Abstract—The goal of this research is to explore effects of dimensionality reduction and feature selection on the problem of script identification from images of printed documents. The k-adjacent segment is ideal for this use due to its ability to capture visual patterns. We have used principle component analysis to reduce the size of our feature matrix to a handier size that can be trained easily, and experimented by including varying combinations of dimensions of the super feature set. A modular approach in neural network was used to classify 7 languages – Arabic, Chinese, English, Japanese, Tamil, Thai and Korean.

Index Terms—Feature reduction; Feature selection, Neural Networks, Principle component analysis; Script identification

I. INTRODUCTION

The application of a language identification system is diverse. Almost all OCR, document indexing and classifier applications such as Google books, and all translation systems require a prior knowledge of the language used in the document. Printed documents offer a structured layout and intra-class repeatability in pattern, but they also pose the problem of repeated patterns in different languages which makes it difficult to identify discriminating patterns for inter-class classification. Much works have been done in the field of document image classification based on language. These approaches exploit different features found in distinct languages. Schwwarz et la [1] used character-based features and cluster analysis to distinguish two languages – English and Arabic. Spitz [2] used topline, baseline, and different zones in a line to extract features. Such techniques require a spatial structure and fail to perform when presented with different spatial arrangement of texts. It is observed that characters are not always well segmented, and even the best character segmentation technique poses the possibility of fragmentation of characters. The approach by Boles et la used connected component filtering to remove continuous segments of characters. This approach causes loss of information and segments and is inappropriate for use in both handwritten and printed text as in both cases, such possibilities exist. All the features and techniques discussed above are difficult to extend to new languages. The features employed are hand-picked to work for a specific set of languages. When new languages are introduced, the same set of features may not work and a new set of features need to be derived.

Bowers et la [4] used Gabor filters in small segments of text. However, this is restricted to a defined range of text segments, and no report of applying this technique on full length document image is available. Tan [5] used template based techniques to compute the most likely script after construction of templates from clusters. These techniques are dependent on line and character segmentation. In our work, we explore the possibility of a feature that is not dependent on line or character segmentation, can be extended to other languages, and also captures the inherent characteristics of a language.

II. METHODOLOGY

A. Data Set Description

For this work, we have used a standard dataset reported by Kelly et la [6]. It contains 62 images of total 7 languages. The following table lists the languages and the number of images for each language.

<table>
<thead>
<tr>
<th>Language (7)</th>
<th>No. of images (62)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>10</td>
</tr>
<tr>
<td>Chinese</td>
<td>9</td>
</tr>
<tr>
<td>English</td>
<td>10</td>
</tr>
<tr>
<td>Japanese</td>
<td>8</td>
</tr>
<tr>
<td>Korean</td>
<td>8</td>
</tr>
<tr>
<td>Tamil</td>
<td>9</td>
</tr>
<tr>
<td>Thai</td>
<td>7</td>
</tr>
</tbody>
</table>

TABLE I LIST OF LANGUAGES IN THE DATA SET

This research is supported by the City Group Bangladesh (http://www.citygroup.com.bd/).

Saami Rahman is with the Computer Vision & Cybernetics Research, SECS, Independent University, Bangladesh, Bashundhara, Dhaka 1229, Bangladesh (e-mail: saamirahman@gmail.com)

Bruce Poon is with the School of Electrical & Information Engineering, University of Sydney, NSW 2006, Australia (e-mail: bruce.poon@sydney.edu.au).

M. Ashraful Amin, is with the Computer Vision & Cybernetics Research, SECS, Independent University, Bangladesh, Bashundhara, Dhaka 1229, Bangladesh (e-mail: amirulashraful@gmail.org).

Hong Yan is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China (e-mail: h.yan@cityu.edu.hk)

ISBN: 978-0-9803267-5-8
Figure 1 shows samples from each of the seven languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>不知道是有意刺探</td>
</tr>
<tr>
<td>formation of</td>
<td>慶應義塾大学、</td>
</tr>
<tr>
<td>있다. 조(간기)</td>
<td>ดีที่สุดที่จะไปอยู่ในมาก</td>
</tr>
<tr>
<td>法律への影響力</td>
<td>韓国の影響力</td>
</tr>
</tbody>
</table>

Figure 1. Samples from the seven languages: Arabic, Chinese, English, Japanese, Korean, Tamil, and Thai.

