SN Computer Science (2020) 1:293
https://doi.org/10.1007/s42979-020-00308-7

ORIGINAL RESEARCH

Recognition of Off-line Handwritten Uyghur Words Using Bayesian Networks with Grapheme Nodes

Yamei Xu[1] · Zhigang Xu[1]

Received: 9 March 2020 / Accepted: 23 August 2020 / Published online: 4 September 2020

Abstract
This study proposes a new algorithm that constructs a word Bayesian network (BN) framework with grapheme nodes to recognize off-line handwritten Uyghur words. First, we build an Uyghur script grapheme library according to the rules and morphological structure of Uyghur. The library includes main grapheme, affix grapheme, and dot grapheme categories. Second, word images are segmented into grapheme sequences by subjecting the individual strokes to extraction, segmentation, and clustering operations. Then we design specific feature extractors and classifiers for specific graphemes to detect and identify small differences between similar words. Finally, we construct a hierarchical matching model for graphemes, conjoined segments, and words using a discrete BN. The BN infers word categories from grapheme features, calculates the confidence of inference, and integrates the grapheme recognition information and word-formation prior information to obtain the final word recognition results. A word recognition rate of 91.65% is obtained during experiments conducted with a database consisting of 12,500 samples and a total of 58 trained grapheme categories. These results indicate that the proposed algorithm not only provides a high word recognition rate by effectively avoiding character over-segmentation errors, but also employs a small and fully predictable number of training categories, which facilitates strong expansibility.

Keywords Computer application · Text recognition · Uyghur language · Off-line handwritten words · Bayesian network · Grapheme

Introduction

Handwritten text recognition is an important application in the field of pattern recognition because it is crucial for the digitalization of handwritten documents that is required to facilitate the electronic storage and computer analysis of the information contained therein. To this end, significant advances have been made for applications involving Chinese and Latin characters [1–4]. However, relatively little research has been conducted for the recognition of handwritten Uyghur text, and existing studies have mainly focused on the recognition of individual Uyghur characters [5].

Uygur language has 24 consonants and 8 vowels, for a total of 32 letters, as shown in Fig. 1. Uyghur employs a cursive script based on Arabic where the shapes of individual letters depend on where they are connected when combined into words. Therefore, the 32 letters form 128 characters. The unique aspects of Uyghur script is illustrated in Fig. 2, which defines its structural rules according to the following four points. (1) Words are the smallest linguistic units with semantic meaning in the Uyghur language. (2) Uyghur words contain multiple characters written from right to left along an imagined horizontal axis (baseline). (3) Strokes written along the baseline are denoted as major strokes, while the remaining strokes are denoted as subordinate strokes. (4) One or more characters are connected to form conjoined segments. (5) Neither the heights nor widths of these connecting characters are equivalent. (6) The positions of subordinate strokes tend to vary in handwritten Uygur words because subordinate strokes are typically written only after the major strokes have first been written. Therefore, the recognition of handwritten Uyghur text requires an approach that focuses on entire words rather than individual characters.

✉ Yamei Xu
yameixu@126.com

Zhigang Xu
xzg_cn@163.com

[1] School of Computer and Communication, Lanzhou University of Technology, No. 287 Langongping Road, Qilihe District, Lanzhou 730050, Gansu, China

Uyghur letters								
consonants	ن	ب	پ	ت	ي	س	ش	م
	چ	ج	خ	ف	ق	گ	ك	ڭ
	غ	ل	ر	ز	ژ	ۋ	د	ە
vowels	ئا	ئە	ئې	ئى	ئو	ئۇ	ئۆ	ئۈ

Fig. 1 32 Uyghur letters

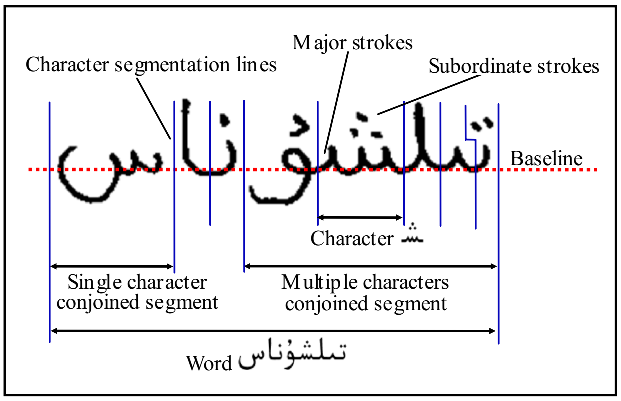

Character segmentation lines
Major strokes
Subordinate strokes
Baseline
Character ـشـ
Single character conjoined segment
Multiple characters conjoined segment
Word تىلشۇناس

