VQ-BASED WRITTEN LANGUAGE IDENTIFICATION

Tuan Pham[1] *and Dat Tran*[2]

[1] School of Computing and Information Technology
Nathan Campus, Griffith University, QLD 4111, Australia
[2] School of Information Sciences and Engineering
University of Canberra, ACT 2601, Australia

ABSTRACT

Humans can recognize different types of written languages by their grammars and vocabularies. However, computers see everything as numbers. We present a computational algorithm for machine classification of written languages using the method of vector quantization. For a language document, each word is converted to a sequence of numbers and forms as a vector of numerical values according to its characters. This collection of vectors is then represented by a codebook that contains a number of template vectors for classification. The proposed method is more effective for machine learning than the n-gram based method, which has been widely used for written language identification. Experimental results of classifying a set of five closely roman-typed scripts show the promising application of the proposed method.

1. INTRODUCTION

Language identification from text is one of important research areas in multimedia information processing that provides computer methods for handling large volumes of electronic data automatically [5]. The sources of text documents may come from email messages, web pages, and other electronic archives. It can be difficult for a human to sort out electronic documents of several different languages so that they can be placed in

appropriate categories for further action and analysis. For instances, an optical character recognition system needs to know the language of the document before performing the decoding process or conversions [11]; an employee can be greatly assisted by knowing the types of the languages of the documents in order to transfer them to appropriate people who understand the languages for further processing. Therefore work performance can be much effective when one has to deal with thousands of electronic documents.

Most methods for written language identification are based on the n-gram approach [4, 14, 11]. In general, the n-gram based method compares the successive n-grams derived from the text with a database of n-gram sets generated for each language. Given an unknown document, the match for each language is obtained by comparing the n-gram frequency profile of the unknown document against the profile of each language using a distance measure. The unknown document is then classified to the language that has the smallest distance. In [12], a written language identification using a trigram-based method is developed and patented. However, no results are presented in this work. A language classification using an n-gram based approach is also presented in [4], and best results are obtained when using the n-gram frequency of 400. It is also indicated that documents of longer in length work a little better than those of shorter in length, and in some cases, the use of more n-gram frequencies decreases the classification performance. Other related works can also be found in references [1, 6, 8].

The main principles for a language identification system is that it must be fast for real-time processing, efficient, requires minimum storage, and robust against textual errors. Based on these principles, the method of vector quantization (VQ) is applied herein because it is an effective method for data compression in speech [10] and image signal processing [3]. VQ can be efficient for language identification because it can process large volumes of training data, and its solution is based on the optimality criteria of the nearest neighbor and centroid conditions. VQ-based methods can speed up classification tasks based on reduced template matching, by which noisy signal can also be tolerable.

2. VQ-BASED IDENTIFICATION

Given a training set $\mathcal{T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$, where each source vector $\mathbf{x}_m$ is of k dimensions. Let N be a given number of codewords and $\mathcal{B} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N\}$ respresents the codebook of size N, where each codeword $\mathbf{c}_y = (c_{y1}, c_{y2}, \ldots, c_{yk})$, $y = 1, 2, \ldots, N$. Each codeword $\mathbf{c}_y$ is assigned to an encoding region R_y in the partition $\Omega = \{R_1, R_2, \ldots, R_N\}$. Then the source vector $\mathbf{x}_m$ can be rep-

0-7803-7946-2/03/$17.00 ©2003 IEEE

resented by the encoding region R_y and expressed by

$$V(\mathbf{x}_m) = \mathbf{c}_y, \text{ if } \mathbf{x}_m \in R_y$$

In general, the VQ design can be stated as follows. Given a training set $\mathcal{T}$, the size N of the codebook, we seek to find the codebook $\mathcal{B}$, and the partition Ω such that the average distortion D is minimized. Using the L_2 norm for a squared-error measure, D is defined by

$$D = \frac{1}{Mk}\sum_{m=1}^{M}(||\mathbf{x}_m - V(\mathbf{x}_m)||_2)^2$$

where $||\mathrm{e}_m||_2$ is the L_2 norm or Euclidean norm

2.1. LBG algorithm

The LBG algorithm [9] requires an initial codebook, and iteratively bi-partitions the codevectors based on the optimality criteria of nearest-neighbor and centroid conditions until the number of codevectors is reached. It is summarized as follows.

