International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:7, 2012

An Semantic Algorithm for Text Categoritation

Xu Zhao

***Abstract*—**Text categorization techniques are widely used to many Information Retrieval (IR) applications. In this paper, we proposed a simple but efficient method that can automatically find the relationship between any pair of terms and documents, also an indexing matrix is established for text categorization. We call this method Indexing Matrix Categorization Machine (IMCM). Several experiments are conducted to show the efficiency and robust of our algorithm.

***Keywords*—**Text categorization; Sub-space learning; Latent Semantic Space

I. Introduction

ALONG with the rapid development of the INTERNET, information lying in the web increases at a progressional speed. Facing these vast and complicated resources, how to efficiently use these huge resources and find some key information from them is an attractive field for researchers to explore. Text categorization is one of these fields which can be seen as a supervised learning task for assigning text documents to predefined categories. Many information retrieval problems such as filtering, searching or routing benefit from the text categorization.

Like some other methods, such as Optimal Character Recognition (OCR), text categorization essentially is a pattern recognition task, and lots of classical algorithms have been proposed to deal with this field, among which Support Vector Machine (SVM) is a typical algorithm. Many researchers have used SVM in their researches to deal with some difficult recognition tasks.

Although some of these algorithms have got beautiful results in the field of pattern recognition, they all focus on how to separate the data points in the high dimensional space, instead of the data points themselves. When the original data points have some noises or something that are not profitable to the recognition task, some of these algorithms may lose their ideal results.

Contrary to them, here we put our focus on the data points and propose a simple but efficient method for text categorization task. This method is based on the latent semantic space. We first get the term-document matrix and then Singular Value Decomposition is used to find the axis of the latent semantic space. After this, we project all the documents and the terms from the original space to the semantic space, in which we find the relationship between any pair of the term and the document. An indexing matrix is established by which the text categorization task can be executed efficiently.

Xu Zhao is with Shanda Innovations, Tsing Hua Tongfang Hi-Tech plaza, No.1 Wangzhuang Road,Haidian District, Beijing China, 100083 (e-mail: zhaoxu166@ gmail.com).

The rest of this paper is organized as follows: in section II, Latent Semantic Indexing is reviewed; section III gives the detailed description of our method; then in section IV, some experiments are conducted to show the efficiency and robust of our algorithm; finally, in section V, we give the conclusions.

II. Review of Latent Semantic Indexing (LSI)

As a document, all its words have some semantic relationships between each other more or less. A source of information about semantic similarity in the given context is co-occurrence analysis: if two terms co-occur in documents very frequently (in a given corpus) they can be considered as semantically related. Incorporating co-occurrence information in a learning system for exploiting semantic similarity between words would seem to be a very expensive task. However, a technique has been developed in the information retrieval literature that can extract this information automatically [1]-[6].That is Latent Semantic Indexing (LSI).

LSI projects all documents into a space with "Latent Semantic Dimensions", where co-occurring terms are projected in similar directions, while non co-occurring ones are projected in very different directions. In such a space two documents can have a high similarity even if they do not share the same terms (as long as they each use terms frequently co-occurring in the corpus) [1].Therefore, the key point is to depict the relationships between the terms and the documents. Usually we can use document-term matrix [7]. For m documents and n terms, let

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \cdots & \cdots & \cdots \\ C_{m1} & \cdots & C_{mn} \end{bmatrix}$$

where rows are indexed by the documents of the corpus and columns by the terms. The (j,i)-th entry of C gives the frequency of term t_i in document d_j. By weighting the terms, we can get a more reasonable relationship between terms and documents. It is obvious that different weighting methods can lead to different impacts [8], [9].

