Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 7 (2007) 908–914
Applied Soft Computing
www.elsevier.com/locate/asoc

Text classification: A least square support vector machine approach

Vikramjit Mitra [a], Chia-Jiu Wang [b,*], Satarupa Banerjee [c]

[a] ECE Department, University of Maryland, College Park, MD, United States
[b] ECE Department, University of Colorado Colorado Springs, CO, United States
[c] CS Department Villanova University Villanova, PA, United States

Received 12 April 2004; received in revised form 1 March 2006; accepted 2 April 2006
Available online 5 June 2006

Abstract

This paper presents a least square support vector machine (LS-SVM) that performs text classification of noisy document titles according to different predetermined categories. The system's potential is demonstrated with a corpus of 91,229 words from University of Denver's Penrose Library catalogue. The classification accuracy of the proposed LS-SVM based system is found to be over 99.9%. The final classifier is an LS-SVM array with Gaussian radial basis function (GRBF) kernel, which uses the coefficients generated by the latent semantic indexing algorithm for classification of the text titles. These coefficients are also used to generate the confidence factors for the inference engine that present the final decision of the entire classifier. The system is also compared with a K-nearest neighbor (KNN) and Naïve Bayes (NB) classifier and the comparison clearly claims that the proposed LS-SVM based architecture outperforms the KNN and NB based system. The comparison between the conventional linear SVM based classifiers and neural network based classifying agents shows that the LS-SVM with LSI based classifying agents improves text categorization performance significantly and holds a lot of potential for developing robust learning based agents for text classification.

Keywords: Least square support vector machines; Latent semantic indexing; Text classification; Kernel based learning algorithms

1. Introduction

Rapid advancement in technology has motivated text documents to be available in electronic form. The World Wide Web itself contains a huge amount of documents, conference materials, publications, journals, editorials, news and information etc., available in electronic form. These materials along with others result in enormous amount of easily available information, which lack organization. The lack of organization of materials in the World Wide Web necessitates a growing interest in assisting people to manage the huge amount of information. Organized search, browsing, information routing, filtering, objectionable material identification, junk mail, topic identification etc., are the central issues in current information management efforts. This requires implementation of sophisticated learning agents that are capable of classifying relevant information and hence increases text organization. Previous research in the field of Internet agents has used manual or simple encoding techniques [1], linear SVMs and neural network [2] based intelligent agents for information retrieval.

Text classification (TC) is a text content-based classification technique that assigns texts to some predefined categories [3–5,19,27]. The key issues in TC are feature encoding and classifier design, tuning and implementation. Feature extraction is a method of document encoding; that automatically construct internal representations of documents. This paper aims to organize the materials available in a library, which has both electronic materials as well as physical documents. A library cataloging system usually stores the entire information regarding a material, which results in higher storage space requirement.

This paper presents an analysis of learning agents based on support vector machines (SVM). In particular least square support vector machine (LS-SVM) [13] with latent semantic indexing (LSI) for feature extraction will be explored. Furthermore the details of the internal structure of the classifying agent along with the results obtained from our research are presented. The results obtained from the LS-SVM based classifier will be compared with K-Nearest Neighbor (KNN) and Naïve Bayesian (NB) based classifier, which claims the potential and pertinence of using LS-SVMs as the intelligent classifying and search agents for semantic text classification.

* Corresponding author. Tel.: +1 719 262 3495; fax: +1 719 262 3589.
E-mail addresses: vmitra@umd.edu (V. Mitra), cwang@eas.uccs.edu (C.-J. Wang), satarupa.banerjee@villanova.edu (S. Banerjee).

