Error Detection of Personalized English Isolated-Word Using Support Vector Machine

Tiong Sieh Kiong, Abu Bakar Hasan, Johnny Koh Siaw Paw and David Yap Fook Weng

1Center of System and Machine Intelligence, Universiti Tenaga Nasional, 43000, Kajang, Selangor, Malaysia
2Power Engineering Centre, Universiti Tenaga Nasional, Kajang, 43000, Selangor, Malaysia
3Faculty of Engineering, Universiti Teknikal Malaysia Melaka, 76109, Melaka, Malaysia

Corresponding Author: Tiong Sieh Kiong, Center of System and Machine Intelligence, Universiti Tenaga Nasional, 43000, Kajang, Selangor, Malaysia Tel: +603 8921 2282 Fax: +603 8921 2116

ABSTRACT

A better understanding on word classification could lead to a better detection and correction technique. In this study, a new features representation technique is used to represent the machine-printed English word. Subsequently, a well-known classification type of artificial intelligent algorithm namely Support Vector Machine (SVM) is used to evaluate those features under two class types of words with proper segregation of correct and erroneous words in two data sets. Our proposed model shows good performance in error detection and is superior when compared with neural networks, Hamming distance or minimum edit distance technique; with further improvement in sight.

Key words: Error detection, support vector machine, English isolated-word, classification

INTRODUCTION

Feature selection, also known as subset selection or variable selection, is a process commonly used in machine learning, whereby a subset of the features available from the data are selected for application of a learning algorithm (Kukich, 1992; Blum and Langley, 1997). Feature selection can be a powerful tool for simplifying or speeding up computations and when employed appropriately it can lead to a little loss in classification quality (Weston et al., 2011). Selecting an optimal set of features is in general difficult, both theoretically and empirically (Dasgupta et al., 2007). For practical supervised learning algorithms, the selection is for a satisfactory set of features instead of an optimal set. In machine learning, satisfactory features selection is typically done by cross-validation that is, by looking at which feature sets that could provide a reasonable realization error.

A distinction must be made between the tasks of error detection and correction (Drucker et al., 1996). Efficient techniques such as n-gram analysis and lookup tables have been devised for detecting strings that do not appear in a given word list, dictionary or lexicon. Many existing spelling correctors exploit task-specific constraints and focus on isolated words and uses different techniques such as rule-based and minimum edit distance technique (Hsu et al., 2003).

We propose a technique using Support Vector Machine (SVM) to detect any error in an isolated English word and propose another possible correct word as the solution to replace the incorrect word. We use Support Vector Classification (SVC) to detect an error in a given word and finally in
order to find a possible correct word as the replacement we use Support Vector Regression (SVR). The main objective of this study is to a better technique that could detect and correct an incorrect-isolated English word and that could be used as an alternative technique in the near future.

**THEORY ON SUPPORT VECTOR MACHINES**

Basically, Support Vector Machines (SVM) is based on the Structural Risk Minimization principle (SRM) from the statistical learning theory (Chang and Lin, 2007). SVM is a set of related supervised machine learning methods used for classification, on the other hand, the empirical risk minimization principle which is used by neural network to minimize the error on the training data, the SRM minimizes a bound on the testing error, thus allowing SVM to generalize better than conventional neural network (Nagi et al., 2008). Apart from the problem of poor generalization and overfitting in neural network, SVM also address the problem of efficiency of training and testing and the parameter optimization problems frequently encountered such as getting trapped into the local minimum in neural network (Zhan and Shen, 2005) as shown in Fig. 1.

Support vectors in SVM can be categorized into two types: the training samples that exactly locate on the margin of the separation hyperplane and the training samples that locate beyond their corresponding margins, whereby the later is regarded as misclassified samples (Wright, 2011). Given training data consisting of labeled vectors represented by \( \{x_i, y_i\} \) for \( i = 1, 2, \ldots, m \) where, \( x_i \in \mathbb{R}^n \) represents an \( n \)-dimensional input vector and \( y_i \in \{1, +1\} \) represents the class label. These training patterns are linearly separable if a vector \( w \) (orientation of a discriminating plane) and a scalar \( b \) (offset of the discriminating plane from origin) can be defined so that inequalities in Eq. 1 and 2 are satisfied (Kukich, 1992):

\[
\begin{align*}
wx_i + b &= 1 & \text{if} & & y_i &= +1 & (1) \\
wx_i + b &= -1 & \text{if} & & y_i &= -1 & (2)
\end{align*}
\]

