Journal of Computational Information Systems 8: 20 (2012) 8569–8579
Available at http://www.Jofcis.com

Refined Semantic Kernel Matching Pursuit for Fast Text Classification ⋆

Qing ZHANG*, Zhiping ZHANG, Li WANG

Institute of Scientific and Technical Information of China, Beijing 100038, China

Abstract

Kernel-based learning algorithms have been proven successful and powerful in many different tasks. However, directly applying them to text classification (TC) field will suffer inherent semantic ambiguity and prohibitive computational costs, which obstruct their practical use in large scale and real time applications with fast testing requirement. To solve this problem, this paper proposes a refined semantic kernel matching pursuit (KMP) approach for fast TC. This approach firstly introduces latent semantic kernel as a new type of dictionary function to KMP, which can deal with high dimensionality, sparsity and ambiguity suffered in text. Moreover, taking the practical issue of data distribution drift with time changing into account, we further propose a method for constructing refined semantic dictionary via concept factorization, which can maintain the updated representative samples in the limited storage to be utilized for various model updating schemes in future. The experimental results demonstrate the proposed method can significantly improve the computational efficiency in predicating phase while preserving considerable performance.

Keywords: Kernel Matching Pursuit; Fast Text Classification; Refined Semantic Dictionary; Concept Factorization

1 Introduction

The widespread and increasing availability of massive textual data stimulates the development of text classification (TC) field, which aims to automatically assign unlabeled documents to a predefined one or more classes according to its contents. The demands for TC are promoted by diverse real applications with various digitized contents, such as web page organizing, personalized news recommending, topic tracking and spam email filtering. However, the challenging issues of TC are characterized by its inherent high dimensionality and the inevitable semantic ambiguity. To solve those problems, in machine learning and information retrieval communities, related issues regarding to document representation, dimensionality reduction and model construction have been comprehensively studied [1]. Specifically, this paper mainly focuses on kernel-based TC problem

⋆Project funded by the National Key Technologies R&D Program of China in 12th Five-Year Plan (No. 2011BAH10B05).

*Corresponding author.
Email address: zhangqing@sse.buaa.edu.cn (Qing ZHANG).

October 15, 2012

to address the emerging challenging issue resulting from fast classification requirement in large scale and real time applications.

In recent decades, a number of powerful kernel-based learning machines [2] have been proposed and achieved competitive performance in many different learning tasks. In particular, as our previous works review [3], kernel methods for text have attracted much attention [4-7]. However, some TC tasks based on kernel methods are still not practical for the scalability demands [8, 9]. In fact, the problem of scalability deficiency for the kernel based approaches is inherent due to the huge operations on kernel matrix [10, 11] and dense final optimal model obtained [12-15], which all largely depend on the whole training samples. More specifically, for the dense final model issues which have direct relations to our problem, some efforts directly approximate the final model in kernel induced space, such as Burges et al. [12] and Zhang et al. [13]. Another way to solve this issue is to control the sparseness of the classifier by adding a constraint to the original optimization problem, e.g., Wu et al. [14]. In addition, Diethe et al. [15], propose an explicit kernel induced subspace mapping approach. Later, Zhang et al. [3] take advantage of the inherent modularity in kernel-based learning machines and the availability of the explicit kernel subspace approximation [15] to propose a semantic kernel-based framework for efficient TC.

In this paper, we proposes a refined semantic kernel matching pursuit approach for fast text classification, which extends kernel matching pursuit originated from single processing field into text classification problem. This approach firstly introduces latent semantic kernel as a new type of dictionary function to KMP, which can deal with high dimensionality, sparsity and the inevitable existence of polysemy and synonym suffered by textual data. Moreover, to address the compressed storage of current data distribution, we propose a refined semantic dictionary method for the further extension of original KMP following the introduction of LSI to KMP. The general idea in this paper is inspired by [3], but the proposed approach is different from [3] in two aspects. Firstly, the sparse model construction for text classifier in our approach only refers to reconstruction error minimum rule instead of complex discriminant function used in [3]. So this approach is relative simple to well understand and easy to implement. Secondly, we further address the issue of semantic dictionary shrinkage. The maintained current data distribution in less storage can be utilized for other various updating schemes dealing with distribution drift with time changing. Detail updating scheme is another active research filed [16] which is beyond our discussion in this paper.

This paper is organized as follows. In section 2 we will review the related background used in this paper. In section 3 we will introduce our proposed approach, followed by the experimental evaluations in section 4. The conclusions are given in section 5.