1568-4946/$ – see front matter
doi:10.1016/j.asoc.2006.04.002

2. Latent semantic indexing

Latent semantic indexing commonly known as LSI [3–6] is a text classification and document indexing technique that generates a vector model of semantics based upon word co-occurrences. Using the estimate of the most significant statistical factors in the weighted word space, LSI [5,6] extracts the underlying semantic structure of a word corpus. LSI considers documents that have many words in common to be semantically close and those with few words in common to be semantically distant. This method emulates human knowledge to classify and categorize a document collection based on its content. LSI search agents look at similarity values it has calculated for every content word, and returns the documents that it thinks best fit the query [4]. This makes LSI successful where a plain keyword search will fail if there is no exact match. At the initial stage, LSI preprocesses the text corpus by purging all the extraneous words from a document [4,5], leaving only content words that have some semantic meaning. This way it eliminates those words that introduces noise to the decision making task.

LSI algorithm generates a matrix representation of the corpus, with rows corresponding to words in the vocabulary and columns to the documents. Each value in this matrix is a weighted frequency of the corresponding term in the corresponding document, which reduces the influence of frequently occurring term [7]. The matrix thus generated is large sparse one, which is then reduced to a compressed matrix based on singular value decomposition (SVD) technique, given in (1):

$$R = UPV \tag{1}$$

where the matrix R is decomposed into a matrix of reduced rank U, a diagonal matrix of singular values P and a document matrix V. The row vector of matrix U and the column vector of matrix V are the projections of word vectors and document vectors into singular value space.

3. Support vector machines (SVM)

The general form of support vector machine is used to separate two classes by a function, which is induced from available examples [8,15]. The main goal of this classifier is to find an optimal separating hyperplane that maximizes the separation margin or the distance between it and the nearest data point of each class. For a set of training vectors belonging to two separate classes, shown in (2):

$$\{(x^1, y^1), \ldots, (x^m, y^m)\}, \qquad x \in R^n, y \in \{1, -1\} \tag{2}$$

A hyperplane as shown in Eq. (3) can be found to separate these two classes:

$$\langle w, x\rangle + b = 0 \tag{3}$$

The above set of vectors is said to be optimally separated by the hyperplane if it is separated without error and the distance between the closest vector to the hyper plane is maximal. Vapnik [15,8] introduced a canonical hyperplane, where the parameters w, b are constrained by Eq. (4):

$$\min_i |\langle w, x^i\rangle + b| = 1 \tag{4}$$

From the above set of equations, it can be derived that the optimal separating hyperplane given by:

$$w^* = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{5}$$

$$b^* = -0.5\langle w^*, x_r + x_s\rangle \tag{6}$$

where α_i is the Lagrange multiplier. x_r and x_s are any support vectors from each class satisfying $\alpha_r > 0$, $y_r = -1$; $\alpha_s > 0$, $y_s = 1$.

3.1. Kernel functions

In the case where a linear boundary is inappropriate the SVM can map the input vector, x, into a high dimensional feature space, z [8]. By selecting the non-linear mapping as a priori, the SVM constructs an optimal separating hyperplane in this higher dimensional space. This idea exploits the method of Aizerman et al. (1964), which enables the curse of dimensionality (Bellman, 1961) to be addressed. The idea of kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Due to this the inner product does not need to be evaluated in the feature space, which provides a way of addressing the curse of dimensionality.

The theory of Reproducing Kernel Hilbert Space (RKHS) (Wahba, 1990; Aronszajn, 1950; Girosi, 1997; Heckman, 1997), claims that an inner product in feature space has an equivalent Kernel in input space:

$$K(x, x') = \langle \phi(x), \phi(x')\rangle \tag{7}$$

provided certain conditions hold. The Gaussian radial basis function (GRBF) has received significant attention and its form is given by:

$$K(x, x') = \mathrm{e}^{-\|x-x'\|^2/2\sigma^2} \tag{8}$$

Classical techniques utilizing RBFs employ some method of determining a subset of centers; typically a method of clustering is employed to select a subset of centers [8]. The most attractive feature of SVM is its implicit selection process, with each support vectors contributing on local Gaussian functions, centered at that data point. By further consideration it is possible to select the global basis function width using the SRM principle (Vapnik, 1995).

3.2. Least square-SVM

SVM is a powerful technique for solving problems in non-linear classification, function estimation and density estimation, which had led to many recent developments in kernel based learning methods [9,11,12,16,15]. Least square support vector machines (LS-SVM) are reformulations to

standard SVMs [13,14]. LS-SVMs are closely related to regularization networks [10] and Gaussian processes [17] but additionally emphasize and exploit primal-dual interpretations.

