted as $X_i=(x_{i1}, x_{i2}, \ldots, x_{iD})$, and the best position where the particle i has been is denoted as $P_i=(p_{i1}, p_{i2}, \ldots, p_{iD})$, also called p_{best}. The best position where all the particles in the group is denoted as $P_g=(p_{g1}, p_{g2}, \ldots, p_{gD})$,

also called g_{best}. All particles have a fitness value calculated by the fitness function. For the particles of each generation, the velocity and position of their d[th] dimension can be calculated according to (7) and (8):

$$v_{id} = w \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id}) \quad (7)$$

$$x_{id} = x_{id} + v_{id} \quad (8)$$

Where w represents inertia weight, c_1 and c_2 are acceleration constants, and *rand()* and *Rand()* are two random values which vary from 0 to 1. In this paper, the number of dimensions of the particles is set as 10, respectively corresponding to the position weighting values of 10 segments. For example, the first dimension of the particle represents the position weighting value of the feature words in the first segment (that is pos[0]), the second dimension of the particle represents the position weighting value of the feature words in the second segment (that is pos[1]), and so on. c_1 and c_2 are set as 2, the number of particles are set as 30, and the number of algorithmic iterations is 30.

In this paper, the fitness function is set as the precision of text classification in order to find the better position weighting value of each feature word.

$$Fitness() = \frac{T}{N} \quad (8)$$

Where T is the number of documents which are correctly classified in testing set, and N is the number of all documents contained in testing set.

IV. Experiments and Analysis

A. Experimental Environment

The environment of the experiment is Win7 64x 4GB Memory. And the text classification platform used in the experiment is based on the development of Java and environment of Eclipse, which integrates all the processes and common algorithms about text classification [17]. In this paper, Position-Bayes algorithm is compared with the Naive Bayes algorithm in the same experimental environment except for the classification algorithm by using control variate method.

B. Experimental Procedures

The corpus used in this paper is Reuters-21578 Corpus, and we select eight categories for text classification. They are acq, crude, earn, grain, interest, money-fx, ship and trade. The corpus is divided into testing set and training set according to the ratio of 1 to 2, and the detail distribution of the number of documents in each category is shown in Table I.

TABLE I. The Categories Distribution of Reuters-21578 Corpus

Category	Training set	Testing set
acq	1596	696
crude	253	121
earn	2840	1083
grain	41	10
interest	190	81
money-fx	206	87
ship	108	36
trade	251	75
Total	5481	2189

The experimental procedures are as follows.

Step 1: Word segmentation. The word segmentation module used in the experiment is ICTCLAS2013 developed by Chinese Academy of Sciences. The indexing module is Lucence.

Step 2: Feature selection. In this paper, chi-square test feature selection algorithm is used in the experiment. The number of dimensions are initialized as 100, and then the number of dimensions of each experiment is increased by 100 on the previous experiment until the precision drops.

Step 3: Feature weighting calculation. The method TF-IDF is used to calculate the feature weighting values in the experiment.

Step 4: Text classification. We use Position-Bayes algorithm and Naive Bayes algorithm respectively to classify the documents and calculate the classification precision.

C. Results and Analysis

Under the condition of the optimal position weighting values, the precisions of text classification in different dimensions are shown in Table II. As the number of dimensions increases, the trend of classification precisions of Position-Bayes is similar to that of Naive Bayes which is shown in Fig. 1. From 100 dimensions, as the number of dimensions increases, the classification precision also increases, and the precision reaches the highest when the number of dimensions is 600. After 600 dimensions, the feature words begin to be redundant, so the classification precision begins to drop. And from Fig. 1 and Fig. 2, we can see that the precision of Position-Bayes are higher than that of Naive Bayes from 100 to 800 dimensions.

TABLE II. THE CLASSIFICATION PRECISIONS OF TWO ALGORITHMS IN DIFFERENT DIMENSIONS

Dimensions	Precision(%)	
	Naive Bayes	***Position-Bayes***
100	71.02	73.41
200	77.57	78.35
300	81.22	83.23
400	90.82	92.33
500	91.73	92.92
600	92.10	93.33
700	83.23	84.42
800	83.28	83.33
Average	83.88	85.17

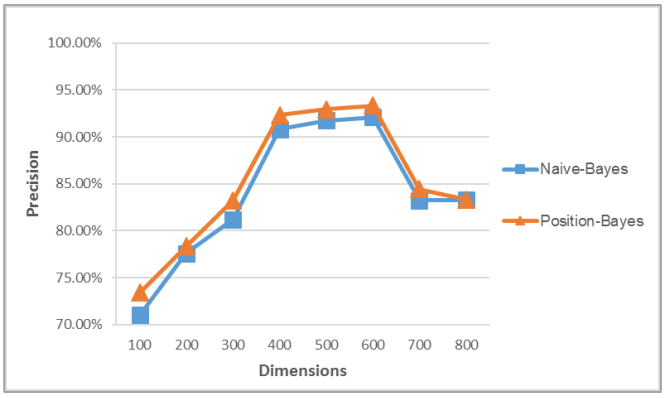

Fig. 1. *The trend of classification precision with the increasing of dimensions*