A hyperplane which divides the data is to be determined. This amounts in determining \( w \) and \( b \) so that:

\[
y_i (wx_i + b) = 0, \quad \text{for} \quad i = 1, 2, \ldots, m \quad (3)
\]

![Fig. 1: The phenomenon of “local minimum” in a neural network](image)

664
If a hyperplane is satisfied, then two classes are known to be linearly separable. Then Eq. 3 is written as:

\[ y(w, x_i + b) = 1 \]  \hspace{1cm} (4)

If the data is not linearly separable, another parameter called the slack variable \( \xi_i \) for \( i = 1, 2, \ldots, m \) is introduced with \( \xi_i > 10 \) such that Eq. 4 can be represented as:

\[ y_i(w, x_i + b) - 1 - \xi_i = 0 \]  \hspace{1cm} (5)

The solution to find a generalized optimal separating hyperplane can be obtained using the following condition (Abu Zarim et al., 2008):

\[ \min \left[ \frac{1}{2} \| w \|_2^2 + C \sum_{i=1} \xi_i \right] \]  \hspace{1cm} (6)

The first term in Eq. 6 above controls the learning capacity while the second term controls the number of misclassified points. The penalty parameter \( C \) is selected by the user (Nagi et al., 2008) which is viewed as a regularization parameter that characterizes willingness to accept possible misclassifications in linearly non-separable datasets. The classification or decision function is:

\[ f(x) = \text{SIGN} \left[ \sum_{i=1} \alpha_i (y_i (w, x_i + b)) - \sum_{i=1} \alpha_i K(x_i, x) + b \right] \]  \hspace{1cm} (7)

whereby, \( \alpha_i \) is the support vector of nonnegative Langrange multipliers.

Parameters \( \alpha_i \) and \( b \) are to be determined by SVM's learning algorithm (Vapnik, 1998) during the classification process. Parameter \( y_i \) and \( K(x_i, x_j) \) is the label or class type and the Kernel function, respectively. Parameter \( x_i \) is the training or test data while \( x_j \) is the input data for prediction. On the other hand, in order to solve nonlinear problems SVMs use a kernel function to allow a better fitting of the hyperplane to more general datasets. There are a few types of kernel functions proposed by Vapnik (1998), for example, the Radial Basis Function (RBF) kernel which non-linearly maps samples into a higher dimensional space. The RBF kernel has less numerical difficulties and has the property defined by \( K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \). The kernel bandwidth parameter \( \gamma \) controls the scaling of the mapping (Vivekan, 2010) and \( C \) for the RBF kernel are pre-determined by \( v \)-fold cross validation or grid-search (Cristiani and Taylor, 2000). If the number of features is large, linear kernel was selected instead of RBF kernel because nonlinear mapping does not improve the SVM performance. A recent result shows that if RBF is used with model selection, then there is no need to consider the linear kernel (Gunn, 1997). At the moment SVM usage generally covers data classification and regression only (Drucker et al., 1996; Kohavi and John, 1997).

**METHODOLOGY ON SVM CLASSIFICATION MODEL**

We defined word literally as a word that could be correctly read or identified without any ambiguity and non-word is defined as vice-versa. The scope of our study comes under the isolated-word error category (Kukich, 1992) and specifically on machine printed English word only. In this
study, we are proposing a model to represent a word to meet the SVM classification requirements (Fig. 2), with a maximum of ten attributes or features namely numbers of consonant or non-vowel (nv) and vowel (v) character in them, total letter length (len) and their weight (w), weight of first (w1) and last letter (wn), position-vowel factor (pvf), sum of position-consonant-vowel factor (pcf), position factor (pf) and lastly, the type for the first letter (typ) in the word. In order to classify these words correctly, hereby we are proposing five rules as stated below:

**Rule 1:** The number of non-vowel and vowel in a word equals to the length of the word: \( \text{len} = \text{nv} + \text{v} \)

**Rule 2:** The weight of each word is equal to the word own weight. We define each character has a specific weight: \( (A, a) = 1, (B, b) = 2, (C, c) = 3, (D, d) = 4, \) until \((Z, z) = 26\) and \((\) = 0: \( w = w_1 + w_2 + ... + w_n \))

**Rule 3:** The type for first character of the word is set to 0 if the first character is a non-vowel and 1 if vowel

**Rule 4:** The position factor is the sum of position-vowel factor and the sum of position-consonant-vowel factor: \( \text{pf} = \text{pcf} + \text{pvf} \)

**Rule 5:** If the weight of the first letter in the first word is different from the first letter of the second word and consecutively, then either word is different

To illustrate our proposed model, for example consider the word reddish, Table 1 illustrates the attributes value with the help of Rule 1, 2, 3 and 4.

