

Using DragPushing to Refine Concept Index for Text Categorization

Xueqi Cheng¹ (程学旗), Songbo Tan¹ (谭松波), and Lilian Tang²

¹*Division of Intelligent Software Systems, Institute of Computing Technology, Chinese Academy of Sciences
Beijing 100080, P.R. China*

²*Department of Computing, University of Surrey, U.K.*

E-mail: cxq@ict.ac.cn; tansongbo@software.ict.ac.cn; h.tang@surrey.ac.uk

Revised May 30, 2006.

Abstract Concept index (CI) is a very fast and efficient feature extraction (FE) algorithm for text classification. The key approach in CI scheme is to express each document as a function of various concepts (centroids) present in the collection. However, the representative ability of centroids for categorizing corpus is often influenced by so-called model misfit caused by a number of factors in the FE process including feature selection to similarity measure. In order to address this issue, this work employs the “DragPushing” Strategy to refine the centroids that are used for concept index. We present an extensive experimental evaluation of refined concept index (RCI) on two English collections and one Chinese corpus using state-of-the-art Support Vector Machine (SVM) classifier. The results indicate that in each case, RCI-based SVM yields a much better performance than the normal CI-based SVM but lower computation cost during training and classification phases.

Keywords text classification, information retrieval, machine learning

1 Introduction

Recent years have seen a tremendous growth of text documents in volumes available on the Internet. Accordingly the management and organization of text have become an important task. A major component in this task is to assign documents into a set of categories, known as Text Categorization (TC) or Text Classification. A number of machine learning algorithms have been introduced to deal with text classification, such as K-nearest Neighbour^[1], Centroid Classifier^[2], Naive Bayes (NB)^[3,4], Winnow^[5] and Support Vector Machines^[6].

The common and often daunting characteristics of text data representation is its extremely high dimensionality. In text classification community, a document is represented by a “bag-of-words”^[7], that is a large vector of vocabularies containing word frequency counts. Even a moderately sized document collection can lead to a dimensionality in thousands^[8]. As a result, this high dimensionality poses an open challenge for classification algorithms. Reducing the dimensionality without sacrificing classification performance is therefore very important for text categorization.

Dimension reduction is generally approached through Feature Extraction (FE)^[9] and Feature Selection (FS)^[10]. The traditional FE algorithms reduce the dimension of data by Principal Component Analysis (PCA)^[11], Linear Discriminant Analysis (LDA)^[12], Maximum Margin Criterion (MMC)^[13], Locally Linear Embedding (LLE)^[14] and Concept Index (CI)^[15]. FS algorithms reduce the dimension of data by selecting features from the original vectors directly. There are a few good feature selection methods^[16] that have been ap-

plied to select reasonable useful and relevant features for text classification, including Mutual Information (MI), Information Gain (IG), CHI Statistics (CHI), Document Frequency (DF), PCA based algorithm^[17], Margin based algorithm^[18] and SVM-based algorithm^[19].

In despite of their soundness of theoretical analysis, many FE algorithms suffer from high computational cost, whilst FS algorithms are more popular for dimension reduction problems of real life text data. However, the Concept Index technique (CI) based on a simple theory turns out to be a very fast and efficient FE method, and yet yields better performance than some FS algorithms.

The key idea behind CI scheme is to express each document as a function of various concepts (centroids) present in the collection. This is achieved by first finding groups of similar documents, and then using these groups to derive the axes of the reduced dimensional space. In the case of supervised dimensionality reduction, CI employs centroids of pre-existing classes of documents as concepts. To a high degree, the representative ability of centroids plays a crucial role for CI.

However, the representative ability of centroids is often influenced by so-called model misfit brought up by a number of reasons, including the choice of features used in the model (e.g., the choice of terms vs. regular expressions, or the choice of the sources such as either from the whole document or from restricted sections), the choice of the number of features to use, the choice of scoring methods used in the vector space model (e.g., feature count vs. tf-idf method), the choice of similarity function (such as Euclidean Distance vs. cosine similarity or other methods), etc. Such factors may also compound. The more serious the model misfit, the poorer the rep-

representative ability will be and the poorer classification performance is going to be.