B. Feature Extraction using k-adjacent segments

Our feature extraction process consists of two steps. The first step is edge detection [7] and the second step is to generate K-adjacent segments (K-as).

For our work, we use K-as as our feature. This feature was first introduced by Fevrier et al [8]. We use 3-adjacent segments as 2-adjacent segments pose too much repeatability across different languages and 4-adjacent segments are too complex as we shall see later in this report. We construct a graph \( G(V, E) \), where the nodes \( V \) represent each segment found from the previous step, and the edges \( E \) represent adjacency. Informally, if two segments share a common start or end point, there exists an edge between the segments represented by the nodes. We use depth first search to extract 3 consecutive segments and form 3-adjacent segments.

In order to compare different K-as, we need a numerical descriptor [8]. At the beginning, it is important to order the K-as

\[ \text{descriptor} \{8\}. \]

At the beginning, it is important to order the K-as.

In order to compare different K-as, we need a numerical descriptor [8].

\[ \text{descriptor} \{8\}. \]

At the beginning, it is important to order the K-as.

B. Feature Extraction using k-adjacent segments

Our feature extraction process consists of two steps. The first step is edge detection [7] and the second step is to generate K-adjacent segments (K-as).

For our work, we use K-as as our feature. This feature was first introduced by Fevrier et al [8]. We use 3-adjacent segments as 2-adjacent segments pose too much repeatability across different languages and 4-adjacent segments are too complex as we shall see later in this report. We construct a graph \( G(V, E) \), where the nodes \( V \) represent each segment found from the previous step, and the edges \( E \) represent adjacency. Informally, if two segments share a common start or end point, there exists an edge between the segments represented by the nodes. We use depth first search to extract 3 consecutive segments and form 3-adjacent segments.

In order to compare different K-as, we need a numerical descriptor [8]. At the beginning, it is important to order the K-as segments \( \{S_i\} \), where \( i = 1..k \) in a repeatable manner so that similar K-as have the same order. In the first segment, we select those with midpoint closest to the centroid of all midpoints \( \{m_i = (x_i, y_i)\} \) \( i = 1..k \). In the descriptor below, this centermost segment is the natural choice as reference point for measuring the relative location of the other segments. The remaining segments take up positions 2 through k and are ordered from left to right in accordance to their midpoint. Note that this order is stable as no two segments can have similar location in both \( x \) and \( y \).

Once the order is established, a K-as is a list \( P = (S_1, S_2, ... , S_k) \) of segments. Let \( r_i = (r_{i1}, r_{i2}) \) be the vector going from the midpoint of \( s_1 \) to \( s_i \). Furthermore, let \( \theta_i \) and \( l_i = \|s_i\| \) be the orientation from the K-as center and the length of \( s_i \) respectively.

The descriptor of \( P \) is composed of 4k-2 values for \( k > 1 \).

\[
\frac{r_{21}^2}{N_2}, \frac{r_{21}^2}{N_2}, ..., \frac{r_{k1}^2}{N_k}, \frac{r_{k1}^2}{N_k}, \theta_1, ..., \theta_k, \frac{l_1}{N_1}, ..., \frac{l_k}{N_k}
\]

(1)

The distance \( N_k \) between the two farthest midpoints is used as normalization factor which makes the descriptor scale-invariant (Hence, both the K-as features and their descriptors are scale-invariant).

In this part, we present a measure \( D(a, b) \) of the dissimilarity between two K-as \( p^a \) and \( p^b \) of the same complexity \( k \), originally proposed by Fevrier et la [8].

\[
D(a, b) = w_r \sum_{i=2}^k \|r_{ai} - r_{bi}\| + w_\theta \sum_{i=1}^k D_\theta(\theta_i^a, \theta_i^b) + \sum_{i=1}^k |\log (l_i^a / l_i^b)|
\]

(2)

The first term is the difference in the relative locations of the segments. \( D_\theta \in [0, \pi] \) measures the difference between segment orientations, and the last summation accounts for the difference in lengths. As segment lengths are often inaccurate, we give higher weight to the two other terms of the comparison measure: in all our experiments \( w_r = 4, w_\theta = 2 \).