We used LibSVM (Hsu et al., 2003; Chang and Lin, 2007) which is a SVM learning tool package, for the training and testing process of the normalized words. The five rules above will
Table 1: Attribute values for 'reddish'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nw</td>
<td>5</td>
<td>wn</td>
<td>8</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
<td>pvf</td>
<td>56</td>
</tr>
<tr>
<td>len</td>
<td>7</td>
<td>pcf</td>
<td>216</td>
</tr>
<tr>
<td>w</td>
<td>67</td>
<td>pf</td>
<td>271</td>
</tr>
<tr>
<td>w1</td>
<td>18</td>
<td>typ</td>
<td>0</td>
</tr>
</tbody>
</table>

1. Set all the training and testing data (i.e., the words) attributes to ten
2. Normalize the raw data
3. Find the best value for parameters C and γ during cross
4. Get the support vectors from the classification process
5. Prompt the user to enter the testing data
6. Calculate the decision function
7. Prints the class for each of test data

Fig. 3: Pseudo-code of training and testing engine

Fig. 4: Training data on SVM-toy

mathematically play an important function in the classification of each word. For the classification, we are proposing a 2-steps simplified data processing procedure as follows (Fig. 3):

**Step 1:** Change the raw data into SVM format. Select the cross-validation parameter (v) equal 10 and the kernel function (RBF). By choosing v = 10, LibSVM package (Chang and Lin, 2007) will automatically divides the training data into ten different parts with basically equal numbers of data. Train the ten parts of these training data (Fig. 4) to determine the best value for parameters C and γ. The best values refer to the highest efficiency
percentage in classifying the training data. After the best \((C, \gamma)\) was chosen, the whole training set is trained again to generate the final classifier or model (Fig. 5), in which support vectors from the trained data are being determined automatically by LibSVM package (Fig. 6). These support vectors play an important function in classifying those trained data.

**Step 2:** Test the unknown data to obtain the final classification efficiency which represents how good the model obtained in step 1 in identifying other data.

**RESULTS AND DISCUSSION**

We found that SVM can be trained on words with and without spelling errors, thus it has the potential to adapt to the unique patterns for each different word. This unique pattern characterized the personnel preference of a person to certain words. Meaning that a person tends to be personalized when comes to using or choosing a word; another person may prefer to use a different set of words during conversation, texting or writing. We define these preferences as the
personalized dictionary of a particular person; thus, we managed to create a data base with a smaller number of words for SVM training purpose. To find the extent of our proposed classification model, we carried out two different studies in which all our data (training and testing) has been normalized accordingly to the range of (0, 1) and each word appears only once in our data base.

For the first work, we took three drafts of three different technical reports of a student, namely Report 1, 2 and 3, whereby normally these reports (Vivekan, 2010) have to be checked, corrected and commented by the respective supervisor before the reports are being allowed to be submitted to the University. We found that in those reports there are very few words with spelling errors made by the student himself. The pattern on the words used in writing three different reports are distinguishable for the student, meaning that he tends to use certain particular words in his writing. Since, the total numbers of spelling errors in the reports are not many, thus, we have created 418 (or 26.2%) spelling errors ourselves from the total 1595 words. We decided to divide those data into 70:30 ratios, whereby 478 (30%) of the data will be used for testing purposes with \( v = 10 \). This time we obtained the testing accuracy of 77% with the values of best \((C, \gamma) = (1, 1)\) and number of attributes equals seven. In the second work, we increased the number of attributes and training words but maintaining both parameters \( C, \gamma \) and \( v \); giving a much better test accuracy value of 79%, as shown in Table 2. Table 3 shows that as we increased the number of attributes used, subsequently we observed the training and test accuracies remained constant at \( C = 1, \gamma = 1 \) and \( v = 10 \). Meaning that for our case study, the accuracy did not depend on the number of attributes of a text. This finding is in line with Kukich (1992) work, whereby he mentioned that there was no optimal value for the number of attributes that one could consider the most appropriate to use.