2 Preliminaries

We review briefly kernel methods first, and then Kernel Matching Pursuit. The aim of this section is to show the obstacle of kernel methods for large scales testing applications in TC and then give why the introduced KMP is an appropriate basic model for further being extended in our proposed solution to solve the problem of inefficient testing.

2.1 Brief review of kernel method

As our review in [3], Kernel Method can be seen as a state-of-the-art learning framework for all kinds of problems and it has been successfully introduced into text classification field pioneered by [4]. The main idea behind this approach is the kernel trick, which employs a kernel function to map the data from the original input space into a kernel-induced space implicitly. Then, the problem in kernel space can be reformulated into dot product form substituted by Mercer kernels [2] and finally standard algorithms in input space are performed to solve the kernel induced learning problem.

The general framework of kernel approach [2] is featured with the modularity, which enable various learning algorithms to obtain the solution with enhanced ability. For example, KPCA is the kernel version of PCA approach with enhanced excellent performance of nonlinear feature extraction via diverse kernel functions implicitly.

Given a training set $\{x_1, x_2, \ldots, x_L\}$, a mapping ϕ and a kernel function $k(x_i, x_j)$, all similarity information between input patterns in kernel feature space is entirely preserved in kernel matrix (also called Gram matrix),

$$\mathbf{K} = \big(k(x_i, x_j)\big)_{i,j=1,L} = \big(\langle \phi(x_i), \phi(x_j) \rangle\big)_{i,j=1,L}. \tag{1}$$

Usually, kernel-based algorithms can seek a linear function solution in feature space [2], as follows

$$f(x) = w'\phi(x) = \sum_{i=1}^{L} \langle \alpha_i \phi(x_i), \phi(x) \rangle = \sum_{i=1}^{L} \alpha_i k(x_i, x) \tag{2}$$

As shown above, the main drawback of this kernel-based TC method is usually lack of sparsity in (2), which is linear proportional to all training samples. Unfortunately, L in (2) is usually very large and even nearly close to all training examples (such as in KFDA). It will seriously undermine the classification efficiency on large scale text corpus in predicting phase, especially in real time applications, because all L samples are needed to compute the final predictive model for every new testing candidate.

2.2 Kernel matching pursuit

To overcome the problem mentioned in 2.1, Kernel Matching Pursuit [17] is employed in this paper as the foundation of the proposed approach. Our mind behind of speeding up kernel-based TC is to make the final classification model in (2) as sparse as possible so as to reduce the computational costs in testing phrase. More particularly, the ultimate goal is that the so-called sparse model in (2) makes the L in (2) far smaller than the number of training examples. The motivation of the introduction of KMP is that it has the same model form shown in (2) but its sparsity can be well controlled to achieve much sparser solution of Equation (2) especially in large scale training set as following described.

KMP [17] is the extension of Basic Matching Pursuit to machine learning field from signal processing community. Given l noisy observations $\{y_1, \ldots, y_l\}$ of a target function f in a Hilbert space at points $\{x_1, \ldots, x_l\}$, we are interested in sparse approximation of f by the linear expansion of finite dictionary functions, $D = \{d_1, d_2, \ldots, d_m\}$, (where $d_m(\cdot) = k(\cdot, x_m)$). The main idea of

this method is to append functions from a redundant dictionary greedily according some certain loss criterion, e.g., the pre-fitting version of KMP using (3), incorporated into our proposed approach,

$$\left(g_{n+1}, \alpha_{1\dots n+1}^{n+1}\right) = \underset{(g\in D, \alpha_{1\dots n+1}\in\mathbb{R}^{n+1})}{\arg\min} \left\| \left(\sum_{k=1}^{n} \alpha_k \vec{g_k}\right) + (\alpha_{n+1}\vec{g} - \vec{y}) \right\|^2, \tag{3}$$

where $\vec{y} = \left[y_1 \dots, y_l\right]^{\mathsf{T}}$ as an initial value of the residue $\vec{R}$, g_{n+1} is the selected function from D at the $n+1$ iteration, $\vec{g}$ is the dictionary vector with the elements of $g_{n+1}(x_i)$, $1 \leqq i \leqq l$ and $\alpha_{1\dots n+1}^{n+1}$ are the updated optimal weights for the combined $n+1$ bases (selected functions), which contains all updated $n+1$ weights at the $n+1$ iteration. In particularly, $(g_{n+1}, \alpha_{1\dots n+1}^{n+1})$ are obtained by projecting the target and all dictionary vectors into $B_n = \mathrm{span}(\vec{g_1}, \dots, \vec{g_n})$ and its orthogonal complement space $B_n^{\perp}$. Then the algorithm searches the optimal basis with updated weights. The procedure can be summarized as following main steps (See details in [17]),