We transpose C to get the term-document matrix C^T. Then we apply Singular value decomposition (SVD) [10] to C^T and get

$$C^T = UQV^T \tag{1}$$

Where $U^TU=V^TV=In$ and $Q = diag(\sigma_1,...,\sigma_n), \sigma_i > 0$, for $1 \le i \le r$ (r is the rank of matrix C^T), $\sigma_j = 0$, for $j \ge r+1$. The first r columns of the orthogonal matrixes U and V define the orthonormal eigenvectors associated with the first r nonzero eigenvalues of C^TC and CC^T, respectively. Now we need to choose a value k ($0 < k \le r$) to approximate C^T by

$C_k^T=U_kQ_kV_k^T$, and the space spanned by U_k is the semantic space we want to get. Notice that C^TC is the covariance matrix of the corpus, so the first k diagonal elements of matrix Q are the variances corresponding to the directions of the principal axes that span U_k. So, the larger the dimension k of the subspace U_k, the greater percentage of the variance that is captured.

According to this idea, we choose k ($0 < k \leq r$) that makes $\sigma_k >> \sigma_{k+1}$, which means that we want to capture the variances as many as the original dataset, and we get

$$C_k^T=U_kQ_kV_k^T \quad (2)$$

where U_k is made up of the first k columns of U ,and Q_k is made up of the first k rows and the first k columns of Q, and V_k is made up of the first k columns of V. So, C_k^T is the approximate representation of C^T in the least squares sense. We can project all the documents from the original space into the semantic space spanned by U_k, and get every document's representation in the semantic space.

III. Indexing Matrix Categorization Machine (IMCM)

Indexing Matrix Categorization Machine (IMCM) can be viewed as a huge text relationship warehouse by which we can get the relationship between any pair of term and document. In section A and section B we will show how to find this semantic relationship automatically and the whole algorithm of how to get the indexing matrix is given. In section C Indexing Matrix Categorization Machine is illustrated

As mentioned above, we use LSI as an auxiliary tool to establish the huge relationship warehouse since the Latent Semantic Space is the basic elements of our algorithm.

Concretely, we first centralize the term-document matrix to locate the origin at the centroid of the corpus. We just the same use C^T to represent the term-document matrix which has been centralized. By applying SVD to C^T, we can get the derived k ranked approximate matrix $C_k^T=U_kQ_kV_k^T$. So the Latent Semantic Space is expanded by U_k.

A. Compute the semantic relationship between any pair of terms

Assume that term I and J are represented as I=(0,0,…,i,0,…0) and J=(0,0,…,j,0,…0), respectively, when I ≠ J, i and j are at the different dimension. The only non-zero dimension indicates the weight of the term to this "document". In this way, we have the representation of term I and J in the semantic space:

$$I_{semantic} = I * U_k, \quad (3)$$
$$J_{semantic} = J * U_k. \quad (4)$$

where $I_{semantic}$ and $J_{semantic}$ are all k ranked row vectors. Let $H = U_k*U_k^T$, the inner product of $I_{semantic}$ and $J_{semantic}$ can be computed as: $(I_{semantic}, J_{semantic}) = I*U_k * U_k^T * J^T = I*H*J^T = i*j*H_{ij}$. Consequently, the similarity between I and J in the semantic space can be computed as follows:

$$\cos(I_{semantic}, J_{semantic})$$
$$= (I_{semantic}, J_{semantic}) / \sqrt{(I_{semantic}, I_{semantic})(J_{semantic}, J_{semantic})}$$
$$= i*j*H_{ij} / \sqrt{i*i*H_{ii} * j*j*H_{jj}}$$
$$= H_{ij} / \sqrt{H_{ii} * H_{jj}}$$

Let

$$T_{ij} = H_{ij} / \sqrt{H_{ii} * H_{jj}} \quad (5)$$

We obtain the matrix T whose elements are the similarity between terms in the semantic space.

B. Compute the semantic relationship between any pair of terms and documents

In order to get the semantic relationship between any pair of terms and documents we must devise a structure by which we can terms and documents interact with each other.

Here, we get help from section A. We first get matrix T whose elements are the similarity between terms in the semantic space, and we use a different form to represent a term in the semantic space, namely, every row of T can be regarded as a vector which represents a term, and every entry of the vector is the similarity between this term and the other terms in the semantic space.

What should be done next is to represent every document by its similarity with every term. Notice that not all terms belong to a document are positive to this document, some terms may be noise.