Fig. 2 Structural rules of Uyghur script

Currently, the two dominant strategies for recognizing cursive words can be divided according to whether character segmentation is conducted [6], i.e., holistic word recognition [7–10] and segmentation-driven recognition [11–13]. Algorithms that conduct training and recognition of words as a whole are relatively simple, but their ability to discriminate small differences between similar words is comparatively poor. In addition, the expanding ability of these algorithms is limited severely by the very large number of word categories required. In contrast, recognition strategies based on segmentation address the issues associated with holistic word recognition by first dividing the words into characters, and then recognizing words according to the characters with which they are composed. However, the accurate segmentation of Uyghur words into characters represents the most challenging aspect of applying segmentation-driven recognition approaches to this language because Uyghur script often includes touching strokes, broken characters, writing errors, scanning noise, and large variations in handwriting styles. In addition, Uyghur words contain some multi-segment characters, that is, characters that are morphologically similar to a combination of other multiple characters, e.g., س = ـد + ـ + ں. As such, Uyghur script presents over-segmentation problems that cannot be solved by currently available segmentation algorithms, which thereby further affects the results of word recognition.

To avoid unstable character segmentation results, some researchers have proposed using parts of characters, which were called graphemes or primitives, to replace the whole character as the segmentation unit [14, 15]. Moreover, feature design for handwritten word image is another important factor that influences the performance of a handwriting recognition system, because extracting good features from a word image is difficult when the image is degraded or handwriting variability exists in the word. Youssouf Chherawala et al. [16, 17] evaluated a total of five feature sets from four different categories: pixel distribution, concavity, direction distribution and automatically learned, and provided a weighted combination scheme to quantify the relative importance of each feature set, which reflects the strength and complementarity of multiple features including handcrafted features and automated ones.

In light of the above discussion, the present study proposes an algorithm to decompose Uyghur words at the level of graphemes (i.e., at the graphical level of a writing system involving individual characters or character segments) based on the unique features of the Uyghur script, and to recognize words based on the obtained combination of graphemes. Accordingly, a grapheme library of individual Uyghur words is first established, and word images are segmented to form grapheme sequences. Then each grapheme is recognized according to a combination of morphological features and position information. Different feature extractors and classifiers were designed for different grapheme categories based on their individual characteristics. Finally, a grapheme-to-word BN matching model is constructed, and the grapheme recognition confidence and word-formation prior information are integrated to obtain the word recognition results.

Related Work

According to whether character segmentation is performed, the methods of recognizing cursive words can be divided into two strategies: holistic word recognition and segmentation-driven recognition. The algorithms used can be divided into word recognition, character segmentation, and character recognition based on their objectives. The current research status is described from the following three aspects.

Word Recognition

Unlike Latin languages which are only cursive in handwritten text, Uyghur and Arabic are cursive by nature. For both printed and handwritten text, words are written in continuous strokes. Currently, cursive word recognition algorithms are commonly based on the holistic word recognition strategy. Studies have been carried out on the recognition of handwritten Arabic words using a convolutional neural network model (CNN) [10]. In addition, algorithms based on multi-stream hidden Markov model (HMM) [7], support vector machine (SVM) classifier [8], and multi-classifier fusion [9] have also achieved good results in off-line handwritten Arabic word recognition. The holistic approach performs well when the lexicon is predefined, fixed and small in size. However, as analyzed in the preceding section, holistic word recognition algorithms have two disadvantages for Uyghur words: first, they have poor discrimination ability to distinguish small differences between similar words; second, they require training of all word categories, so the algorithm must be retrained when expanding to larger vocabularies.

The segmentation-based recognition strategy can solve the problems that the holistic word recognition algorithm has poor discrimination ability and the algorithm's difficulty to expand. Attempts have been made on cursive text recognition based on the segmentation strategy [11–15]. For off-line Arabic word recognition, Parvez et al. [11] proposed a character segmentation and word recognition algorithm that combines structural features with a fuzzy polygon matching algorithm. Zaiz et al. [12] proposed a character segmentation and word recognition algorithm based on a SVM character classifier and post-processing with a puzzle algorithm. Ahmad et al. [13] presented a multi-stage HMM-based Arabic text recognition system that represents Arabic characters by separating the core shapes from the diacritics and employs a multi-stream contextual sub-core-shaped HMMs as the word classifier.