1. Given a training data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$, where $\mathbf{x}_m = (x_{m1}, x_{m2}, \ldots, x_{mk}), m = 1, 2, \ldots, M$; and a small real number $\epsilon > 0$

2. Set $N = 1$, compute initial cluster center and average distortion

$$\mathbf{c}_1^* = \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m \quad (1)$$

$$D^* = \frac{1}{Mk}\sum_{m=1}^{M}(||\mathbf{x}_m - \mathbf{c}_1^*||_2)^2 \quad (2)$$

3. Splitting: calculate c_{i1}, c_{i2} then set $N = 2N$

$$\mathbf{c}_{i1} = (1+\epsilon)\mathbf{c}_i^*,\ 1 \le i \le N \quad (3)$$

$$\mathbf{c}_{i2} = (1-\epsilon)\mathbf{c}_i^*,\ 1 \le i \le N \quad (4)$$

4. Iteration: Set $j = 0$ and let $D^j = D^*$

   (a) Assign vector to closest codeword

$$V(\mathbf{x}_m) = \mathbf{c}_y^* = \arg\min_y(||\mathbf{x}_m - \mathbf{c}_y^j||_2)^2 \quad (5)$$

   (b) Update cluster centers

$$\mathbf{c}_y^{j+1} = \frac{1}{|V(\mathbf{x}_m)|}\sum_{\mathbf{x}_m \in V(\mathbf{x}_m)}\mathbf{x}_m,\ 1 \le y \le N \quad (6)$$

   $|V(\mathbf{x}_m)|$ is the number of $V(\mathbf{x}_m) = \mathbf{c}_y^*$.

   (c) Set $j = j + 1$, and compute

$$D^j = \frac{1}{Mk}\sum_{m=1}^{M}(||\mathbf{x}_m - V(\mathbf{x}_m)||_2)^2 \quad (7)$$

   (d) Go to step (a) if

$$\frac{D^{j+1} - D^j}{D^{j+1}} > \epsilon \quad (8)$$

   (e) Set $D^* = D^j$, relabel $\mathbf{c}_y^* = \mathbf{c}_y^j$, $1 \le y \le N$

5. Repeat steps 4 and 5 until the desired number of codewords is obtained.

2.2. Algorithm for VQ-based language identification

Given a language document L^q, we first preprocess the document by removing all common characters, and punctuation marks such as commas, columns, semi-columns, quotes, stops, exclaimation marks, question marks, signs, etc. The next step is to convert all the characters into lower cases. These lower-case characters are translated into the corresponding ASCII values. For example, $(a, b, \ldots, z) = (97, 98, \ldots, 122)$. Based on the LBG-VQ approach, the codebook is designed for L^q given N using this training set. The identity of this template codebook is then used for identifying an unknown pattern. The training and classification procedures are summarized as follows.

2.2.1. Training

1. Given a textual document L of language q: $L^q \in \mathcal{C}$, where $\mathcal{C}$ is the universe of languages.

2. Remove all common characters from L^q to obtain a set of word $\mathbf{W}^q = \{\mathbf{w}_1^q, \mathbf{w}_2^q, \ldots, \mathbf{w}_G^q\}$, where $\mathbf{w}_g^q = (w_{g1}^q, \ldots, w_{gb}^q)$, $1 \le b \le B$.

3. Map each word of b characters, to a vector of numbers of fixed size M:
   $f : |\mathbf{w}_g^q| = b, \mathbf{w}_g^q \in \mathcal{C} \rightarrow |\mathbf{w}_g^q| = M, \mathbf{w}_g^q \in \mathbf{Z}$, $b \le M, \forall b$, if $b < M$, $w_{b+1}^q, \ldots, w_M^q = 0$.