3.3. Why LS-SVM with LSI?

Kernel based learning Methods are highly sophisticated learning algorithms whose ideal example is SVM [18,9]. In SVM approach, items are mapped to high dimensional spaces, where information about their mutual positions is used for classification, regression or clustering. These systems work with high accuracy in text categorization, since the documents are usually represented by very high dimensional vectors and the standard information retrieval techniques are based on the inner product of the vectors. Joachims [19,27] has proved the suitability of SVM for text classification. This paper incorporates the LSI technique with LS-SVMs in order to incorporate more information in the kernel. SVM based systems were found to suffer during information retrieval [18], as the semantic relations between the data terms are not considered. Titles that have related topics but use different terms are mapped to distant regions in feature space. LSI enables the system to capture the semantic information and hence establishes the similarity between two titles by considering the relation between two terms. LS-SVM is well known to solve optimization problems with high accuracy. Because of these advantages an LSI based LSSVM architecture is selected for the text classification (TC) purpose. This paper uses document titles, instead of the entire document, mainly because of two reasons: (1) this paper addresses classification task for documents in a Library system, where the document information is mainly present in terms of the document title, ISBN number, document barcode, author name, publisher information etc. As apart from the document title none of the other information is relevant for the purpose of semantic classification, hence only the document title has been selected for this case. (2) The proposed classifier is also aimed to perform real time data classification, where classification based on a title will obviously be faster than the classification based on the entire document content.

3.4. K-nearest neighbor classification

K-nearest neighbor classification is a well-known statistical approach that has been widely applied to text classification since its inception [20,21]. The popularity of the algorithm is due to its simplicity. The classifier finds the k nearest neighbors of the test document, and then uses majority voting among the neighbors in order to decide the test document category. Similarity between two documents is measured by the cosine between the vectors representing the corresponding documents. If a specific category is shared by more than one of the K-neighbors, then the sum of the similarity scores of those neighbors is obtained from the weight of that particular shared category.

3.5. Naïve Bayes classifier

Naïve Bayes (NB) classifier is a probabilistic classifier that has been used extensively for the purpose of document classification [22]. NB classifier uses the joint probabilities of words and their categories to estimate the probabilities of categories given a test document using the celebrated Bayes rule. The naïve part in this classification algorithm is the assumption of word independence, which can be stated as, the probability of a word, given a category is independent from the conditional probabilities of other words given that same category; i.e. it does not use word combinations as predictors. The naïve assumption helps in saving computation time to a great extent [23,24].

4. Text classification

The Penrose library document collection contains real-world titles that are present in their cataloging system. All the titles in this word corpus belong to one or more of the six main categories: engineering (ENG), mathematics (MTH), music (MUS), history (HIS), economics and management (ECM) and literature and arts (LAR). Table 1 shows example titles from the Corpus and their categories. Eleven thousand eight hundred and ninety two titles were used for this experiment and each of them belongs to at least one of the above-mentioned categories. The total number of words used is 91,229, with 12,303 different

Table 1
Example titles from the Corpus and their categories

Semantic category	Title
Engineering (ENG)	Randomized algorithms for analysis and control of uncertain systems; applications in time–frequency signal processing; nanoelectro-mechanics in engineering and biology
Mathematics (MTH)	Stochastic processes with applications to finance; practical extrapolation methods; solving polynomial equation systems
Music (MUS)	Unplayed melodies; the music of European nationalism; elements of music
History (HIS)	A history of the Osage people; a history of modern Germany since 1815 Shivaji; Hindu King in Islamic India
Economics and management (ECM)	The economics of self-employment and entrepreneurship; ownership and governance of enterprises; the new knowledge management
Literature and arts (LAR)	Rethinking social realism: African American art and literature, 1930–1953; teaching and learning about multicultural literature; literary texts and the arts: interdisciplinary perspectives