In the word recognition algorithm based on character segmentation, the number of training categories for word recognition is fixed to the number of character categories, but the accuracy of character segmentation has a great impact on the final word recognition results. Some scholars have sought to find a more suitable segmentation unit than characters. For example, Xu Liang et al. [14] built a meta-stroke library with prior knowledge and presented a Chinese character conceptual model based on stroke relationship learning. Partha Pratim Roy et al. [15] decomposed text lines into character primitives and used string matching to spot words in historical documents.

Therefore, the Uyghur graphemes generated after decomposition were taken as the segmentation unit in this study. The extracted grapheme library contained 58 graphemes. Not only was the number of grapheme training categories less than the number of Uyghur characters, which was 128, but the grapheme segmentation accuracy was also much higher than the character segmentation accuracy because it could avoid the over-segmentation of multi-segment characters and the incorrect segmentation of subordinate strokes discussed in the preceding section, which ensured the effectiveness of the segmentation-based Uyghur word recognition algorithm proposed in this study.

Character Segmentation

The existing character segmentation algorithms are mainly image analysis methods, including an algorithm based on vertical projection [18], an algorithm based on pixel localization [19], and an algorithm based on contour structure [20]. Among them, Al Hamad et al. [18] proposed an Arabic character segmentation algorithm based on vertical differential projection. Elzobi et al. [19] proposed an Arabic character segmentation algorithm based on least-pixel localization and optimal topological structure filtering. Abdelhay Zoizou et al. [20] proposed an Arabic character segmentation algorithm combining template matching and local minima of the contour.

In the existing character segmentation algorithms, the difficulties of Uyghur word segmentation are mainly over-segmentation of multi-segment characters and incorrect segmentation of subordinate strokes. Therefore, in this study, a grapheme was taken as the segmentation unit, and a handwritten Uyghur grapheme segmentation algorithm based on the main stroke over-segmentation and additional stroke clustering was designed, which effectively solved the above problems.

Character Recognition

Compared with word recognition, there are more studies on the recognition algorithms of English, Chinese, and Arabic characters and numbers. Currently, the popular algorithms are those based on neural network or deep learning, which commonly use a convolutional neural network as the classifier. For example, Lamghari et al. [21] proposed a handwritten Arabic character recognition algorithm combining hybrid features with a feed-forward neural network. Chaouki Boufenar et al. [22] investigated the applicability of deep convolutional neural networks using transfer learning strategies.

The word recognition algorithm proposed in this study is based on grapheme segmentation and recognition. Although using a convolutional neural network as the classifier to identify Uyghur graphemes can achieve a high recognition rate, the method is also faced with various problems, such as numerous parameters, and a large consumption of computing resources. The Uyghur grapheme library that we have established has the following characteristics: (1) the main graphemes and the affix graphemes are mostly each composed of a single stroke; (2) the dot graphemes are composed of dots, and have distinct structural characteristics; (3) the dot-connected strokes are two- or three-dot continuous strokes, and the shape of the dot-connected strokes is completely different from that of the unconnected dot strokes. Considering the unique characteristics of the above Uyghur character graphemes, we used different classification methods for the three types of graphemes, achieving a shorter recognition time while obtaining a recognition rate comparable to the convolutional neural network classifier.

Grapheme Library

We divided graphemes from 128 types of Uyghur characters to obtain relatively independent and integrable script components. The graphemes can be divided into the following three categories: (1) main graphemes (MGs), i.e., components of major strokes written along the baseline; (2) dot graphemes (DGs), i.e., combinations of dot strokes; (3) affix graphemes (AGs), i.e., components of subordinate strokes other than DGs. The constructed grapheme library is listed in Table 1, and consists of 46 MGs, 7 DGs, and 6 AGs. As shown in Table 1, MGs, which can be pre-connected, double-connected, post-connected, and stand-alone, refer to the positions of MGs in conjoined segments, and the DGs are denoted according to dotted lines indicative of whether the graphemes lie above or below the baseline. Uyghur words can then be segmented into grapheme sequences according to the constructed grapheme library. This is illustrated in Fig. 3 for the word تىلشۇناس. This word includes the following sequence of characters from right to left: تـ, ـىـ, ـلـ, ـشـ, ـۇ, نـ, ـا, and ـس. The corresponding grapheme composition of the word can then be expressed by the MG sequence ٮـ, ـٮـ, ـلـ, ـٮـ, ـٮـ, ـٮـ, ـو, ٮـ, ـا, ٮـ, ـٮـ, and ـں, by the DG sequence ¨, ∴, and ˙, finally, by the AG ۥ.