4. Build a codebook $\mathcal{B}^q$ of N codewords $(\mathbf{c}_1^q, \mathbf{c}_2^q, \ldots, \mathbf{c}_N^q)$ for $\mathbf{W}^q \in \mathbf{Z}$

2.2.2. Classification

1. Given a textual document of an unknown language L.

2. Do steps 2 and 3 for L as described in the training phase to obtain a set of vectors $\mathbf{W} \in \mathbf{Z}$.

Table 1. Classification rates (%) – validations $|\mathcal{B}| = 128$

#words	English	French	Spanish	Italian	German
1	88.4	51.4	66.0	63.5	63.4
2	100	64.6	69.3	75.8	76.4
3	100	74.1	74.2	79.7	85.4
4	100	77.5	80.3	85.2	89.8
5	100	86.7	86.0	89.1	92.5
6	100	87.0	88.7	89.5	96.6
7	100	89.6	89.1	92.1	99.0
8	100	86.9	94.1	96.3	97.6
9	100	91.9	95.5	95.8	98.6
10	100	92.2	93.7	96.5	98.5

Table 2. Classification rates (%) – tests ($|\mathcal{B}| = 128$)

#words	English	French	Spanish	Italian	German
50	100	100	100	100	95.0
60	100	100	100	100	85.0
70	100	100	100	100	100
80	100	100	100	100	100
90	100	100	100	100	100
100	100	100	100	100	100

3. Calculate the average minimum distance between $\mathbf{W}$ and $\mathcal{B}^q, q = 1, \ldots, Q$, where Q is the number of languages:

$$d_q = \frac{1}{N} \sum_{y=1}^{N} \min[d(\mathbf{W}, \mathbf{c}_y^q)] \tag{9}$$

where $d(\cdot)$ is defined in (2).

4. Assign L to the language q that has the minimum distance:

$$q^* = \arg\min_q (d_q) \tag{10}$$

3. EXPERIMENTS

We test the proposed algorithm using a set of selected five most closely roman-typed languages: English, French, Italian, Spanish, and German. After removing the special characters, the data sets for English, French, Italian, Spanish, and German consist of 618, 795, 869, 815, and 690 words respectively. We extract the first 518, 695, 769, 715, and 590 words as the training sets for English, French, Italian, Spanish, and German respectively. Each word is translated to a vector of 25 ASCII values (M=25) as an estimate to

Table 3. Classification rates (%) – validations $|\mathcal{B}| = 256$

#words	English	French	Spanish	Italian	German
1	99.4	86.6	66.3	72.2	85.1
2	100	90.4	87.0	91.3	94.4
3	100	95.7	91.9	94.3	98.0
4	100	96.5	96.7	96.1	99.5
5	100	97.7	98.3	97.4	98.6
6	100	98.6	99.3	98.0	100
7	100	98.3	100	98.4	100
8	100	97.9	100	98.2	100
9	100	98.8	100	98.9	100
10	100	100	100	98.8	100

Table 4. Classification rates (%) – tests ($|\mathcal{B}| = 256$)

#words	English	French	Spanish	Italian	German
50	100	100	100	100	95.0
60	100	100	100	100	85.0
70	100	100	100	100	100
80	100	100	100	100	100
90	100	100	100	100	100
100	100	100	100	100	100

Table 5. Confusion matrix for five-word validation

→	English	French	Spanish	Italian	German
English	129	0	0	0	0
French	1	169	1	2	0
Spanish	0	2	175	1	0
Italian	2	0	2	187	1
German	0	1	1	0	145

Table 6. Confusion matrix for fifty-word test

→	English	French	Spanish	Italian	German
English	20	0	0	0	0
French	0	20	0	0	0
Spainish	0	0	20	0	0
Italian	0	0	0	20	0
German	0	0	1	0	19

Table 7. Average classification rates (%)

Method	English	French	Spanish	Italian	German
n-gram	98.5	90.0	81.5	88.2	100
VQ (128)	100	100	100	100	96.7
VQ (256)	100	100	100	100	96.7

accomodate the longest word. If the number of characters in a word is less than 25, then it is padded with zero values to ensure a fixed size M. The last 100 words of each document are used as test sets. A codebook size of 128 codevectors is designed for each language using the LBG-VQ algorithm. Validations are carried out by classifying the languages with small numbers of words ranging from 1 to 10. The validation results for the five languages are given in Table 1, where the recognition rates for English is found to be the highest. To get a good balance between the amount of data for training and testing, we select 20 sets for each 50, 60, 70, 80 words; 11 sets for each 90 words, and 1 set for each 100 words among the test sets of the five languages to carry out the testing. The recognition rates for test sets are given in Table 2 where the percentage average rates for English, French, Spanish, Italian, and German are 100, 100, 100, 100, and 96.7 respectively.