A novel approach, called “DragPushing”, for refining centroids, has been proposed by Tan *et al.*^[20] The DragPushing refinement strategy is based on taking advantage of misclassified examples in the training data to iteratively refine and adjust the centroids of text data. The performance of the implemented Refined Centroid Classifier is comparable, if not better, to that of state-of-the-art SVM classifier. Furthermore, the computational time of DragPushing scales linearly with the training documents, demonstrating the proposed Refined Concept Index (RCI) meets the performance and speed requirements in information retrieval community.

In this work, we conduct extensive experiments to compare RCI against traditional CI and other FS algorithms. It is worth mentioning that we only use SVM to test our method as SVM has been recognised as a reliable technique for text classification. The experimental results indicate that RCI achieved a significant improvement over the performance of CI, and the speed of SVM using RCI is much faster than using CI.

The rest of this paper is organized as follows. Section 2 describes CI and DragPushing refinement strategy. RCI is presented in Section 3. Experimental results are given in Section 4. Finally Section 5 concludes this paper.

2 Related Work

2.1 Concept Index

The concept-indexing algorithm computes a lower dimensional space by finding groups of similar documents and using them to derive the axes of the lower dimensional space.

In unsupervised setting, CI uses a clustering algorithm to partition the documents into k disjoint sets, S_1, S_2, \dots, S_k . For each set S_i , it computes the corresponding centroid vector C_i^N . These centroid vectors are scaled to unit length. Let $\{C_1^N, C_2^N, \dots, C_k^N\}$ be these unit length centroid vectors. Each of these vectors forms one of the axes of the reduced k -dimensional space, and the k -dimensional representation of each document is obtained by projecting it onto this space. This projection can be written in matrix notation as below,

$$d_k = dC \quad (1)$$

where C is the $W \times K$ matrix such that the i -th column of C corresponds to C_i^N . Similarly the k -dimensional representation of the entire collection is given by the following matrix,

$$D_k = DC \quad (2)$$

where D is a $T \times W$ document-term matrix (T is the number of documents, and W is the number of distinct terms in the collection).

In the case of supervised dimensionality reduction, CI uses the pre-existing classes of documents as the

groups of similar documents. In this case, the rank of the lower dimensional space will be identical to the number of classes. We outline the supervised CI in Fig.1.

-
1. Load text data;
 2. Calculate C_i^N for each class C_i ;
 3. Employ (2) to compute reduced space.
-

Fig.1. Outline of concept index.

The time complexity of Step 2 is $O(TW)$, whilst Step 3 will take $O(TKW)$. As a result, the total running time is $O(TW + TKW)$, i.e., $O(TKW)$. Therefore the concept index algorithm scales linearly with the training documents (T).

2.2 DragPushing Strategy

The DragPushing method is to make use of training errors to adjust class centroids so that the misfits or biases can be reduced gradually, and then the training-set error rate can also be reduced gradually.

First, for each category C_i we calculate one summed centroid $C_i^{S,0}$ (using (3)) and one normalized centroid $C_i^{N,0}$ (using (4)). Note that 0 denotes current iteration-step, i.e., the 0th iteration-step.