Table 1 Grapheme library of Uyghur script, including main graphemes (MGs), dot graphemes (DGs), and affix graphemes (AGs)

MG				DG	AG
Stand-alone	Pre-connected	Double-connected	Post-connected		
ں	ـں	ـٮـ	ٮـ	˙	ء
م	ـم	ـمـ	مـ	¨	ۥ
ح	ـح	ـحـ	حـ	∴	ٚ
ڡ	ـڡ	ـڡـ	ڡـ	.	ٰ
ک	ـک	ـکـ	کـ	..	ـَ
ع	ـع	ـعـ	عـ	:	ۦ
ل	ـل	ـلـ	لـ	∵	
ه	ـه	ـهـ	هـ		
ا	ـا				
و	ـو				
ل	ـل				
ر	ـر				
د	ـد				
ى	ـى				
لا	ـلا				

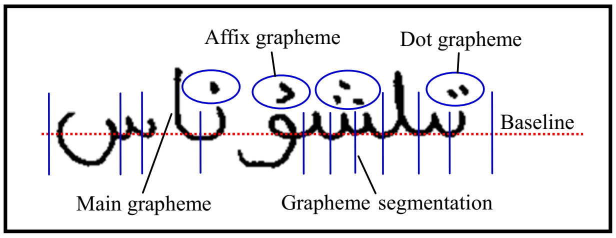


Fig. 3 A example for grapheme composition of the Uyghur word

Decomposing words at the level of graphemes rather than individual characters not only effectively solves the problem of over-segmentation, but also enhances minute differences between similar words, which facilitates the detection and recognition of these differences. For instance, the difference between similar words, e.g., ئېغىر and ئىغىر, is very small, but the DG sequence in the former word is :, ., and ., while that of the latter word is : and ., which are quite different, and the words are readily distinguished based on the classification results obtained using the grapheme classifiers proposed in this paper.

Grapheme Segmentation and Recognition

Grapheme Segmentation

The specific steps adopted for grapheme segmentation of Uyghur words in the present study are given as follows.

Pre-processing and stroke extraction An image of an off-line handwritten Uyghur word is pre-processed using binarization, normalization, broken stroke repair, thinning, slant correction, and correction of conjoined segments, using algorithms presented in a previous work [23]. The set of N' strokes, denoted as $\boldsymbol{S} = (S_1, S_2, \ldots, S_{N'})$, is then extracted through connected domain detection.

Detection of disconnected dot strokes The area of each connected domain $C(S_i)$ is calculated, and a dot stroke is identified based on the criterion $C(S_i) < T$, where T is a preset dot area threshold (i.e., 1/6 of the average stroke area of the training samples). The set of s' disconnected dot strokes is denoted as $\boldsymbol{P} = (P_1, P_2, \ldots, P_{s'})$.

Calculation of the baseline position and baseline domain The Hough transform is applied to the remaining strokes, i.e. $\boldsymbol{S} - \boldsymbol{P}$, and the baseline position B is determined according to local maxima to obtain the baseline domain $[B_u, B_d]$. The baseline domain extraction is conducted according to the ratio of the horizontal projection value of the strokes to the total projection value, denoted as σ, where the factor σ determines the size of the baseline domain. The empirical value of σ obtained in the experiments was 0.7.

Determination of the MG segmentation points Strokes written along B are extracted as a set of strokes, denoted as $\boldsymbol{S}_{\mathrm{M}}$. Then the vertical differential projection [18] of $\boldsymbol{S}_{\mathrm{M}}$ in the interval $[B_{\mathrm{u}}, B_{\mathrm{d}}]$ is calculated, and the intersections of $\boldsymbol{S}_{\mathrm{M}}$ and vertical lines corresponding to local projection minima are taken as the segmentation points. However, because graphemes at the end of conjoined segments, e.g., ں and ى, are likely to be over-segmented, segmentation points are deleted if they lie outside of $[B_{\mathrm{u}}, B_{\mathrm{d}}]$.

Acquisition of MGs The set S_{M} is vertically segmented from the MG segmentation points, and the set of r MGs is denoted from right to left as $\boldsymbol{M} = (M_1, M_2, \dots, M_r)$.