Another codebook size of 256 codewords is also built for each language. Similar validations are carried out by classifying the languages with small numbers of words ranging from 1 to 10. The validation results for the five languages are given in Table 3, Classification rates for French, Spanish, Italian, and German are 96.0%, 93.9%, 94.4%, and 97.6% respectively. As in the case of the German text, an anomaly can be seen in the recognition rates for tesing the sets of 50 and 60 words in which the performance of the later set (ten more words) is lower than the first set having less words. This anomaly has also been pointed out in [4]. The recognition rates for these test sets are given in Table 4, which are the same as for the codebook size of 128. Typical confusion matrices for 5-word validations, and 50-word tests are given in Table 5, and Table 6 respectively. The average results from testing the n-gram based method using 100 n-gram frequencies are 98.5%, 90.0%, 81.5%, 88.2%, and 100.0% for English, French, Spanish, Italian, and German respectively. The average recognition rates of the n-gram, and VQ-based methods for less than and equal to 100 words are shown in Table 7. The VQ-based method using both codebook of sizes 128 and 256 gives the same results when the input sizes are reasonably sufficient.

4. CONCLUSIONS

We have presented a new approach for written language identification based on the designs of vector quantization. Unlike the conventional n-gram based methods that use frequency statistics, the proposed approach models the language problem with multidimensional template matching which is more convenient for machine learning and classification. The experimental results have shown the useful application of the VQ-based method. It is worth investigating other classification methods for written language identification such as neural networks [6] and support vector machines [2, 7] using the VQ codebooks for training.

5. ACKNOWLEDGEMENTS

This work was carried out when the first author was with DSTO. The second author would like to acknowledge the support of the University of Canberra Research Grant

6. REFERENCES

[1] K. R. Beesley, Language identifier: A computer program for automatic natural-language identification on on-line text, *Proc. 29th Annual Conference of the American Translators Association*, pp. 47–54, 1988.

[2] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2:2 (1998) 121-167.

[3] V. Castelli, L.D. Bergman, editors, *Image Databases*. Wiley, New York, 2002.

[4] W.B. Cavnar, and J.M. Trenkle, N-gram-based text categorization, *Proc. 3rd Annual Symp. Document Analysis and Information Retrieval*, pp. 161-175, 1994.

[5] R.A. Cole, J. Mariani, H. Uszkoreit, G.B. Varile, A. Zaenen, A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology*. Cambridge University Press, 1998.

[6] V.J. Hodge, J. Austin, A comparison of a novel neural spell checker and standard spell checking algorithms, *Pattern Recognition Letters*, 35 (2002) 2571-2580.

[7] T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, Boston, 2002.

[8] R.E. Kim, Searching for text? Send an N-gram!, *Byte*, May 1988, 297-312.

[9] Y. Linde, A. Buzo, and R.M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Communications* 28:1 (1980), 84-95.

[10] J. Makhoul, S. Roucos, and H. Gish, Vector quantization in speech coding, *Proc. of the IEEE*, 73:11 (1985) 1551-1588.

[11] Y.K. Muthusamy, and A. L. Spitz, Automatic Language Identification, in: *Survey of the State of the Art in Human Language Technology*, eds. R. A. Cole, J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, A. Zampolli. Cambridge University Press, 1998.

[12] J. C. Schmitt, Trigram-based method of language identification, October 1991. U.S. Patent number: 5062143.

[13] P. Sibun and L. A. Spitz, Language determination: Natural language processing from scanned document images, *Proc. Fourth Conf. on Applied Natural Language Processing*, pp. 15-21, 1994.

[14] J.R. Ullman, A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words, *The Computer Journal*, 20:2 (1977) 141-147.