(1) Searching function in D which is the most collinear with the residue $\vec{R}$ in $B_n^{\perp}$,
$r_n \leftarrow \underset{k=1\dots m}{\arg\min} \left| \left\langle D_{B^{\perp}}(\cdot, k), \vec{R} \right\rangle \Big/ \left\| D_{B^{\perp}}(\cdot, k) \right\| \right|$.

(2) Computing the new basis optimal weight, $\alpha_n \leftarrow \langle D_{B^{\perp}}(\cdot, r_n), \vec{R}\rangle / \|D_{B^{\perp}}(\cdot, r_n)\|^2$.

(3) Updating the residue, $\vec{R} = \vec{R} - \alpha_n D_{B^{\perp}}(\cdot, r_n)$.

(4) Updating all previous weights, $(\alpha_1, \dots, \alpha_{n-1}) \leftarrow (\alpha_1, \dots, \alpha_{n-1}) - \alpha_n D_B(\cdot, r_n)'$.

(5) Redecomposing the dictionary vectors into B_n and $B_n^{\perp}$ taking the influence of new acquired basis into account. see [17].

After above procedures, the sparse version of Equation (2) is obtained. However, directly applying KMP to TC problem is not desirable due to its deficiency of semantic information for high dimensional textual data. Meanwhile, the volume of redundant dictionary is usually very large containing all training examples in large scale applications. To solve these problems, our proposed approach is presented in following section to show how to adapt KMP to TC problem with refined semantic dictionary.

3 Refined Semantic Kernel Matching Pursuit for Fast Text Classification

In this section, we present our proposed approach for fast TC with the consideration of further model updating for data distribution drift with time changing. As shown in our proposed Algorithm 1, we adapt the original KMP to TC problem from three aspects. Firstly, the VSM representation of textual data is employed and then we introduce LSI kernel function as a new type dictionary for textual data. Finally, we further proposed a method for constructing a compact representation of the current data distribution, i.e., refined semantic dictionary.

Algorithm 1 Refined Semantic Kernel Matching Pursuit for Fast Text Classification

Input: training data $\{x_1, \ldots, x_l\}$ with class observations $\{y_1, \ldots, y_l\}$

Output: sparse kernel-based classification model for TC

(1) Text preprocessing including stemming, removing stop words.
(2) Run 3.1: Vector Space Mapping.
(3) Run 3.2: Semantic Kernel Dictionary Building.
(4) Run 3.3: Refined Semantic Dictionary Constructing (For Further Model Updating).
(5) Run 2.2: KMP with pre-fitting in Refined Semantic Dictionary Space.
(6) Obtain the sparse model for TC.

3.1 Vector space mapping

The majority of kernel-based algorithms (e.g., SVM) are originally designed for the numerical vector-based samples in input space. Therefore, vector space model (VSM) [5] representation for textual data is a prerequisite. In VSM representation, each document d_i in corpus is represented as a bag of words (BOW) in N dimensional vector space through the irreversible mapping,

$$\phi : d_i \mapsto \phi(d_i) = (tf(t_1, d_i), tf(t_2, d_i), \ldots, tf(t_N, d_i))' \in \mathbb{R}^N, \tag{4}$$

where $tf(t_i, d_i)$ is the frequency of the term t_i in document d_i and N is the unique terms extracted from the corpus. As a result, the term-document matrix shown in (4) containing L documents in corpus can be acquired,

$$\mathbf{D}_{\text{VSM}} = \begin{pmatrix} tf(t_1, d_1) & \cdots & tf(t_1, d_L) \\ \vdots & \ddots & \vdots \\ tf(t_N, d_1) & \cdots & tf(t_N, d_L) \end{pmatrix}. \tag{5}$$

3.2 Semantic kernel dictionary building

As seen in 3.1, the dimensionality using the representation of VSM for each document is usually much higher than the true occurrence of words in each document. These characteristics of high dimensionality and sparsity in massive dataset not only lead to huge computational costs for learning algorithms but also aggravate the inherent problem of semantic ambiguity. Thus, we need to seek a suitable dictionary function for KMP to overcome those problems. In order to solve ambiguity in text, various methods have been developed for the extraction of semantic information in large scale corpus through textual contents [3].