Acquisition of DGs Dot strokes are often written connectedly in handwriting, and Uyghur script usually employs six forms of connectedly written dot strokes, which are given as follows along with the DGs that are conjoined within parentheses: ∽(‥), |(:), ʖ or ∧(∴), and ʔ or ∨(∵). First, connectedly written dot strokes are recognized in the range of the remaining undetermined strokes, i.e. $\boldsymbol{S} - \boldsymbol{P} - \boldsymbol{M}$. These strokes may include ɕ, ʾ, ᵥ, ˌ, ⁻, s, ∽, |, ʖ, ∧ , ʔ and ∨, for a total of 12 categories.

Then the connectedly written dot strokes are established as disconnected dot strokes, and the maximum and minimum algorithm [24] are employed for clustering the set of fully disconnected dot strokes. According to the rule that dot strokes are written on only one side of the baseline, the distance measure between dot stroke P and dot group $\boldsymbol{C}$ is defined as follows:

$$d(P, \boldsymbol{C}) = \begin{cases} +\infty, & [\bar{y}(P) - B][\bar{y}(\boldsymbol{C}) - B] < 0 \\ [\bar{x}(P) - \bar{x}(\boldsymbol{C})], & otherwise, \end{cases} \tag{1}$$

where $\bar{x}(\cdot)$ and $\bar{y}(\cdot)$ are the x-axis coordinate and y-axis coordinate of the image center, respectively. The set of s clustered dot groups is denoted from right to left as DGs, i.e. $\boldsymbol{D} = (D_1, D_2, \dots, D_s)$.

Acquisition of AGs The AGs are formed from all single strokes not denoted as MGs and DGs, and the set of t AGs is denoted from right to left as $\boldsymbol{A} = (A_1, A_2, \dots, A_t)$.

Grapheme matching Each DG or AG is matched to its corresponding nearest MG. If two AGs or two DGs simultaneously match a single MG, the MG will be matched with the nearest AG or DG. Then the other AG or DG will be matched to its second nearest MG, respectively. Figure 4 presents an illustration of the grapheme matching process, where the matching results are DG1 matching MG1, DG2 matching MG5, DG3 matching MG10, and AG1 matching MG4.

Grapheme Recognition

Different feature extractors and classifiers were designed for MGs, AGs, and DGs based on their individual characteristics. These characteristics and the extracted features are defined as follows.

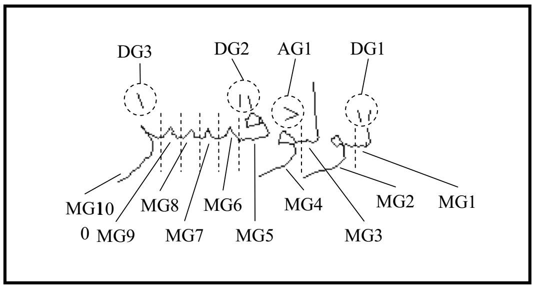


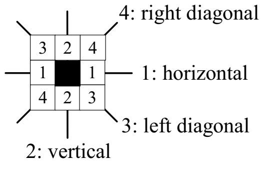
Fig. 4 Illustration of a grapheme matching process

Because the structural features of DGs, e.g., the number and position of dots, are determined and intuitive, the characteristic features were defined as the number of dots $n_d = 1, 2, 3$, the relative position of the dots with respect to the baseline $p_d = 0, 1$, where 0 indicates that the dot lies above the baseline and 1 indicates that it lies below, and the relationship between two dots $r_d = 0, 1$, where 0 indicates a horizontal relationship between two dots and 1 indicates a vertical relationship. The recognition distance of a grapheme $\boldsymbol{x}$ for the candidate of the ith category is denoted herein as $d_i(x)$, and this value can be calculated as follows:

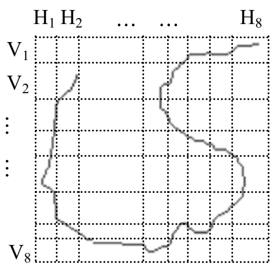
$$\begin{aligned} d_i(x) =& [|n_{\mathrm{d}} - (\boldsymbol{n}_{\mathrm{d}})_i| + 1] \times [|r_{\mathrm{d}} - (\boldsymbol{r}_{\mathrm{d}})_i| + 1] \\ & \times 10^{|p_{\mathrm{d}} - (\boldsymbol{p}_{\mathrm{d}})_i|},\ i = 1, \dots, N_{\mathrm{D}}, \end{aligned} \tag{2}$$

where $(\boldsymbol{n}_{\mathrm{d}})_i$, $(\boldsymbol{p}_{\mathrm{d}})_i$, and $(\boldsymbol{r}_{\mathrm{d}})_i$ express the grapheme features n_{d}, p_{d}, and r_{d}, respectively, for the candidate of the ith category, and N_{D} is the number of categories. Here, $N_{\mathrm{D}} = 7$ for DGs. Considering that the probability of obtaining an erroneous dot position is far less than the probability of obtaining an erroneous number of dots, the first moment of p_{d} was adopted as an exponential term to enhance the effect of dot position features, and the feature distances for n_{d} and r_{d} were both increased by 1 to avoid cases of $d_i(\boldsymbol{x}) = 0$.