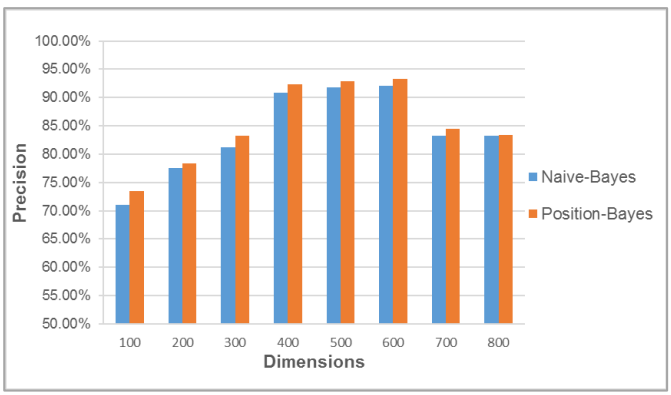

Fig. 2. *The comparison of the precisions of Position-Bayes and Naive Bayes*

Then we use paired sample t-test for the above results by SPSS in order to test whether it is statistically significant or not. The confidence level is set as 0.95 and the results are as follows.

TABLE III. PAIRED SAMPLE T-TEST RESULTS OF THE PRECISIONS OF TWO ALGORITHMS

	t	df	Sig. (two-sided)
Native- Bayes — Position-Bayes	−5.167	7	0.001

According to the Table III, the value of Sig. is 0.001 which is less than 0.05, so the experimental results are statistically significant. Therefore, the classification precisions of Position-Bayes are significantly different from those of Naive Bayes, and the performance of Position-Bayes is better than that of Naive Bayes.

After the optimization of PSO, when the classification precision is the highest, the position weighting value of each segment are shown in Table IV and Table V. In order to make the data be a unified comparison, all the values are processed by the function (9).

$$f(x) = x/max \tag{9}$$

Where x is the position weighting value of each segment and max is the maximum value among all the position weighting values in the same dimensions. Processed by the function, all the values are in the range of 0 to 1. The segment 1 represents the top 10% of the document, the segment 2 represents the top 10% to 20% of the document, and so on. In different dimensions, the position weighting values of the front and the latter segment are relatively stable. The position weighting values of the 1st and the 2nd segment are relatively bigger, the values of the 9th and the 10th segment are relatively smaller, and the fluctuation of the position weighting values of segment 3rd to segment 8th is relatively larger.

TABLE IV. THE POSITION WEIGHTING VALUES OF SEGMENT 1 TO 5 IN DIFFERENT DIMENSIONS

Segments / Dimensions	1	2	3	4	5
100	1	1	1	0.804	0.462
200	0.985	1	0.322	0.466	0.536
300	1	1	0.659	1.00	0.447
400	1	0.849	0.553	0.504	0.872
500	1	1	0.46	0.582	0.718
600	1	1	0.632	0.779	0.837
700	1	1	0.459	0.663	0.321
800	1	1	1	1	1
Average	1.00	0.98	0.64	0.72	0.65

TABLE V. THE POSITION WEIGHTING VALUES OF SEGMENT 6 TO 10 IN DIFFERENT DIMENSIONS

Segments / Dimensions	6	7	8	9	10
100	0.297	0.411	0.129	0.507	0.391
200	0.386	0.841	0.861	0.257	0.577
300	0.58	1.00	0.211	0.305	0.494
400	0.205	0.464	0.239	0.421	0.32
500	0.382	0.707	0.303	0.16	0.476
600	0.46	0.607	0.291	0.352	0.281
700	0.344	0.448	0.897	0.276	0.247
800	1	1	1	0.404	0.543
Average	0.48	0.68	0.49	0.34	0.42

The classification precisions of 400, 500 and 600 dimensions are relatively higher and more stable, so we can observe further the position weighting values of 400, 500 and 600 dimensions, and the trends of the values are shown in Figure 3. Under the condition of 400, 500 and 600 dimensions, the fluctuation trends are similar and the position weighting values of the same segment are close to each other. As the position of segment becomes close to the end, the position weighting values become a downward trend. So, it can be concluded that the feature words in the top 20% of the document are more representative and contain more important information of the document, and they need to be assigned the higher weighting values.