In this paper, LSI approach is used to build semantic kernel as described in Cristianini etc al. [6] in order to deal with the semantic deficiency problem for KMP. LSI is a feature reduction approach which maps the document in VSM into a semantic subspace spanned by several concepts using Singular Value Decomposition (SVD) in an unsupervised way [3]. In that low-dimensional subspace, the similarity between documents can reflect the semantic structures based on common concepts, which takes words co-occurrence information into consideration. More specifically, the term document matrix derived from (5) is decomposed using SVD,

$$\mathbf{D}_{\text{VSM}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}', \tag{6}$$

where the columns of matrix $\mathbf{U}$ and $\mathbf{V}$ are the eigenvectors of $\mathbf{D}_{\mathrm{VSM}}\mathbf{D}'_{\mathrm{VSM}}$ and $\mathbf{D}'_{\mathrm{VSM}}\mathbf{D}_{\mathrm{VSM}}$ respectively, $\mathbf{\Sigma}$ is a diagonal matrix with nonnegative real diagonal singular values sorted in decreasing order.

The key to building LSI kernel is to find the matrix $\mathbf{P}$ defined by the mapping $\phi' : d \mapsto \phi'(d)$, i.e., $\phi'(d) = \mathbf{P}\phi(d)$. For LSI case, the concept subspace is spanned by the first k columns of $\mathbf{U}$, which form the matrix $\mathbf{P}$,

$$\mathbf{P} = \mathbf{U_k}' = (u_{1,}u_{2,\cdots,}u_k)' \tag{7}$$

Hence the LSI kernel mapping is $\phi' : d \mapsto \phi'(d) = \mathbf{U_k}'\phi(d)$ and the kernel matrix is

$$\mathbf{K} = (\phi(d_i)'\mathbf{U_k}\mathbf{U_k}'\phi(d_j))_{i,j=1,L} = \mathbf{D}_{VSM}{}'\mathbf{U_k}\mathbf{U_k}'\mathbf{D}_{VSM} \tag{8}$$

Consequently, semantic dictionary functions $d_m(.) = k(., x_m) = \phi(.)'\mathbf{U_k}\mathbf{U_k}'\phi(x_m)$ in $D = \{d_1, d_2, \ldots, d_m\}$ are acquired for KMP and $\mathbf{K}$ can be seen a semantic dictionary with function values of each sample

$$\begin{bmatrix} d_{x_1}(x_1) & \cdots & d_{x_i}(x_1) \\ \vdots & \ddots & \vdots \\ d_{x_1}(x_l) & \cdots & d_{x_i}(x_l) \end{bmatrix}. \tag{9}$$

3.3 Refined semantic dictionary constructing

Moreover, another problem encountered in real applications is the instability of data distribution with time changing. So, the model updating to adapt to distribution drift is inevitable. Thus, we further propose a method for constructing refined semantic dictionary via concept factorization. The obtained refined dictionary can be seen as a compressed version of the current data distribution, maintaining the updated representative samples in the limited storage to be utilized for various model updating schemes in future. Detail updating scheme [16] is another active research filed which is beyond our discussion in this paper.

Our idea for refining semantic dictionary is simple and intuitive, which can be seen as the product of joint judging from the results of unsupervised learning and its prior label information. For example, if we want to judge a student whether an excellent one, we must examine him or her from two aspects. One is the previous performance reflected by its prior label, i.e., excellent or not excellent in last period. The other is the present performance in the current period, i.e., unsupervised partition with all other new or old students according to the current academic results or related criteria. For our problem in this paper, we want to seek a compressed representative distribution for the current data. We incorporate concept factorization [18] into our approach for seeking the present predicted label, due to its capability of dealing with the data containing negative values and working in kernel space, which is superior to non-negative matrix factorization (NMF).

Given a $m \times n$ data matrix $\mathbf{X} = [X_1, X_2, \ldots, X_n]$ and k concepts, concept factorization models each concept R_c as a linear combination of each data point $R_c = \sum_{i=1}^{n} w_{ic}X_i$, and each data point as a linear combination of concepts $X_i \approx \sum_{c=1}^{k} v_{ic}R_c$. w_{ic} is a non-negative association weight

Algorithm 2 Refined Semantic Dictionary Constructing

Input: training data $\{x_1, \ldots, x_l\}$ with class observations $\{y_1, \ldots, y_l\}$, k indicating the number of concepts, max indicating maximal iterations, kernel matrix $\mathbf{K}$, compressed threshold t

Output: refined data set $\{(x_i, y_i)_{\text{refined}}\}$

(1) Randomly initialize $\mathbf{V}$ and $\mathbf{W}$.