The extraction of MGs and AGs conducted in the present work combines the Freeman chain codes in four directions, i.e., horizontal, vertical, left diagonal and right diagonal, with the extraction of elastic mesh directional features (EMDFs) [25]. In particular, we first adopt the pre-processing proposed to normalize the grapheme image. Second, the foreground pixels of the grapheme binary image are detected, and the neighboring points of each pixel are marked in the four directions, as shown in Fig. 5a, so that the four directional features of the pixel are obtained by calculating the number of neighboring points in each direction. Then the image is partitioned into elastic meshes according to the image pixel projection onto the horizontal and vertical directions. The mesh parameters were determined by experiments. Taking into account the area ratios of MGs and



(a) Freeman chain codes in four directions



(b) 8 × 8 mesh of EMDFs for the MG ى

Fig. 5 Feature extraction for MGs and AGs

AGs, 8×8 was set as the mesh size for MGs and 4×4 for AGs. This is illustrated by the 8×8 mesh of the MG shown in Fig. 5b. Finally, the sum of each directional feature for all pixels in each grid is counted to obtain the $8 \times 8 \times 4$ dimension features for the grapheme.

For classification, we note that a compact modified quadratic discriminant function (MQDF) [26] can facilitate the optimal classification of Gaussian samples, and can be used to improve the classification performance obtained with a limited number of samples. Therefore, the compact MQDF classifier was applied to calculate the recognition distance of MGs and AGs. The MQDF is defined as follows:

$$d_i(\boldsymbol{x}) = \frac{1}{\delta_i}\left\{ \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 - \sum_{j=1}^{q}\left(1 - \frac{\delta_i}{\tau_{i,j}}\right)[\boldsymbol{\varphi}'_{i,j}(\boldsymbol{x} - \boldsymbol{\mu}_i)]^2 \right\} + \sum_{j=1}^{q} log\tau_{i,j} + (l - q)log\delta_i,\ i = 1, \ldots, N_M, \tag{3}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\varphi}_{i,j}$ are the mean vector and covariance matrix of the ith category, respectively, which can be obtained by the maximum likelihood estimation during training, while $\tau_{i,j}$ represents the jth eigenvalue of the corresponding eigenvector of $\boldsymbol{\varphi}_{i,k}$. In addition, l is the feature dimension, and q denotes the number of dominant eigenvectors, where $q < l$, and the constant δ_i is a compensation factor. In particular, before using MQDF, the dimension l is usually compressed by linear discriminant analysis (LDA). Then the storage of the original MQDF parameters is further compressed through mapping the parameter space into multiple subspaces by splitting the dimension using a small set of prototypes clustered from the multiple subspaces [26]. In addition, $N_M = 46$ for MGs and $N_M = 6$ for AGs, as discussed with respect to Table 1.

Finally, the number of graphemes included in a sample word and word categories is made uniform by defining an empty grapheme Φ category for each type of grapheme, where the characteristic feature of Φ is an all-zero vector with the same size as the grapheme type to which it belongs, e.g., $8 \times 8 \times 4$ for MGs, $4 \times 4 \times 4$ for AGs, and 3 for DGs. Therefore, we adopted a total of 47 MGs, 8 DGs, and 7 AGs when including Φ categories.

Confidence Transformation

A confidence transformation was applied to the recognition distance $d_k(\boldsymbol{x})$ output by the grapheme classifier to obtain the grapheme recognition confidence $P(\omega_k|\boldsymbol{x})$ of the grapheme $\boldsymbol{x}$ relative to the candidate of the kth category ω_k. The confidence transformation method [27], employs the soft-max function to modify the sigmoid function as follows:

$$P(\omega_k|\boldsymbol{x}) = \frac{exp[-\alpha d_k(\boldsymbol{x}) + \beta]}{1 + \sum_{i=1}^{N_G} exp[-\alpha d_i(\boldsymbol{x}) + \beta]},\ k = 1, \ldots, N_G, \tag{4}$$

where α and β are the confidence parameters whose values are optimized by minimizing the cross entropy (CE) loss function for the training data set [27].