$$C_i^S = \sum_{d \in C_i} d, \quad (3)$$

$$C_i^N = \frac{C_i^S}{\|C_i^S\|_2}. \quad (4)$$

If one document d labelled as class “A” is classified into class “B”, DragPushing modifies the summed and normalized centroids of class “A” and class “B” by the following formulas:

$$C_{A,l}^{S,o+1} = C_{A,l}^{S,o} + \text{Error Weight} \times d_l \text{ if } d_l > 0, \quad (5)$$

$$C_{A,l}^{N,o+1} = \frac{C_{A,l}^{S,o}}{\|C_{A,l}^{S,o}\|_2} \text{ if } C_{A,l}^{S,o} > 0, \quad (6)$$

$$C_{B,l}^{S,o+1} = [C_{B,l}^{S,o} - \text{Error Weight} \times d_l]_+ \text{ if } d_l > 0, \quad (7)$$

$$C_{B,l}^{N,o+1} = \frac{C_{B,l}^{S,o}}{\|C_{B,l}^{S,o}\|_2} \text{ if } C_{B,l}^{S,o} > 0, \quad (8)$$

where o denotes the o -th iteration-step and l stands for feature index of document vectors and centroid vectors. $[z]_+$ denotes the hinge function which equals to the argument z if $z > 0$ and is zero otherwise, i.e., $[z]_+ = \max\{z, 0\}$. (5) and (6) are called “drag” formulas and those in (7) and (8) are called “push” formulas. The time requirement of DragPushing is $O(MTKW)$ where

-
1. Load training data and parameters;
 2. Calculate C_i^S and C_i^N for each class C_i ;
 3. **For** iter=1 to MaxIteration **Do**
 - 3.1 **For** each document d in training set **Do**
 - 3.1.1 Classify d labeled “A₁” into class “A₂”;
 - 3.1.2 **If** ($A_1 \neq A_2$) **Do**
 - Drag centroid of class A₁ to d using (5) and (6);
 - Push centroid of class A₂ against d using (7) and (8).
-

Fig.2. Outline of DragPushing strategy.

M denotes the max iteration steps. The detailed DragPushing algorithm is given in Fig.2.

3 Refined Concept Index

3.1 Rationale

It is worth noticing that the representative ability (of centroids) is on the basis of classification performance (of centroids classifier). As it is understood, model misfits or biases often degrade the classification accuracy of centroids (classifier) on both training and test documents, and that is to say, they often influence the representative ability of centroids. In order to improve the representative ability of centroids, we employ the DragPushing strategy to refine the centroids.

Let us take a two-class text data as an example. The data distribution is illustrated as Fig.3. Class A spread as grey is elliptically populated; while Class B packed as white is roundly distributed. C_A and C_B are the traditional centroids of Class A and Class B respectively (calculated by (4)). From the view of isolated meaning in a local scale, C_A and C_B are both able to perfectly represent the examples of its own category respectively. However, from the global point of view, the comprehensive representative ability of C_A and C_B is heavily influenced by the example distribution. This point can be easily explained according to Fig.4.

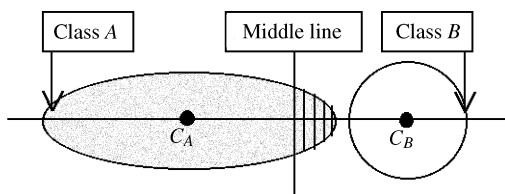


Fig.3. Outline of original centroids of Class A and Class B.

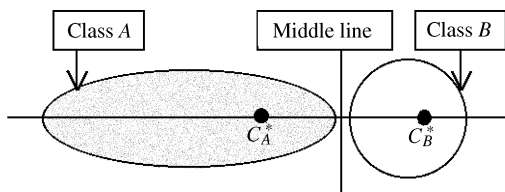


Fig.4. Outline of refined centroids of Class A and Class B.

Obviously, the examples of category A on the right of *middle line* share more similarity with centroid C_B rather than C_A , so they will be misclassified into class B. In this sense sample distribution can affect the representative ability of centroids.

Based on this observation, we use DragPushing strategy to refine the centroids. With the *drag* and *push* operations, C_A and C_B both move to the right gradually.

At the end of the DragPushing strategy (see Fig.4), no example of Class A locates at the right of *middle line* so no example will be misclassified. As a result, the representative ability of centroids is improved.