Here, in view of the computation efficiency we deem one term which appears more than twice in a document is the term that has a topically indicative effect, and name it "topic term". (Of course a term that appears more than 3 times or more times in a document also can be a criterion for selecting "topic term", but it will take a bit of time). Meanwhile, we view the terms which appear only one time in a document as noise. In this way, we can represent a document as the centroid of its topic terms.

Assume that document d contains h topic terms $t_1, t_2, \ldots, t_h$ which appear $f(t_i)$ times in d, respectively. Then d can be represented as

$$\text{doc} = \sum_{i=1}^{h} (f(t_i) * T(t_i)) / \sum_{i=1}^{h} f(t_i) \quad (6)$$

where $T(t_i)$ corresponds to the row vector of term t_i in the matrix T. Vector doc is the centroid of all d's topic terms. By this representation, every entry of doc is a similarity between document d and a term. Obviously, the entries corresponding to d's topic terms will be larger than the noise terms.

According to this way, we compute all m document vectors, and compose m*n matrix B. Every row of B represents a document, and the (i,j)-th entry of B gives similarity between the i-th document and the j-th term. Let

$$A=B^T. \quad (7)$$

Obviously, the (i,j)-th entry of A gives the similarity between i-th term and the j-th document, and the i-th row of A gives the similarity between the i-th term and all the documents in the corpus. By sorting every row of A, respectively, we can get the order of every term with all the documents.

The whole algorithm procedure of getting Indexing Matrix is given below:

1. Pretreat the corpus, and get the term-document matrix C^T;
2.Centralize C^T, and apply SVD to get the k ranked approximated matrix $C_k{}^T=U_kQ_kV_k{}^T$;
3.Use equality (5) to obtain matrix T;
4.Compute every document's centroid vector of all its topic terms, and store the result in the matrix B;
5.Let $A=B^T$, and sort every row of A to get the indexing matrix.

C. Indexing Matrix Categorization Machine (IMCM)

After getting the Indexing Matrix A (section B), we can find the semantic relationship between any pair of term and document. Then, we use vote to category a document. We give every categorization a counter to note the votes it has. When a document is coming, we should cut it and for every term the document contains if the term appears in the matrix A of our algorithm we will get the first K documents' labels with which the term has the largest semantic relationship and add corresponding votes to the counter of the categorization. After the same process to all the terms the document contains has been finished, the final decision can be done by return the label that has the most votes. The pseudocode is like this: (here for simplicity, we assume that there are only two classes, namely, C1 and C2)

```
Set counter[2] = {0}
For(every term in the text)
  If(indexing matrix A contains this term)
    Find the corresponding row r of the term
    For(i=1…K)
      If(A[r][i] belongs to class C1)
        Counter[1]++;
      Else
        Counter[2]++;
Return Counter;
```

In this case, we return a vector that contains the votes of all the classes, which can be used in the multi-class case easily.

IV. Experiments and Discussion

In this section, we will conduct some experiments to demonstrate the efficiency and robust of our proposed algorithm.

A. Comparison experiment results

The experiments are conducted on a set of 7500 web documents randomly download from the internet. There are 5 different categorizations, namely, 电影(Film), 乒乓球(Ping Pang), 美食(Eating), 财经(Finance) and 军事(Military Affairs). Every categorization has 1500 documents and we use 1000 documents to train, the left are used for test.

Here, we compare our method with the famous algorithm Support Vector Machine (SVM), since SVM has been deemed as the most efficient and robust algorithm for categorization works. Also here we make K in our algorithm equal 20, and the comparison results of changing parameter K will be given in the next section.

The experiments results are shown in table I and table II:

TABLE I
THE ACCURACY RATIO AND RECALL RATION OF IMCM.

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.912477	0.980000
乒乓球(Ping Pang)	0.995475	0.880000
美食(Eating)	0.915285	0.994000
财经(Finance)	0.944773	0.958000
军事(Military Affairs)	0.940552	0.886000

TABLE II
THE ACCURACY RATIO AND RECALL RATION OF SVM.