To illustrate the steps in our methodology in finer details, we will use a sample image of a character and generate 3-adjacent segments from it. For better understanding, we will use the image of a character-like object rather than a character. We start the process of line fitting. Canny edge detection is used to detect roughly straight line segments and break the image structure into a combination of lines. This is demonstrated in the following image.

Figure 2. Sample character and line fitting

We now detect 3 adjacent line segments. The above image contains 5 straight lines. From these 5 lines, we can obtain the triplets as such: \( \{1, 2, 3\}, \{1, 3, 4\}, \{3, 4, 5\}, \{2, 3, 5\}, \{2, 3, 4\} \) and \( \{1, 3, 5\} \).

C. Feature Selection

After extracting contour segments, each K-as segment is represented in a 10-dimensional vector. In our set of languages, it is natural for different languages to share the same set of K-as features. We use principle component analysis (PCA) [9, 10], a statistical tool, is used to extract out discriminating features and use them in our classifier. PCA allows us to project high dimensional data to a lower dimension. The principle dimension is the axis on which the largest number of data points can be discriminated. As subsequent orders of dimension are included, more of the data points can be discriminated. In our experiment, we have found the first 3 principal components to be most effective. We will discuss this in further detail in the results section.

We have used 300 training samples for each language. For 7 languages, we have a feature matrix of 2100 by 10. By applying PCA, we reduce this matrix to 2100 by 3.

D. Classification

Artificial neural networks are used to classify languages. From our previous research work [11], we have observed that probabilistic neural networks train faster and fit a better model compared to other popular neural network models. Moreover, it is experimentally reported that probabilistic neural networks are well known to converge [12] on large training sets.

For our simulation, we have designed our architecture with 7 probabilistic neural networks. Given a single set of network
parameters such as weight and bias, it is more time consuming and difficult to draw decision boundaries that separate 7 different classes. We have used the modular neural network approach [13] motivated by the flexibility it offers to an individual network. Each of the networks only needs to classify two classes – a positive class and a negative class. Each neural network learns a single language. Given an input, all 7 networks respond with a confidence value. The trained language of the network that responds with the highest value is considered to be the language of the input image.

Features from a total of 49 images and 300 segments from each image are collected. This results in a feature matrix of size 300 by 10. To represent a language in a vectorized format of 1 by 3000 and to train it in a neural network with other samples of the other languages is an arduous task. We have tackled this by arguing that several segments occur repetitively across different languages and we can reduce the redundancy in the features by projecting them onto a different space such that the most discriminating dimensions remain. We achieve this by using PCA. After applying PCA, the feature matrix for one image has been reduced to 300 by 3. This gives us a handier feature size of 1 by 900. We collect more of these feature vectors and train our networks for each of the languages. Ten feature samples from each language have been trained with a total of 70 feature vectors. Concretely, our set of feature training matrix is 70 by 900.

III. RESULT AND ANALYSIS

At the beginning of this report, we argue that the 3-as features which have been generated capture the inherent pattern in a language. To prove this, we run experiment using different sizes of samples for the training phase.

![Figure 3](image_url)

**Figure 3.** Accuracy plot against number of training samples extracted from each language

The above plot shows the trend in accuracy with varying number of samples used for training. The accuracy begins to level off after 250 samples for each language. The best accuracy is obtained at 400 samples for each language. However, generating 400 samples requires exponentially longer time than extracting 300 features, hence we settle with 300 features for an optimum trade off between accuracy and time. The following plot shows the time required for feature generation against number of features generated.