3.2 Refined Concept Index

As the refined centroids become more representative, we use such centroids instead of the traditional centroids as the axes of reduced space. Instinctively speaking, the refined concept index is expected to produce more efficient and approximate reduced space. The refined concept index can be outlined as Fig.5.

The time complexity of Step 2 is $O(TW)$. According to previous section, DragPushing refinement (Step 3) can be completed in $O(MTKW)$. Step 4 will take $O(TKW)$. As a result, the total running time is $O(TW + MTKW + TKW)$, i.e., $O(MTKW)$. The refined concept index therefore still scales linearly with the training documents (T).

-
1. Load text data;
 2. Calculate C_i^S and C_i^N for each class C_i ;
 3. Refine C_i^N for each class C_i via DragPushing strategy;
 4. Employ (2) to compute reduced space.
-

Fig.5. Outline of refined concept index.

4 Experimental Results

4.1 Data Collections

In our experiment, we use two English corpora (WebKB^①, 20NewsGroup^②) and one Chinese collection (TanCorp^③).

WebKB—The WebKB dataset contains Web pages collected from university computer science departments. There are approximately 8,300 documents in the set and they are divided into seven categories. Among the seven categories, student, faculty, course, and project are the four most populous. The subset we use consists only of these four categories and in total 4,199 documents, called WebKB4.

20NewsGroup—The 20Newsgroup (20NG) dataset contains approximately 20,000 articles evenly divided among 20 Usenet newsgroups. We use a subset consisting of total categories and 19,446 documents.

TanCorp—The TanCorp corpus is collected and processed by Songbo Tan. The corpus is categorized in two hierarchies. The first hierarchy contains 12 large categories and the second hierarchy consists of 60 small classes. The total documents amount to 14,150. This corpus can serve as three categorization datasets: one hierarchical dataset (TanCorpHier) and two flat datasets (TanCorp-12 and TanCorp-60). In our experiment we use TanCorp-12.

① <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/>.

② <http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/wwkb>.

③ <http://lcc.software.ict.ac.cn/~tansongbo/corpus1.php>.

4.2 Performance Measure

To evaluate a text classification system, we use the F1 measure introduced by van Rijsbergen^[16]. This measure combines recall and precision in the following way:

$$\text{Recall} = \frac{\text{number of correct positive predictions}}{\text{number of positive examples}}$$

$$\text{Precision} = \frac{\text{number of correct positive predictions}}{\text{number of positive predictions}}$$

$$\text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{(\text{Recall} + \text{Precision})}$$

For ease of comparison, we summarize the F1 scores over different categories using the Micro- and Macro-averages of F1 scores:

MicroF1 = F1 over categories and documents

MacroF1 = average of within-category F1 values

MicroF1 and MacroF1 emphasize the performance of the system on common and rare categories respectively. Using these averages, we can observe the effect of different kinds of data on a text classification system.

4.3 Experimental Design

We split each dataset into three parts and used two parts for training and the remaining third for testing. We conducted the training-test procedure three times and averaged the performances as the final result. This is so called three-fold cross validation.

We employed TFIDF as input features. The formula for calculating the TFIDF is as follows:

$$W(t, \mathbf{d}) = \frac{tf(t, \mathbf{d}) \times \log(N/n_t + 0.01)}{\sqrt{\sum_{t \in \mathbf{d}} [tf(t, \mathbf{d}) \times \log(N/n_t + 0.01)]^2}},$$

where N is the total number of training documents, and n_t is the number of documents containing the word t .

In our experiments we applied SVMTorch, which uses one-versus-the-rest decomposition and can directly deal with multi-class classification problems. (www.idiap.ch/~bengio/projects/SVMTorch.html).

4.4 Comparison and Analysis

RCI against CI and Information Gain (IG) were compared in our experiments. No other FE algorithms except CI have been tested because most of them are too time-consuming to be suitable for large-scale text data. Extensive experiments of FS algorithms conducted by Yang^[10] have showed that IG always delivers top performance; hereby we regard IG as the representative of FS algorithms.