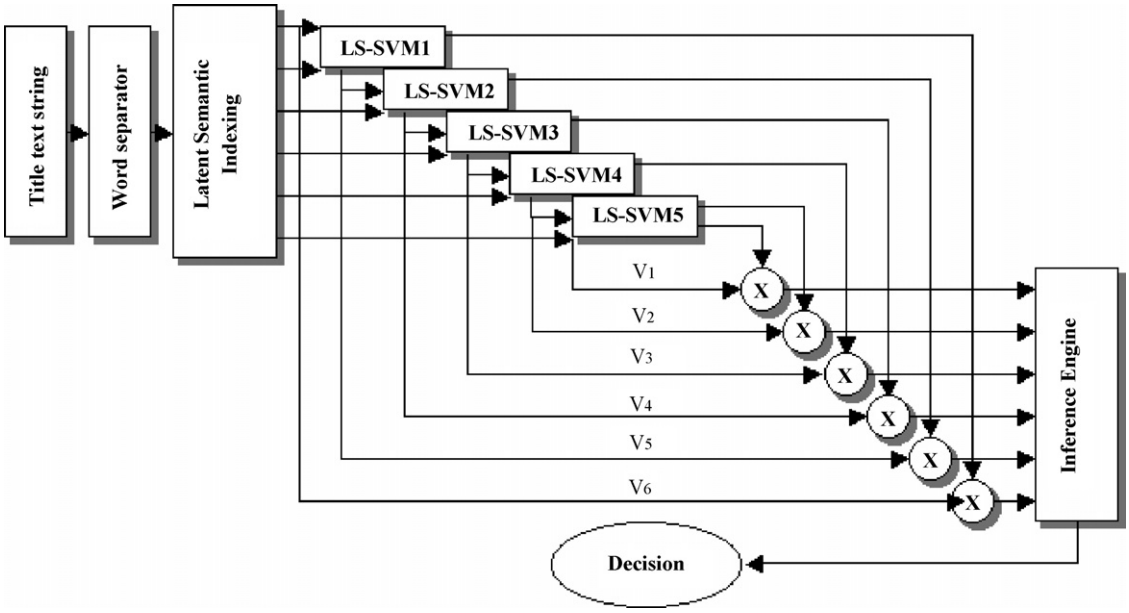

Fig. 1. Block diagram of the LS-SVM text classifier.

words. The first 482 titles from each category, which is 2892 titles altogether, were used for training the LS-SVM module. The remaining 9000 titles were used for the testing purpose.

LSI is used to represent the title corpus as semantic significant vectors 'V', which determines the frequency of word occurrences in different semantic categories. Each word 'a' is represented with a vector:

$$[V(a, b_1), V(a, b_2), \ldots, V(a, b_n)]$$

where b_i represents a certain semantic category. A value $V(a, b_i)$ is computed for each dimension of the semantic vector as the normalized frequency of occurrences of word 'a' in semantic category b_i divided by the normalized frequency of occurrences of word 'a' in the Corpus. Hence, the vectors V represent the plausibility of a word 'a' occurring in a particular semantic category 'b'. These vectors 'V' for each word 'a' together form the matrix R, which is a large sparse matrix due to the large number of words. R is then reduced to a compressed matrix based on singular value decomposition (SVD) technique, which generates the LSI coefficients; these coefficients are then fed to the SVM module.

From the model depicted in Fig. 1, it can be observed that at the beginning, the system starts with a text string. The word separator module separates each word from that string and feeds them one by one to the LSI module. LSI purges most of the extraneous words that have no semantic meaning and hence contributes little or nothing towards semantic categorization. Then LSI uses a stemming algorithm named 'Porter Stemmer' to remove common endings from the words. These words are further processed by the LSI algorithm to generate the LSI coefficients which are fed to five LS-SVM modules as shown in Fig. 1. Each of the first four LS-SVM module, classifies between one category and the rest, where as the last one classifies between two categories. The conclusions drawn by the LS-SVMs are then multiplied with the LSI coefficients $[V_1, V_2, V_3, V_4, V_5, V_6]$ that are used as confidence factors (CF) to increase the accuracy of the inference engine (IE).