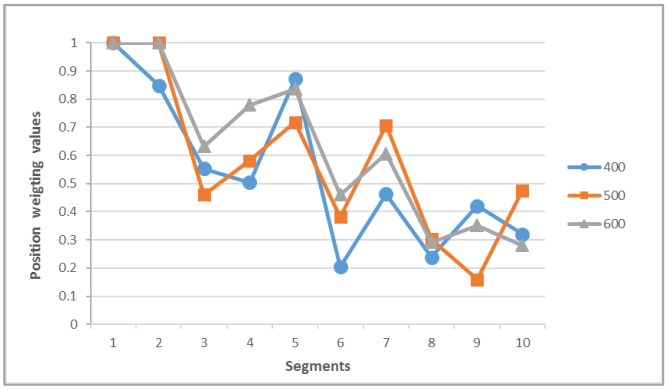

Fig. 3. *The trend of the position weighting values in the dimensions of 400, 500, and 600*

V. CONCLUSIONS

Naive Bayes algorithm is a commonly used and has a good performance of text classification. In this paper, we propose a position-weighted Naive Bayes classification algorithm to reduce the impact of the "conditional independence" hypothesis of Naive Bayes algorithm, and use PSO algorithm to find the optimal position weighting values of each segment. According to the experimental results, it can be concluded that compared with Naive Bayes algorithm, the improved Position-Bayes algorithm can do better in text classification and improve the precision of text classification. What's more, it can be deduced that the feature words in the top 20% of the document are more representative and contain more important information of the document, and they need to be assigned the higher weighting values, which can improve the precision of text classification.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (Grant No. 71373291), and Science and Technology Planning Project of Guangdong Province, China (Grant No. 2016B030303003).

REFERENCES

[1] Y. Yang, X. Liu. A re-examination of text categorization methods. International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 42-49, 1999.

[2] X. Shi, Y. Lin, Z. Chen. Mail Filtering Based on the Risk Minimization Bayes. Computer Science, vol. 29, no. 8, 50-51. 2002.

[3] M. M. Drugan, M. A. Wiering. Feature selection for bayesian network classifiers using the mdl-fs score. International Journal of Approximate Reasoning, vol. 51, no. 6, pp. 695-717, 2010.

[4] X. Zhou. Text classification model of uyghur based on improved bayes. Journal of Computational Information Systems, vol. 9, no. 11, 4319-4327, 2013.

[5] X. Li, C. Guo, K. Chen. Bayesian network consumer credit scoring models based on minimum overall risk rule. Application Research of Computers, vol. 26, no. 1, pp. 50-53, 2009.

[6] X. Wang. Study of Bayesian Network Classification Models and Its Application in Credit Scoring. Computer & Digital Engineering, vol. 38, no. 8, pp. 107-109, 2010.

[7] Y. Zeng, D. Feng, D. Fu. Face recognition algorithm of binary image by smallest risk Bayesian method. Computer Engineering and Design, vol. 32, no. 10, pp. 3511-3513, 2011.

[8] D. Wu, P. Chen. Bayesian minimum hazard control model of traffic accident. Journal of Traffic and Transportation Engineering, vol. 7, no. 6, pp. 266, 2007.

[9] Y. Zhang, J. Chen, Z. Xiong. Improved Naive Bayes Text Classification Algorithm. Journal of Guangxi Normal University : Natural Science Edition, vol. 25, no. 2, pp. 206-209, 2007.

[10] Y. Zhang, L. Zhang, J. Yan. Naive Bayes Text Classifier Based on Association Features. Journal of North western Polytechnical University, vol. 22, no. 4, pp. 413-416, 2004.

[11] N. Friedman, G. Dan, M. Goldszmidt. Bayesian network classifiers. Machine Learning, vol. 29, no. 2, pp. 131-163, 1997.

[12] L. Bai, H. Huang, S. Liu, Q. Yan. Naive Bayes Classifier Based on Bootstrap Average. Computer Engineering, vol.33, no. 15, pp. 190-192, 2007.

[13] L. H. Lee, D. Isa, W. O. Choo, W. Y. Chue. High relevance keyword extraction facility for bayesian text classification on different domains of varying characteristic. Expert Systems with Applications, vol.39, no. 1, pp. 1147-1155, 2012.

[14] Y. Chen, A. Halidan, D. Yiliyaer, A. Yaliqin. Uyghur text classification based on weighted improved Bayes. COMPUTER ENGINEERING AND DESIGN, vol. 35, no. 6, pp. 1999-2003, 2014.

[15] L. Jiang, Z. Cai, H. Zhang, D. Wang. Naive Bayes text classifiers: a locally weighted learning approach. Journal of Experimental & Theoretical Artificial Intelligence, vol. 25, no. 2, pp. 273-286, 2013.

[16] J. Liu, Y. Tan, J. Li, N. Yuan. Automatic Extraction Method of Chinese Text Theme Based on Multi-Factor. COM PUTER TECHNOLOGY AND DEVELOPMENT, vol. 20, no. 7, pp. 72-75, 2010.

[17] Y. Lu, Y. Peng, W. Liu, Building a Text Classification Platform for Scientific Research and Teaching. Journal of Modern Information, vol. 35, no. 9, pp. 56-62, 2015.