(2) Fixing $\mathbf{V} = [v_{ic}]$, update $\mathbf{W}$ using $w_{ij} \leftarrow w_{ij} \frac{(\mathbf{KV})_{ij}}{(\mathbf{KWV}^T\mathbf{V})_{ij}}$ to decrease J (11).

(3) Fixing $\mathbf{W} = [W_{ic}]$, update $\mathbf{V}$ using $v_{ij} \leftarrow v_{ij} \frac{(\mathbf{KW})_{ij}}{(\mathbf{VW}^T\mathbf{KW})_{ij}}$ to decrease J (11).

(4) Normalize $\mathbf{W}$ using $\mathbf{V} \leftarrow \mathbf{V}[\text{diag}(\mathbf{W}^T\mathbf{KW})]^{1/2}$ and $\mathbf{W} \leftarrow \mathbf{W}[\text{diag}(\mathbf{W}^T\mathbf{KW})]^{-1/2}$.

(5) Repeat step 2, 3, 4, until the result converges or achieve the maximal iterations max.

(6) Ranking $\{(x_i, y_i)\}$ according each column of $\mathbf{V}$ in descending order, then obtain $\mathbf{V}_{\text{rank}}$ with elements x_i.

(7) Extract top N rows of $\mathbf{V}_{\text{rank}}$ with the $t \times 100$ percentage of total samples in each class, then obtain $\mathbf{V}^t_{\text{rank}}$.

(8) Computing the proportion of label of x_i in each column of $\mathbf{V}^t_{\text{rank}}$, then assign the label with largest proportion to each column of $\mathbf{V}^t_{\text{rank}}$ as the concept label (each column represents a concept).

(9) For each sample x_i in $\mathbf{V}^t_{\text{rank}}$, if its prior label y_i equals to the concept label (current predicted label) of that column of $\mathbf{V}^t_{\text{rank}}$, then keep it as a final refined data points $(x_i, y_i)_{\text{refined}}$, else drop it out.

indicating to which degree data point i is related to concept c and v_{ic} is a non-negative number showing the projection value of X_i onto the base (concept center) R_c. Consequently, we have

$$X_i \approx \sum_{c=1}^{k} v_{ic} R_c = \sum_{c=1}^{k} v_{ic} \sum_{j=1}^{n} X_j w_{jc}. \tag{10}$$

In order to seek w_{ic} and v_{ic} used for current label prediction, we minimize the squared error in (11) between $\mathbf{X}$ and its approximation using iterative algorithm [18]

$$J = \frac{1}{2}\left\|\mathbf{X} - \mathbf{XWV}^T\right\|^2 = \frac{1}{2}(\text{tr}(\mathbf{K}) - 2\text{tr}(\mathbf{W}^T\mathbf{KV}) + \text{tr}(\mathbf{W}^T\mathbf{KWV}^T\mathbf{V})) \tag{11}$$

where $\mathbf{W} = [w_{ic}]$ is $n \times k$ association matrix, $\mathbf{V} = [v_{ic}]$ is the $n \times k$ projection matrix and $\mathbf{K} = \mathbf{X}^T\mathbf{X}$. After we acquire v_{ic} for each sample, the appropriateness of being refined can be well judged. More particularly, to achieve our goal, it can be summarized that

(1) We firstly employ concept factorization with the number of concepts the same as that of the labeled class to compute the present predicted label with possibility for each sample.

(2) Then, we select the top-N samples with the highest possibility for each class and make comparison with its prior label information.

(3) If the current predicted label obtained in unsupervised way is identical to its prior label, then we keep it for the representative of that class otherwise drop it out from the dataset.

(4) Finally, the most representative samples for the current distribution (in each class) are preserved only in small scale with time changing.

The formal expression of our presented idea can be seen in Algorithm 2 in details.

4 Experiments

In order to evaluate the performance of the proposed method for fast TC, several tests are performed on 20-Newsgroups (20NG) dataset [19]. Because the determining factor of the computational efficiency for TC is the sparsity of the classifier model, our metric [3] used to examine our proposed approach is the number of basis in final learned model with the volume of semantic dictionary rather than the relative time costs. Nevertheless, if our model is fast but with low performance in classification accuracy, it is also useless. Thus, the balance of speediness and accuracy is also considered to examine.