In order to make an LS-SVM model, two parameters are required, γ, the regularization parameter, which determines the trade-off between the fitting error minimization and smoothness. The other parameter is the σ^2, which is the Gaussian bandwidth. The parameters are optimized by the use of Bayesian framework, which uses the eigen value decomposition of the kernel matrix. With increase in number of data points, the size of the kernel matrix increases and hence approximation techniques [25] are used to handle large data sets. It is also known [25] that the principal eigen values and corresponding eigen vectors are relevant, hence iterative approximation methods such as the Nyström method [25] is used for approximation. Input selection is performed by using automatic relevance determination (ARD) [26].

The regularization parameter, γ, and the Gaussian bandwidth parameter, σ^2, are inferred by optimizing the cost at the initial levels of inference. The optimization of a cost function with possibly multiple optimal points is performed by

Table 2
LS-SVM parameter values

Parameters (for tuning)	Parameter values
Optimization routine	Gridsearch
Cost function	Crossvalidate
γ (SVM-1)	1.23791
σ^2 (SVM-1)	12.1539
γ (SVM-2)	0.0444
σ^2 (SVM-2)	1.1728
γ (SVM-3)	0.6726
σ^2 (SVM-3)	0.3818
γ (SVM-4)	0.5342
σ^2 (SVM-4)	0.2466
γ (SVM-5)	0.0324
σ^2 (SVM-5)	0.1568

Table 3
Rule base for the decision logic

Categories	Vectors					
	A_1	A_2	A_3	A_4	A_5	A_6
ENG	$0.5 \leq A_1$	$A_2 < 0.5$	$A_3 < 0.5$	$A_4 < 0.5$	$A_5 < 0.5$	$A_6 < 0.5$
MTH	$A_1 < 0.5$	$0.5 \leq A_2$	$A_3 < 0.5$	$A_4 < 0.5$	$A_5 < 0.5$	$A_6 < 0.5$
MUS	$A_1 < 0.5$	$A_2 < 0.5$	$0.5 \leq A_3$	$A_4 < 0.5$	$A_5 < 0.5$	$A_6 < 0.5$
HIS	$A_1 < 0.5$	$A_2 < 0.5$	$A_3 < 0.5$	$0.5 \leq A_4$	$A_5 < 0.5$	$A_6 < 0.5$
ECM	$A_1 < 0.5$	$A_2 < 0.5$	$A_3 < 0.5$	$A_4 < 0.5$	$0.5 \leq A_5$	$A_6 < 0.5$
LAR	$A_1 < 0.5$	$A_2 < 0.5$	$A_3 < 0.5$	$A_4 < 0.5$	$A_5 < 0.5$	$0.5 \leq A_6$
No category	$A_1 < 0.5$	$A_2 < 0.5$	$A_3 < 0.5$	$A_4 < 0.5$	$A_5 < 0.5$	$A_6 < 0.5$

evaluating a grid over the parameter space and then selecting the minimum on that grid, which iteratively zooms into the candidate optimum, where a priori starting values specify the limits of the grid over parameter space. The optimization process optimizes the posterior probabilities of the hyper parameters with respect to the different Bayesian inference levels. Before optimization of the parameters, the model was initiated with appropriate starting values, where it optimizes the support values and the bias. After that the system was tuned to obtain the appropriate γ and σ^2 values, which were used as the initial values for a three stage Bayesian optimization, where the Bayesian framework is initialized at the first stage, optimized γ value was obtained in the second stage and the optimized σ^2 value was obtained in the third stage. The parameter values used are shown in Table 2.

The inference engine (IE) has a decision logic that uses a rule-based algorithm. It accepts a vector of six values [A_1, A_2, A_3, A_4, A_5, A_6] obtained by the product of the SVM decisions and the confidence factors. It uses this vector to make the final judgment. The rule base is given in Table 3. It should be noted that the rule base in Table 3 gives inferences only for a limited number of cases, as certain cases might arise where more than one member of the vector, A, have values $\geq$0.5. In such a case the decision will comprise all the weighted categories, corresponding to which the vectors are $\geq$0.5. In the practical research we have incorporated all the possible combination of vector A for which the values are $\geq$0.5.