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.995833	0.478000
乒乓球(Ping Pang)	0.995444	0.874000
美食(Eating)	0.899420	0.930000
财经(Finance)	0.474738	0.996000
军事(Military Affairs)	1.000000	0.510000

From the results, we can see that the average performance of IMCM is better than SVM for a lot. Through the accuracy ratio of SVM may surpass IMCM for some special categorizations, for instance, "电影(Film)" and 军事(Military Affairs). But the recall ratios of these two categorizations gotten by SVM are very low. On the contrary, the recall ratios gotten by IMCM are almost two times than that in SVM, which indicates that IMCM is more robust than SVM.

B. Discussion about parameter K used in IMCM

In this section, we will test the stability of IMCM with changing of parameter K. We change the parameter K by interval 5 and report the Accuracy Ratio and the Recall Ratio of the algorithm. The results are shown below:

TABLE I
THE ACCURACY RATIO AND RECALL RATION OF IMCM WITH K EQUALS 5..

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.912477	0.980000
乒乓球(Ping Pang)	0.997763	0.892000
美食(Eating)	0.980198	0.990000
财经(Finance)	0.844948	0.970000
军事(Military Affairs)	0.963387	0.842000

TABLE II
THE ACCURACY RATIO AND RECALL RATION OF IMCM WITH K EQUALS 10.

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.910615	0.978000
乒乓球(Ping Pang)	0.995516	0.888000
美食(Eating)	0.927239	0.994000
财经(Finance)	0.919694	0.962000
军事(Military Affairs)	0.958515	0.878000

TABLE III
THE ACCURACY RATIO AND RECALL RATION OF IMCM WITH K EQUALS 15.

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.913858	0.976000
乒乓球(Ping Pang)	0.995506	0.886000
美食(Eating)	0.915285	0.994000
财经(Finance)	0.941176	0.960000
军事(Military Affairs)	0.942308	0.882000

TABLE IV
THE ACCURACY RATIO AND RECALL RATION OF IMCM WITH K EQUALS 20.

Categorization	Accuracy Ratio	Recall Ratio
电影(Film)	0.912477	0.980000
乒乓球(Ping Pang)	0.995475	0.880000
美食(Eating)	0.915285	0.994000
财经(Finance)	0.944773	0.958000
军事(Military Affairs)	0.940552	0.886000

From these results we can see that there is no big disturbance during the changing of K, which demonstrates the robust of IMCM.

V. CONCLUSIONS

In this paper, we propose a simple but efficient method called Indexing Matrix Categorization Machine (IMCM) for text categorization task, which is based on the latent semantic space. By using the relationship between any pair of term and document, some counters are set to vote and return the class that has the most votes. Several experiments are conducted to show the efficiency and robust of our algorithm.

REFERENCES

[1] Cristianini, N., Lodhi, H., Shawe-Taylor, J.: Latent Semantic Kernels for Feature Selection. NeuroCOLT Working Group (2000), http://www.neurocolt.org
[2] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R.(1990). Indexing By Latent Semantic Analysis. Journal of the American Society For Information Science, 41, 391-407.
[3] Landauer, T. K., Foltz, P. W., & Laham, D. (this issue). An introduction to Latent Semantic Analysis. Discourse Processes .
[4] Foltz, P. W. (1996). Latent Semantic Analysis for text-based research. Behavior Research Methods, Instruments and Computers, 28(2), 197-202.
[5] Letsche, T.A. & Berry, M.W. (1997). Large-scale information retrieval with Latent Semantic Indexing. Information Sciences – Applications, 100, 105-137.
[6] Landauer, T. K. & Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis heory of acquisition, induction and representation of knowledge. Psychological Review, 104, 211-240.
[7] Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University, Press, Cambridge (2004)
[8] Dumais, S.T.: Improving the Retrieval of Information from External Sources. Behav. Res. Meth. Instr. Comput. 23, 229–236 (1991)
[9] Debole, F., Sebastiani, F.: Supervised Term Weighting for Automated T ext Categorization. In: SAC 2003, pp. 784–788. ACM Press, New York (2004)
[10] Berry, M.W., Dumais, S.T., O'Brien, G.W.: Using Linear Algebra for Intelligent Information Eetrieval. SIAM: Review 37, 573–595 (1995)