Modeling and Recognition of Uyghur Words

Bayesian Network Modeling of Uyghur Words

A BN, which is also known as a confidence network, is a data model based on probabilistic analysis and graph theory that is used to predict uncertain events. The ability of BN models to synthesize prior information and sample information effectively has led to their increasing application in the field of pattern recognition in recent years [28, 29]. To capitalize on these benefits, the present work constructed a hierarchical matching model between words, conjoined segments, and graphemes using a discrete BN based on the grapheme sequences generated by the segmentation of an Uyghur word. The model was employed to form a relational network graph using individual graphemes as the state nodes, where the directed edges between state nodes represent the relationships and probabilities of state occurrences, and the parent nodes point to the child nodes.

Figure 6 illustrates the structure of the grapheme-based Uyghur word recognition algorithm established in this study, which includes two parts, i.e., the word sample features and the grapheme BN model of Uyghur words. Among these, the sample features are the grapheme feature sequences obtained

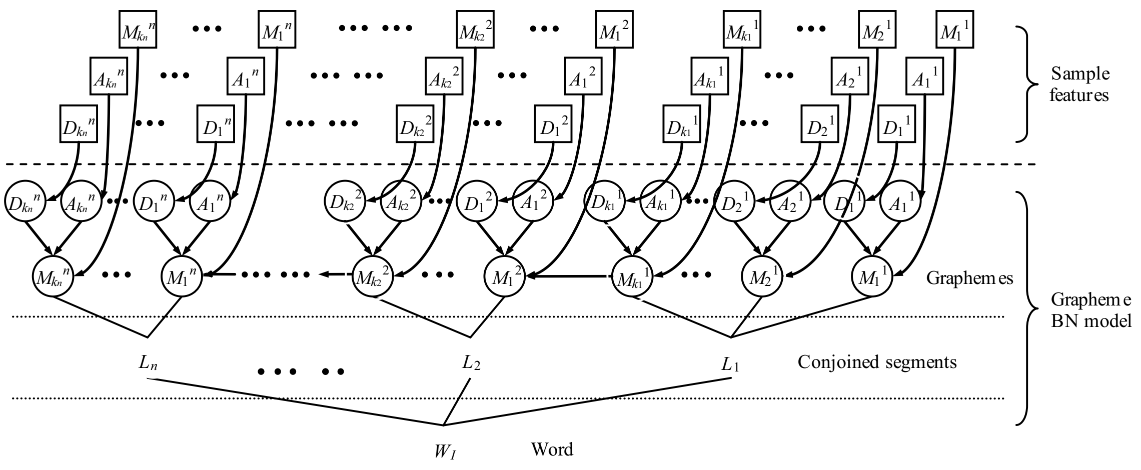


Fig. 6 BN model of Uyghur words

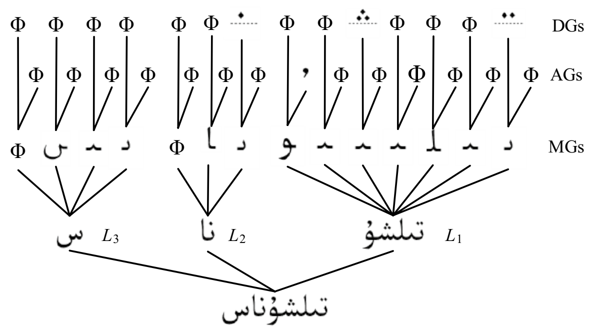


Fig. 7 Structure regularization of the BN model of Uyghur words

after an individual word sample is decomposed into graphemes and the features extracted. The model can be specifically interpreted as follows.

Structure regularization of the BN model The numbers of conjoined segments and graphemes included in a sample word and a word category are usually not equivalent. In this case, the recognition probability for the word category derived from the sample word will not be calculated. Therefore, the empty grapheme Φ is used to expand the number of graphemes contained in the conjoined segments, and regularize the model structure of the sample word and the word category. For example, the word category تلشۇناس in Fig. 7 contains three conjoined segments, i.e., L_1, L_2, and L_3 from right to left, where each segment contains 7, 2, and 3 MGs, respectively. This is denoted as a (7, 2, 3) structure. However, if a sample word is (7, 3, 4) structure, Fig. 7 shows that including the Φ category allows the (7, 2, 3) structure of the word تلشۇناس to be modified into a (7, 3, 4) structure.