If all the elements of the vector A are less than 0.5, then the decision logic infers 'No Category', as shown in the last row of Table 3. Another intuitive way to construct the decision logic would be to use the category of the vector element that has the maximum value, however, in this particular way it was observed that (1) the 'No category' case will never occur, (2) for two elements of vector A having the same maximum value, will result in unpredictable decision and finally and (3) due to lack of threshold, unrelated or semantically distant titles, that should belong to 'No Category' group will obtain a specific category, even though it does not belong to that category remotely. To

Table 4
Results using LS-SVM for text classification

Category	Training set (%)	Test set (%)
ENG	100.0	100.0
MTH	100.0	100.0
HIS	100.0	99.8
MUS	100.0	99.9
ECM	100.0	100.0
LAR	99.7	99.6
All titles	99.95	99.90

Table 5
Different stages of the text classification module

Test categories	LS-SVM (×CF) output						Decision logic output						Inference
	ENG	MTH	HIS	MUS	ECM	LAR	ENG	MTH	HIS	MUS	ECM	LAR	
ENG	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	ENG
	0.7	0.0	0.2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	ENG
MTH	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	MTH
	0.0	0.9	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	MTH
HIS	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	HIS
	0.0	0.0	0.9	0.0	0.0	0.1	0.0	0.0	1.0	0.0	0.0	0.0	HIS
MUS	0.0	0.0	0.0	0.7	0.0	0.3	0.0	0.0	0.0	1.0	0.0	0.0	MUS
	0.0	0.2	0.1	0.8	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	MUS
ECM	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	1.0	0.0	ECM
	0.0	0.1	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	1.0	0.0	ECM
LAR	0.0	0.0	0.0	0.2	0.0	0.7	0.0	0.0	0.0	0.0	0.0	1.0	LAR
	0.0	0.0	0.2	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.0	1.0	LAR

Table 6
Classification accuracy from LS-SVM, KNN and NB for the six different categories

Category	LS-SVM		KNN		NB	
	Training set (%)	Test set (%)	Training set (%)	Test set (%)	Training set (%)	Test set (%)
ENG	100.0	100.0	98.3	94.3	96.7	93.4
MTH	100.0	100.0	96.7	95.1	94.5	89.3
HIS	100.0	99.8	97.3	96.7	92.4	91.9
MUS	100.0	99.9	89.7	85.8	88.2	84.4
ECM	100.0	100.0	98.3	97.7	95.5	95.3
LAR	99.7	99.6	87.5	86.6	83.4	81.2
All titles	99.95	99.90	94.6	92.7	91.8	89.3

filter out these unwanted scenarios, the threshold based decision logic has been implemented in this research.

5. Simulation results

In this study a six-input one-output system is considered, the six inputs corresponding to the six LSI coefficients generated after processing the text titles, where each coefficient gives the membership value of that word to a certain category. The test results are presented below in Tables 4 and 5. Table 4 depicts the accuracy for text classification in different categories, whereas Table 5 depicts the different stages of the classifying module. The first group of six columns in Table 5 presents the data obtained from the product of the SVM decision with the confidence factors (CF), the next group of six columns gives the decision logic output based on the rule base given in Table 2 ad the last column gives the final decision presented by the Inference Engine.

As evident from Table 5, the decision logic based on a rule base makes a wise estimation of the SVM outputs, based on which the Inference engine selects the category to which a specific string of word belongs. The output of the system is thus the final decision presented by the Inference engine. In case of a multi-class problem, the IE will present the weighted decision of both the classes.