In order to improve the accuracy, we have tried to rescale our data from \((0, 1)\) range to \((-1, 1)\) range as proposed by Hsu et al. (2003). We observed no change in the value of the test accuracy for both data samples; meaning that our normalization of all the words to \((0, 1)\) range is acceptable. For future work on the accuracy improvement possibility, one that we could look into is word pronunciation. Wright (2011) has proposed the 4S approach, whereby for a good spelling skill, one must understand the importance of vowels in any syllable in a word. Presently, our proposed model does not involve voice for pronunciation purpose as one of the attributes, meaning that a unique technique has to be deployed so that words can be 'logically' pronounced for the purpose of SVM processes. Comparing work by Lyon and Yaeger (1996), Al-Jawfi (2009) and Khawaja et al. (2006) in classification of hand-printing English, Arabic and Chinese characters using neural network, respectively, we found that their selected attributes was mainly based on the character size or the

<table>
<thead>
<tr>
<th>No. of attributes used</th>
<th>No. of training word</th>
<th>No. of test word</th>
<th>Test accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1117</td>
<td>478</td>
<td>77</td>
</tr>
<tr>
<td>10</td>
<td>1142</td>
<td>451</td>
<td>79</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of attributes used</th>
<th>No. of training word</th>
<th>No. of test word</th>
<th>Training accuracy (%)</th>
<th>Test accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1142</td>
<td>451</td>
<td>69.58</td>
<td>78.44</td>
</tr>
<tr>
<td>6</td>
<td>1142</td>
<td>451</td>
<td>69.58</td>
<td>78.44</td>
</tr>
<tr>
<td>8</td>
<td>1142</td>
<td>451</td>
<td>70.00</td>
<td>79.00</td>
</tr>
<tr>
<td>10</td>
<td>1142</td>
<td>451</td>
<td>70.00</td>
<td>79.00</td>
</tr>
</tbody>
</table>
Table 4: Comparison with Kuich (1992) study

<table>
<thead>
<tr>
<th>Technique</th>
<th>1142-word lexicon (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test accuracy</strong></td>
<td></td>
</tr>
<tr>
<td>Minimum edit distance (use grepe)</td>
<td>62</td>
</tr>
<tr>
<td>Similarity key (boscast token reconstruction)</td>
<td>78</td>
</tr>
<tr>
<td>Simple N-gram vector distance (use hamming distance)</td>
<td>68</td>
</tr>
<tr>
<td>Probabilistic (use Kournah-Church-Gale error probe)</td>
<td>78</td>
</tr>
<tr>
<td>Neural network (use back propagation classifier)</td>
<td>75</td>
</tr>
<tr>
<td>Support vector machines (use radial basis function)-proposal technique</td>
<td>79</td>
</tr>
</tbody>
</table>

Table 5: Comparison with UCI benchmark repository

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Cross-validation</th>
<th>Test accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (use Gauss C = 1, s = 0.1)</td>
<td>10x</td>
<td>76.6</td>
</tr>
<tr>
<td>Fuzzy (use SOM)</td>
<td>2x</td>
<td>73.5</td>
</tr>
<tr>
<td>SVM (use RBF)</td>
<td>N/A</td>
<td>53.5</td>
</tr>
</tbody>
</table>

Table 6: Partial calculation for Kernel function K(x_i, x_j) for i = 1 and 2, γ = 1

| Iteration | a_i, a_j | Kernel K(x_i, x_j) = \text{exp}(-\gamma ||x_i - x_j||^2) |
|-----------|---------|--------------------------------------------------------|
| 1         | 0.9886678441001486 | EXP[-SQR((0.033282905-0.148260212)^2+(0.081141998-0.211119459)^2+(0.064791562-0.189854345)^2+(0.33333333-0.44444444)^2+(0.25-0.5)^2+(0.25-0.4375)^2+(0.18636364-0.054666667)^2+(0.2-0.48)^2+(0.45833333-0.541666667)^2+(0-0)^2)] |
| 2         | 0.639286959619962  | EXP[-SQR((0.019667171-0.148260212)^2+(0.072875536-0.211119459)^2+(0.054745359-0.189854345)^2+(0.22222222-0.44444444)^2+(0.25-0.5)^2+(0.1876-0.4375)^2+(0.21818182-0.353636364)^2+(0.96-0.48)^2+(0.70833333-0.541666667)^2+(0-0)^2)] |

pixel counts obtained from character own images. Rani *et al.* (2011) used Gabor filters to get the concentration of energies in various directions for identifying printed English numerals at word level from Punjabi words document and she has chosen 5-fold cross-validation. Our proposed SVM model is using seven to ten different behavioral features in representing an English word and we used 10-fold cross-validation. Referring to Table 4, we observed that our work managed to produce the highest accuracy; meaning that SVM could provide us the best tool when comes to classification of isolated-word error in texts. Table 5 shows the comparison with UCI benchmark repository on the related work.