Table 1: Six binary classification problem settings on 20-newsgroups dataset

Lab-ID	Class-P	Class-N	N-Train	N-Test	D-VSM
ID-1	talk.politics.guns	talk.politics.mideast	1110	740	12825
ID-2	talk.politics.guns	talk.politics.misc	1011	674	10825
ID-3	talk.politics.mideast	talk.politics.misc	1029	686	12539
ID-4	rec.autos	rec.motorcycles	1192	794	9573
ID-5	com.sys.ibm.pc.hardwar	com.sys.mac.hardwar	1168	777	8793
ID-6	sci.electronics	sci.space	1184	787	10797

4.1 Experimental setup

In our experiments, the proposed approach with refined semantic dictionary (called R_SKMP prefitting) and its simplified version with original redundant dictionary (called SKMP_prefitting) are examined in comparison with several baseline methods, i.e., LSI-kernel (LSI-SVM), KNN in LSI feature space (LSI-KNN) and the method in [3] (SKF-ETC). As the experiments setting in [3] we employed, to make the task more challenging, we select the most similar sub-topics in the lowest level in 20NG as our six binary classification problems listed in Table 1. The approximate 5 fold cross validation scheme is adopted as in [3]. After some preprocessing procedures including stop words filtering and stemming, VSM model is created (5). The average dimensionalities of VSM generated (D-VSM) are also shown in Table 1. It is noted that KNN is implemented in the nearest neighbor way and the LSI space holds 100 dimensions. We set the number of concepts $k = 2$ with the max iteration $max = 100$ and compressed threshold t optimal chosen by cross validation between 0.3 and 1.1. The optimal parameters for SVM with 5 fold cross validation are used for SVM rather than the default choosing in [3].

4.2 Experimental results and discussion

The experimental (best average) results are shown in Table 2 for the proposed method (R_SKMP prefitting and SKMP_prefitting) with baseline methods, LSI-SVM, LSI-KNN and SKF-ETC. Since the experiments implemented for LSI-KNN and SKF-ETC are same as [3] with identical setting, results of the two methods are directly taken from [3], shown in Table 2. Particularly, in Table 2, 'Acc.' represents the accuracy of classification, 'N-D' represents the number of elements in semantic dictionary and 'N-B' represents the number of basis in final classifier model (for SVM,

represents number of support vectors and for KNN, represents the number of samples required to compute in testing phrase). 'N-B' reflects the general computational costs consumed for a classifier model in testing phrase. The lesser in 'N-B' means the faster for TC. 'N-D' reflects the capability of reduction in semantic dictionary for the proposed method.

Table 2: Results on six binary classifications for evaluating the proposed approach with baseline methods

Lab	R_SKMP_prefitting			SKMP_prefitting			LSI-SVM		LSI-KNN		SKF-ETC		
-ID	Acc.	N-D	N-B	Acc.	N-D	N-B	Acc.	N-B	Acc.	N-B	Acc.	N-D	N-B
ID-1	0.9203	406	40	0.9151	1110	40	0.9462	400	0.9481	1110	0.9108	1110	28
ID-2	0.8300	370	37	0.8190	1011	37	0.8481	527	0.8234	1011	0.8026	1011	17
ID-3	0.8828	466	38	0.8773	1029	19	0.8886	436	0.9131	1029	0.8772	1029	19
ID-4	0.8854	354	38	0.8809	1192	38	0.9179	432	0.8239	1192	0.8836	1192	25
ID-5	0.7840	504	39	0.7789	1168	40	0.8103	722	0.7127	1168	0.7863	1168	31
ID-6	0.8966	568	36	0.8968	1184	40	0.9479	356	0.8694	1184	0.8996	1184	28
Avg.	0.8665	445	38	0.8613	1116	36	0.8932	479	0.8484	1116	0.8600	1116	25

As shown in Table 2, we find that our methods (R_SKMP_prefitting and SKMP_prefitting) can significantly decrease the number of the bases in final solution. Compared with LSI-SVM and LSI-KNN in average performance, our R_SKMP_prefitting and SKMP_prefitting only hold 38 and 36 bases ('N-B' in Table 2) in final model respectively while LSI-SVM and LSI-KNN hold 479 and 1116 bases respectively. It means that when testing a new sample, using our method only needs computing with 38 or 36 bases in final model. However, using LSI-SVM and LSI-KNN needs 479 and 1116 bases respectively. So the time saving in our method is obvious especially in large scale testing tasks with fast requirement. Moreover, our method is also competitive to SKF-ETC with additional merits of idea simple and compressed semantic dictionary. In addition to speediness, the proposed method also maintains considerable classification accuracy. As shown in Table 2, the average accuracy is only about 3% lower than LSI-SVM but with about 92% improvements of average sparsity than it and the case for LSI-KNN is more obvious.