Apart from 2892 regular titles that belonged to unique categories, 250 additional titles were used, out of which 125 titles belonged to more than one category and the remaining 125 titles belonged to none of the six pre-specified categories. For multi-category cases, titles like 'Introduction to Engineering Mathematics and Calculus' etc., were used where, clearly the category belongs to both engineering (ENG) and mathematics (MTH). In all the multi-class cases, the LS-SVM based architecture detected the multiple classes accurately, where more than one elements of the vector A had value ≥ 0.5. For titles that belonged to neither categories, it was found that the proposed LS-SVM classifier detected them with a high degree of accuracy and all of the elements of the vector A had value <0.5. Instead of using the threshold based decision logic, if the maxima based decision logic was used, it was observed that the 'no category' cases yielded specific categories corresponding to the element that had the maximum value, even though the maximum element value was below 0.3. More over the other short coming of the maxima based decision logic was the failure to detect multiple classes, in which case even though two elements of the vector A had value greater than 0.5, but the maximum value was unique and the decision logic inferred the class corresponding to the maximum element, ignoring the next maximum element(s), even though they were above the threshold value of 0.5. The threshold value of 0.5 for the decision logic was obtained by trial and error, after processing a large number of document titles, where the obtained classes from the LS-SVM classifier were compared against the pre-specified classes of the text materials.

Parallel to the LS-SVM classifier, a K-nearest neighbor and Naïve Bayesian classifier based algorithms were also implemented for the purpose of the proposed text classification task. Table 6 presents the classification accuracy obtained from the implementation of them. LSI coefficients were used as the input to the KNN and NB based classifiers. The value of K for the KNN classifier was selected to be 28. This value was obtained by implementing KNNs for different values of K and thus optimizing the performance of the KNN classifier according to the pre-specified classes of the text titles.

It can be observed from Table 6, that the LS-SVM based classifier provided better accuracy than the remaining two. The KNN, however provided better accuracy than NB based classifier, in all the categories, but still the obtained accuracy from LS-SVM classifier clearly claims, the superiority in the performance of the LS-SVM based classification for the proposed task of text classification.

6. Conclusion

This paper presents an LSI coefficient based LS-SVM module, for text classification. With a corpus of 91,229 words, the classification accuracy rate was fairly high with 99.9% accuracy in classifying the titles. Table 5 depicts the different stages of the proposed text classification module for only two samples per category, but actually 1500 samples per category (for six different categories) were used, which is equivalent to 9000 samples altogether. Thus 9000 samples were used for testing and Table 6 depicts that the system efficiently classified those samples with a high degree of accuracy. The overall accuracy for classifying the different categories is presented in Table 4. The high accuracy justifiably claims that LS-SVMs are highly capable to perform text classification tasks. Text documents are characterized by their high dimension and the

sparse property. SVMs use over-fitting protection and they have been proved to be well suited for both sparse and dense problems, these attributes make them particularly well suited for text classification. Previous works on text classification have implemented neural networks and linear SVMs, where recurrent neural networks and linear SVMs showed an accuracy of 93.05% [2] and 97%, respectively. The precision rate of the proposed system clearly outperforms the previous models.

Prior to our research, LS-SVMs with GRBF kernel and LSI techniques have not been designed and implemented for a high scale task of text categorization. The robustness of the proposed system enables it to classify noisy text titles with a high degree of precision. Scalability is one of the key issues concerning text classification. The proposed classifier addresses classification between six classes and it was observed that as the number of classes was increased, the number of LSI coefficients also increased, thus increasing the feature set resolution. Due to this, for a small increase in the number of classes, no major deviation from the obtained accuracy was noted. Future research should be devoted to explore SVMs with other possible kernel functions or Neuro-SVM based hybrid models that might act as a better classifying agent. Future direction should also address to increase the number of categories to a large extent to observe the robustness and predictive accuracy of the system.

Acknowledgement

The authors would like to acknowledge Penrose Library, University of Denver, for providing real world experimental data.

References

[1] H. Schuetze, D.A. Hull, J.O. Pedersen, A comparison of classifiers and document representations for the routing problem, in: Proceedings of the Special Interest Group on Information Retrieval, SIGIR Forum; Seattle, WA, The Association for Computing Machinery, New York, 1995 , pp. 229–237.