![Figure 4](image_url)

**Figure 4.** Plot for time required versus number of feature samples generated from a document image

The above plot demonstrates the trend in time required to extract 3-adj segments from a document image. This trend is quite expected if we observe that the 3-adj segments are generated from a graph whose size depends on the number of segments found in the document. We have experimentally observed that for 300 3-adjacent segments, a graph of 3500 by 3500 should be generated. This is a large graph and running repeated depth-first search to extract 3-adjacents segments on this is a computationally intensive task. This task is increased in many folds for a larger number of 3-adjacent segments such as 400 which requires a much larger graph to be traversed.

A. Number of Principal Components

When projecting using PCA, we have to select the number of dimension to project on. The appropriate number of choice is dependent on the data being projected, and can be determined experimentally. We have conducted several trials of our experiment with varying number of principle components. The corresponding accuracies are demonstrated in the plot below.

![Figure 5](image_url)

**Figure 5.** Changes in accuracy on varying number of principle components

Figure 5 shows that the classifier has consistent learning performance, starting from the first 3 principle components to the 6th principle component being used. We deduct that the 4th, 5th and 6th component do not add any significant discriminating feature information to the classifier and hence no drastic changes in accuracy are observed. Adding the 4th component results in a very slight improvement in accuracy, but we settle with the 3rd component in lieu of reduced computation time and cost. When the 7th component is added, there is a drop in accuracy to 60.8%. Further addition of components leads to a consistent accuracy around 60%. We predict that adding further dimensions to our feature causes the classifier to be fed with
misleading data and creates confusion for the classifier. The projection on the 7th dimension and further contain an overlap of the classes which makes it difficult for the classifier to perform well. To confirm our prediction, we choose different numbers of principle components for classifying them.

The second most confusing classification is caused when classifying Chinese documents from Japanese ones if the Japanese documents consist mainly of Kanji character sets. This can be explained by the fact that Japan borrowed Chinese characters to form her own Kanji character sets which are similar to the Chinese characters.

The best classification is for Arabic at 93.8%. Arabic is visually very different from all the other languages which avoid confusion with other documents and results in a much better classification when compared to other languages. From the confusion matrix, the mean diagonal is 80.5% which we report as our average accuracy.

C. Cross Validation

The above plot shows the 10-fold validation for our system. It shows that the performance of our system is consistent. The training accuracy is higher than the testing accuracy. We attribute this to the nature of probabilistic neural networks, whose design allows them to fit a model much better during the training phase than compared with the testing phase. We intend to explore this variation in our future work.

IV. CONCLUSION

We have proposed the use of 3-adjacent segments for language classification with the use of PCA for feature reduction and a modular neural network design for classification. The results show a consistent performance at an average accuracy of 80.5%. The strength of our system is that it uses a feature set which captures the visual arrangement in a language’s character set, rather than the character information. It is easy to extend the system for further languages. The accuracy found is for languages that share a great deal of visual similarity. For languages which are dissimilar to each other to a great extent, we have found accuracy is greater than 90%. There is no report of any time measure in this area of research. We have reported the times of our system’s training phase for a varying number of samples and believe that this is smaller than those in other reported works since our feature extraction phase samples a defined number of features rather than processing the whole document.

![Figure 6](image_url)

**Figure 6.** Accuracy plot against different choices of principle components

The above plot shows the obtained accuracies with different choices of principle components. It is established that components 1, 2 and 3 are the strongest in their discriminating power, whose performance measure is added to the plot for reference. We observe the accuracy by adding the 7th, 8th and 9th component to see the effect on the discriminating power of the first 3 components. The results support our prediction that the projection on 7th component results in confusion and overlapping of classes for the classifier. Adding the 8th and the 9th component to the first 3 principle components comparatively reduces the accuracy less. Using only the 7th, 8th and 9th component results in a low accuracy of 47.9%. Since the 8th and 9th component are weaker than the first 3 components, the effect of adding the 7th component is more pronounced.

![Figure 7](image_url)

**Figure 7.** 10-fold validation for the system

The above plot shows the 10-fold validation for our system. It shows that the performance of our system is consistent. The training accuracy is higher than the testing accuracy. We attribute this to the nature of probabilistic neural networks, whose design allows them to fit a model much better during the training phase than compared with the testing phase. We intend to explore this variation in our future work.
REFERENCES