To further investigate the performance of R_SKMP_prefitting with incremental bases, we find that the proposed refining dictionary method is effective according to Fig. 1 to Fig. 6 and can well fit the curve of SKMP_prefitting, holding only about 40% of total data shown in Table 2 without loss of performance. Thus, the refined semantic dictionary maintains a compact data distribution for generating the current classification model and can be utilized with the saving of 60% storage (445 versus 1116 shown in Table 2) for further model updating schemes with time changing.

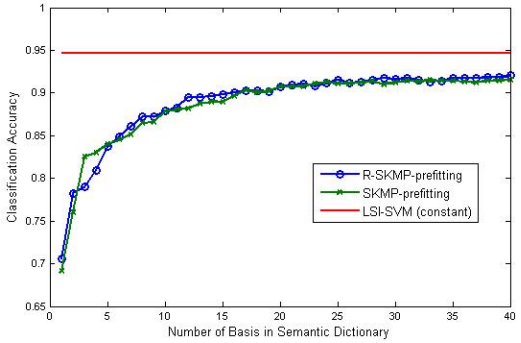

Fig. 1: Result of ID-1 with incremental bases

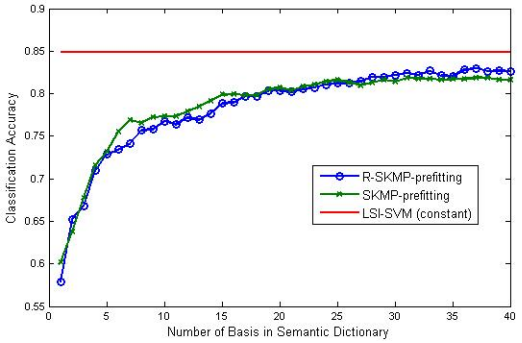

Fig. 2: Result of ID-2 with incremental bases

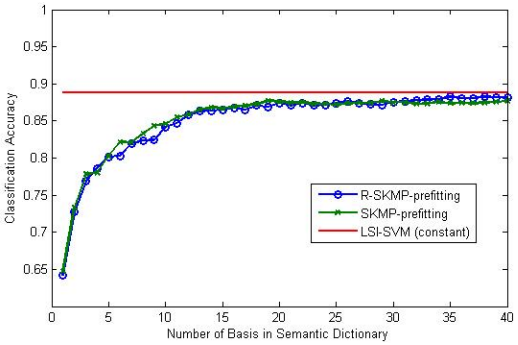

Fig. 3: Result of ID-3 with incremental bases

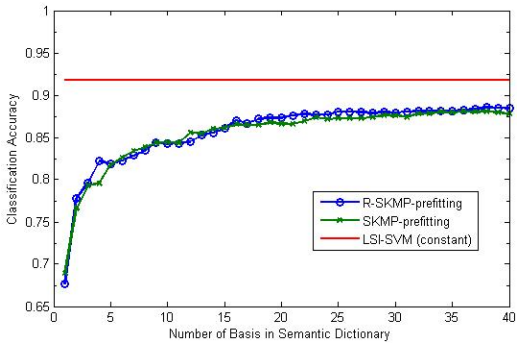

Fig. 4: Result of ID-4 with incremental bases

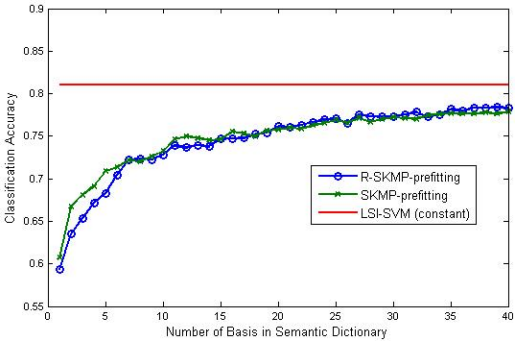

Fig. 5: Result of ID-5 with incremental bases

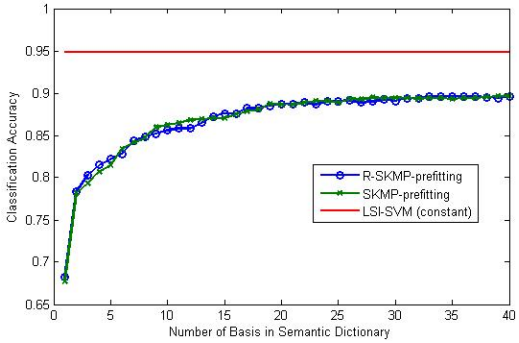

Fig. 6: Result of ID-6 with incremental bases

5 Conclusions

To solve the scalability problem of kernel-based TC with fast predicting requirement in large scale and real time applications [8, 9], we propose a novel approach, called Refined Semantic Kernel Matching Pursuit. The propose approach extends original KMP from two aspects for the novel efficient TC task. Firstly, it introduces LSI kernel to augment the dictionary in KMP for dealing with the weakness suffered in textual data. Secondly, it further takes the data distribution drift with time changing into account. The refined semantic dictionary obtained by our approach can maintain a compact data distribution in much smaller scale than the full redundant dictionary without performance loss, which can be utilized for further model updating scheme efficiently. Detail model updating scheme with time changing is another active research field, which is beyond our discussion in this paper and will be exploited in our further work.

References

[1] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34 (1): 1 – 47, 2002.

[2] J. S. Taylor, N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, 2004.