During the classification process, one of the file produced was a model file (Fig. 7) from which we managed to obtain the parameter \( b = \text{rho} = 0.987929, \gamma = \text{gamma} = 1 \) and the summation of Eq. 7 can be partly shown as in Table 6 for a set of prediction data \( X_{ij} = 0.148260212, X_{i1} = 0.211119459, X_{i3} = 0.189854345, X_{i4} = 0.444444444, X_{i5} = 0.5, X_{i6} = 0.4375, X_{i7} = 0.363636364, X_{i8} = 0.48, X_{i9} = 0.541666667, X_{i10} = 0. \) Parameter \( m \) in Eq. 7 equals to the number of support vectors that is, \( m = i = \text{total SV} = 798 \). Obviously, The RBF kernel is actually calculating the exponential of the negative of Euclidean distance between the support vectors and the unknown vector points that we want to predict.

Accordingly, it can be shown that the decision function \( f(x_i) \) for \( i = 1 \) becomes:

\[
f(x_i) = [0.9886678441001486 \times (0.033282905-0.148260212)^2+(0.081141998-0.211119459)^2+(0.064791562-0.189854345)^2+(0.33333333-0.44444444)^2+(0.25-0.5)^2+(0.25-0.4375)^2+(0.18636364-0.363636364)^2+(0.2-0.48)^2+(0.45833333-0.541666667)^2+(0-0)^2] + (-0.987929) - 0.586585013
\]
Table 7: LibSVM classification selected results

<table>
<thead>
<tr>
<th>Original label</th>
<th>Test data or word</th>
<th>f(x)</th>
<th>Class obtained</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hazardous</td>
<td>SIGN (1.09536784)</td>
<td>-1</td>
<td>Classified correctly</td>
</tr>
<tr>
<td>0</td>
<td>hazardous</td>
<td>SIGN (-0.882304417)</td>
<td>-1</td>
<td>Classified correctly</td>
</tr>
<tr>
<td>0</td>
<td>hazardous</td>
<td>SIGN (-0.905256828)</td>
<td>-1</td>
<td>Classified correctly</td>
</tr>
</tbody>
</table>

```
svm_type c_svc
gamma 1
kernel_type rbf
total_sv 798
label 1 0
nr_sv 451 347
SV
0.98866738441001486 1.0 0.0332823905 2.0 0.0811419983 3.0 0.064791562
4.0 0.333333333 5.0 0.25 6.0 0.25 7.0 0.18060364 8.0 0.2 9.0 0.45833333 10.0
0.03282809599619362 1.0 0.019667171 2.0 0.072877536 3.0 0.064746393
4.0 0.222222222 5.0 0.25 6.0 0.1875 7.0 0.21818182 8.0 0.96 9.0 0.70833333 10.0
```

Fig. 7: Sample of model output file

Similarly, the other values for the decision function for i = 2-798 can be mathematically calculated as f(x2) = 0.224201394 and f(x798) = -0.309491908. Hence:

\[
f(x) = \text{SIGN} \left[ \sum_{i}^{798} y_{i} a_{i} K(x_i, x_j) + b \right] = \text{SIGN} \left[ \sum_{i}^{798} a_{i} y_{i} + b \right] = \text{SIGN} \left[ 0.586585013 + 0.313526685 \cdots + (0.523892553) \right] = \text{SIGN} \left[ 0.913826625 \right] = +1
\]

Graphically f(x) is a mapped point on the positive side of the hyperplane of the feature space, meaning that the test data has been classified into a+1 class. Rechecking with our data base shows that the test or predicted data used in the above calculated was the word ‘maintain’. The word was originally labeled as Label 1 word which means it was a correctly printed English word. Figure 6 represents the overall plot for 478 test data which have been successfully classified into two different classes. Table 7 shows the classification results on three other test data or words.

CONCLUSION

In this study, we have demonstrated in detail the ability of SVM in classifying isolated-word error for English word and one should consider SVM when dealing with classification of data. SVM technique produces the better accuracy for English isolated-word error detection when compared to the other techniques mentioned in this study.

REFERENCES