[2] S. Wermter, G. Arevian, C. Panchev, Recurrent neural network learning for text routing, in: Proceedings of Ninth International Conference on Artificial Neural Networks, vol. 2, ICANN 99, (1999), pp. 898–903.

[3] P.W. Foltz, W. Kintsch, T.K. Landauer, The measurement of textual coherence with latent semantic analysis, Discourse Process. 25 (1998) 285–307.

[4] P.W. Foltz, Using latent semantic indexing for information filtering, in: R.B. Allen (Ed.), Proceedings of the Conference on Office Information Systems, Cambridge, MA, (1990), pp. 40–47.

[5] T.K. Landauer, P.W. Foltz, D. Laham, Introduction to latent semantic analysis, Discourse Process. 25 (1998) 259–284.

[6] T.K. Landauer, S.T. Dumais, Solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge, Psychol. Rev. 104 (2) (1977) 211–240.

[7] R. Bellegarda, A multi-span language modeling framework for large vocabulary speech recognition, IEEE Trans. Speech Audio Process. 6 (1998) 456–467.

[8] S.R. Gunn, Support Vector Machines for Classification and Regression, Technical Report, Department of Electronics and Computer Science, University of Southampton, 1998.

[9] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.

[10] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, Advances in Computational Mathematics No. 1, vol. 13, Springer Science+Business Media (formerly Kluwer Academic) Publishers, 2000, pp. 1–50.

[11] B. Schölkopf, C. Burges, A. Smola, Advances in Kernel Methods–Support Vector Learning, MIT press, 1998.

[12] B. Schölkopf, A.J. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigen value problem, Neural Computation, vol. 10, MIT Press, 1998, pp. 1299–1319.

[13] J.A.K. Suykens, J. Vandewalle, Least square support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.

[14] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least square support vector machine classifiers, Machine Learning, vol. 54, Kluwer Academic Publishers, 2001, pp. 5–32.

[15] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

[16] V. Vapnik, Statistical Learning Theory, John Wiley, New York, 1998.

[17] G. Wahba, Spline models for observational data, in: CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59, SIAM, PA, 1990.

[18] N. Cristianini, J. Shawe-Taylor, H. Lodhi, Latent semantic kernels, J Intell. Inform. Syst. 18 (2–3) (2002) 127–152.

[19] T. Joachims, Text categorization with support vector machines, in: Proceedings of European Conference on Machine Learning ECML-98, Springer, 1998, pp. 137–142.

[20] Y. Yang, X. Liu, A re-examination of text categorization methods, in: The 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), Berkley, CA, USA, (1999), pp. 42–49.

[21] Y. Yang, An evaluation of statistical approaches to text categorization, Information Retrieval, Issue 1–2, vol. 1, Kluwer Academic Publishers, 1999, pp. 69–90.

[22] K. Nigam, A.K. McCallum, S. Thrun, T.M. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learning, No. 3(2), vol. 39, Kluwer Academic Publishers, 2000, pp. 103–104.

[23] D. Lewis, M. Ringuette, A comparison of two learning algorithms for text categorization, in: Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval, 1994, pp. 81–93.

[24] Z. Zheng, X. Wu, S. Srihari, Feature selection for text categorization on imbalanced data, SIGKDD Explor. 6 (1) (2004) 80–89.

[25] T. Van Gestel, J.A.K. Suykens, B. De Moor, J. Vandewalle, Bayesian inference for LS-SVMs on large data sets using the Nyström method, in: International Joint Conference on Neural Networks (WCCI-IJCNN 2002), Honolulu, USA, (2002), pp. 2779–2784.

[26] T. Van Gestel, J.A.K. Suykens, B. De Moor, J. Vandewalle, Automatic relevance determination for least squares support vector machine classifiers, in: Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2001), Bruges, Belgium, (2001), pp. 13–18.

[27] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the 10th European Conference on Machine Learning, ECML-98, No. 1398, Springer Verlag, Heidelberg, DE, (1998), pp. 137–142.