Table 1. MicroF1 of Different Methods

DataSet (Words)	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
WebKB (64412)	0.8428	0.7716	0.8581	0.9090	0.8926	0.8814
NewsGroup (110023)	0.8908	0.8716	0.9187	0.3583	0.7555	0.8891
TanCorp-12 (72641)	0.9389	0.9216	0.9493	0.8922	0.9324	0.9483

Table 2. MacroF1 of Different Methods

DataSet (Words)	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
WebKB (64412)	0.8144	0.7400	0.8343	0.8976	0.8766	0.8632
NewsGroup (110023)	0.8877	0.8678	0.9174	0.3518	0.7517	0.8876
TanCorp-12 (72641)	0.9058	0.8839	0.9169	0.8404	0.9014	0.9172

Tables 1 and 2 report the MicroF1 and MacroF1 comparison of RCI, CI and IG using SVM. From the two tables, we can observe that RCI achieves a significant performance improvement over the traditional CI: On NewsGroup and TanCorp-12 RCI-based SVM beats CI-based SVM by about 2%, and on WebKB by 7%. Furthermore, RCI-based SVM yields competitive results against IG-based SVM on NewsGroup and TanCorp-12. On the three corpora, RCI is only two percent points lower than All-words-based SVM.

Tables 3–5 present the computational time of feature reduction, Training and Prediction in seconds. The comparison is conducted on a personal computer with single Intel Pentium 3.0G (MHz) CPU and 512MB memories. As to feature reduction, RCI consumes a little more CPU time than CI and roughly equivalent at the second place with IG (10000).

With regard to training and testing time, RCI-based SVM gains obvious advantages over CI or IG-based SVM. On training phase, RCI-based SVM is nearly two times faster than CI-based SVM. As a result, RCI transforms a more friendly reduced space for SVM than CI. At the same time, the classification speed of RCI-based SVM is much higher than CI-based SVM. This phenomenon can be explained as: RCI-based SVM trains much less support vectors than CI-based SVM (see Table 6), so the prediction speed is much faster.

Compared with IG or All-words-based SVM, RCI-based SVM has absolute advantages in training and testing stages. During training phase, RCI-based SVM is at least ten times faster than IG or All-words-based SVM; with regards to testing phase, RCI-based SVM performs about four times faster than IG or All-words-based SVM. As a result, the efficiency of RCI-based SVM is very notable.

As far as the support vectors are concerned, RCI-based SVM produces only 1,474 support vectors in total, a little more than IG(100)-based SVM (1334). For IG method, with the increase of feature number, IG-based SVM produces more and more support vectors, which accounts for the fact that with the increase of feature

number, IG-based SVM gets slower and slower in the training process. In summary, these experiments have showed that RCI provides an alternate choice for fast and yet efficient text features reduction method.

Table 3. Feature Reduction Time in Seconds

DataSet (Words)	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
WebKB (64412)	30.359	26.542	0.000	1.417	7.370	66.536
NewsGroup (110023)	100.016	76.760	0.000	5.063	13.453	97.599
TanCorp-12 (72641)	72.604	55.021	0.000	2.380	6.375	48.182

Table 4. Training Time in Seconds

DataSet (Words)	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
WebKB (64412)	0.958	1.776	56.953	11.333	11.937	28.516
NewsGroup (110023)	38.245	61.115	1100.330	820.020	467.135	556.719
TanCorp-12 (72641)	9.073	17.104	701.333	136.902	82.157	375.386

Table 5. Test Time in Seconds

DataSet (Words)	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
WebKB (64412)	0.198	0.317	32.808	0.818	3.922	15.432
NewsGroup (110023)	7.756	10.818	505.167	31.453	40.083	249.182
TanCorp-12 (72641)	1.766	2.687	427.307	5.865	41.510	252.823