[3] Q. Zhang, J. W. Li, Z. P. Zhang. Efficient Semantic Kernel-Based Text Classification Using Matching Pursuit KFDA. In Proceedings of ICONIP, 382 – 390, 2011.

[4] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Proceedings of ECML, 137 – 142, 1998.

[5] G. Salton, A. Wong, C. Yang. A Vector Space Model for Automatic Indexing. Commun. ACM (CACM), 18 (11): 613 – 620, 1975.

[6] N. Cristianini, S. Taylor, H. Lodhi. Latent Semantic Kernels. Journal of Intelligent Information System, 18 (2 – 3): 127 – 152, 2002.

[7] G. Tsatsaronis, I. Varlamis, M. Vazirgiannis. Text Relatedness Based on a Word Thesaurus. Journal of Artificial Intelligence Research. 37: 1 – 39, 2010.

[8] H. Wang, Y. Chen, Y. Dai. A Soft Real-Time Web News Classification System with Double Control Loops. In Proceedings of WAIM, 81 – 90, 2005.

[9] E. Miltsakaki, A. Troutt. Real-time Web Text Classification and Analysis of Reading Difficulty. In Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications at ACL, 89 – 97, 2008.

[10] A. J. Smola, B. Schölkopf. Sparse Greedy Matrix Approximation for Machine Learning. In Proceedings of ICML, 911 – 918, 2000.

[11] S. Fine, K. Scheinberg. Efficient SVM Training Using Low-Rank Kernel Representations. Journal of Machine Learning Research. 2: 243 – 264, 2001.

[12] C. J. C. Burges. Simplified support vector decision rules. In Proceedings of ICML, 71 – 77, 1996.

[13] Q. Zhang, J. W. Li. Constructing Sparse KFDA Using Pre-image Reconstruction. In Proceedings of ICONIP, 658 – 667, 2010.

[14] M. Wu, B. Schölkopf, G. Bakir. Building Sparse Large Margin Classifiers. In Proceedings of ICML, 996 – 1003, 2005.

[15] T. Diethe, Z. Hussain, D. R. Hardoon, et al. Matching Pursuit Kernel Fisher Discriminant Analysis. In Proceedings of AISTATS, 121 – 128, 2009.

[16] G. João, M. Pedro, C. Gladys, R. Pedro. Learning with drift detection. In Proceedings of the 17th Brazilian Symposium on Artificial Intelligence, 286 – 295, 2004.

[17] P. Vincent, Y. Bengio. Kernel matching pursuit. Machine Learning Journal. 48 (1): 165 – 187, 2002.

[18] X. Wei, Y. Gong. Document Clustering by Concept Factorization. In Proceedings of SIGIR, 202 – 209, 2004.

[19] Ken L. 20 Newsgroups dataset. http://people.csail.mit.edu/ jrennie/20Newsgroups/: MIT, 2008.