Table 6. Support Vectors of SVM on WebKB

Categories	RCI	CI	All	IG(100)	IG(1000)	IG(10000)
1	187	407	1154	135	271	646
2	612	906	1821	450	534	1087
3	226	419	1321	289	349	792
4	449	925	1742	460	493	984
Total	1474	2657	6038	1334	1647	3059

5 Conclusion

In this work we have developed and applied the "DragPushing" Strategy to refine the centroids and used the refined centroids for concept index. An extensive experimental evaluation of the refined concept index (RCI) method is conducted on two English collections and one Chinese corpus using SVM classifier. The experimental results demonstrated that RCI-based SVM yields much better performance than CI-based SVM. This result gives evidences of validity of centroids refinement for CI. It is also proved that RCI-based SVM is faster than CI-based SVM during training and classification phases partially due to the fact that RCI-based SVM produces much less support vectors than CI-based SVM.

References

- [1] Yang Y, Lin X. A re-examination of text categorization methods. In *The 22nd ACM Int. Conf. Research and Development in Information Retrieval*, Berkeley, 1999, pp.42–49.
- [2] E Han, G Karypis. Centroid-based document classification analysis & experimental result. In *The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, France, 2000, pp.424–431.
- [3] D D Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *The 10th European Conference on Machine Learning*, Springer Verlag, Heidelberg, DE, 1998, pp.4–15.
- [4] Andrew McCallum, Kamal Nigam. A comparison of event models for Naive Bayes text classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization [C]*, Menlo Park, CA: AAAI Press, 1998, pp.41–48.
- [5] P P T M van Mun. Text classification in information retrieval using Winnow. <http://citeseer.csail.mit.edu/133034.html>
- [6] T Joachims. Text categorization with support vector machines: Learning with many relevant features. In *The 10th European Conference on Machine Learning*, Springer Verlag, Heidelberg, DE, 1998, pp.137–142.
- [7] G Salton, M J McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.
- [8] Inderjit S Dhillon *et al.* A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 2003, 3: 1265–1287.
- [9] Liu H, Motoda H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic, Norwell, MA, USA, 1998.
- [10] Yang Y, Pedersen J O. A comparative study on feature selection in text categorization. In *Proc. the 14th Int. Conf. Machine Learning Table of Contents*, San Francisco, CA, USA, 1997, pp.412–420
- [11] Jolliffe I T. *Principal Component Analysis*. New York: Springer Verlag, 1986.
- [12] Martinez A M, Kak A C. PCA versus LDA. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2001, 23(2): 228–233.
- [13] Li Haifeng, Jiang Tao, Zhang K. Efficient and robust feature extraction by maximum margin criterion. In *Proceedings of the Advances in Neural Information Processing Systems 16*, (Vancouver, Canada), MIT Press, 2004, pp.97–104.
- [14] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290: 2323–2326.
- [15] George Karypis, EuiHong (Sam) Han. Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In *The 9th ACM International Conference on Information and Knowledge Management*, ACM Press, New York, US, 2000, pp.12–19.
- [16] Rijsbergen C. *Information Retrieval*. London: Butterworths, 1979.
- [17] Malhi A, Gao R X. PCA-based feature selection scheme for machine defect classification. *IEEE Transactions on Instrumentation and Measurement*, 2004, 53(6): 1517–1525.
- [18] Ran Gilad-Bachrach, Amir Navot, Tishby N. Margin based feature selection — Theory and algorithms. *The 21st Int. Conf. Machine Learning*. Banff, Alberta, Canada. 2004, 43.
- [19] Douglas Hardin, Ioannis Tsamardinos, Aliferis C F. A theoretical characterization of linear SVM-based feature selection. In *The Twenty-First International Conference on Machine Learning*, Banff, Alberta, Canada. 2004, p.48.
- [20] Songbo Tan, Xue-Qi Cheng, Moustafa M Ghanem *et al.* A novel refinement approach for text categorization. In *The 14th ACM Int. Conf. Information and Knowledge Management Table of Contents*, Bremen, Germany, 2005, pp.469–476.