State nodes The circular boxes in Fig. 6 represent the state nodes of the BN model. The state nodes can be divided into MGs, DGs, and AGs, where any individual word is an ordered combination of graphemes. As shown in Fig. 6, the word W_I includes the conjoined segments $L_1, L_2, \ldots, L_n$, among which the conjoined segment $L_i, i = 1, \ldots, n$ includes MG nodes, where each MG node M_j^i, $j = 1, \ldots, k_i$ corresponds to two parent nodes, i.e., DG node D_j^i and AG node A_j^i. Because the conjoined segments in the word are divided by only 7 categories of MGs (i.e., ا, ۋ, ـل, ر, د, ى, and ﻻ), the state transition relationships between MG nodes at the convergence of adjacent conjoined segments was taken into consideration.

State transition probability The solid arrows between the state nodes in Fig. 6 represent the state transition probabilities, with parent nodes pointing to child nodes. The state transition probability can be divided into two categories: (1) the transition probability representing the compositional relationship of the graphemes, which is denoted as $P(M_j^i|D_j^i)$ from DGs to MGs, and $P(M_j^i|A_j^i)$ from AGs to MGs, $i = 1, \ldots, n$, $j = 1, \ldots, k_i$; (2) the transition probability indicating the correlation between the graphemes, i.e., the transition probability between MGs at the convergence of adjacent conjoined segments, which is denoted as $P(M_1^i|M_{k_{i-1}}^{i-1})$, $i = 2, \ldots, n$.

Word sample features The square box in Fig. 6 presents the grapheme features of the sample word, including the features of MG sequence $M_1^1, \ldots, M_{k_1}^1, \cdots, M_1^n, \ldots, M_{k_n}^n$, the features of DG sequence $D_1^1, \ldots, D_{k_1}^1, \cdots, D_1^n, \ldots, D_{k_n}^n$, and the features of AG sequence $A_1^1, \cdots, A_{k_1}^1, \cdots, A_1^n, \ldots, A_{k_n}^n$.

Grapheme recognition probability The solid arrows from the grapheme features of the sample word to the state nodes of the graphemes in Fig. 6 indicate the recognition probabilities of the grapheme of the sample word with respect to the grapheme categories, i.e., the grapheme recognition confidences $P(\omega_k|\boldsymbol{x})$, which were obtained according to the aforementioned recognition algorithm.

Word Recognition Algorithm

The proposed BN model was employed for word recognition. The process essentially calculates the recognition confidence values of all grapheme categories from the grapheme features of the word sample, and then outputs the categories according to their recognition confidence in descending order. The complete recognition process includes the learning stage and the recognition stage, which are described as follows.

Learning stage The network topology structure of the BN model is determined by the component structure of the word category, as shown in Fig. 6. However, the parameters of the BN model must be learned from training data. These parameters include the state transition probability and the grapheme recognition probability. The state transition probability includes $P(M_j^i|D_j^i)$, $P(M_j^i|A_j^i)$, $i = 1, \dots, n$, $j = 1, \dots, k_i$ and $P(M_1^i|M_{k_{i-1}}^{i-1})$, $i = 2, \dots, n$, which were learned using maximum likelihood estimation with 15,000 Uyghur words extracted from the Uyghur language corpus as the statistical data. The grapheme recognition probability is the grapheme recognition confidences $P(\omega_k|\boldsymbol{x})$, $k = 1, \dots, N_G$, which were calculated by Eq. (4).

Recognition stage Let $P(W_I|\boldsymbol{X})$ denote the recognition confidence with respect to the categories of Uyghur words W_I, where $I = 1, \dots, N_W$, and N_W is the number of word categories. The $P(W_I|\boldsymbol{X})$ values for the features of a test sample $\boldsymbol{X}$ were calculated according to the network topology structure and parameters of the BN model. We calculate $P(W_I|\boldsymbol{X})$ according to the probability multiplication formula and the conditional independence of the BN as follows:

$$P(W_I|\boldsymbol{X}) = \prod_{i=1}^{N} P(V_i|\mathrm{Pa}(V_i), S^{\mathrm{h}}),\ I = 1, \dots, N_W, \tag{5}$$

where V_i represents the state node associated with the word W_I in the BN model, $\mathrm{Pa}(\cdot)$ is the set of parent nodes of the node V_i, and S^{h} represents the path distribution of $\mathrm{Pa}(\cdot)$. In addition, the following equation for $P(W_I|\boldsymbol{X})$ can be obtained based on the word composition and the topological structure of the BN model:

$$P(W_I|\boldsymbol{X}) = \prod_{i=1}^{n} P(L_i|\boldsymbol{X}) = \prod_{i=1}^{n}\prod_{j=1}^{k_i} P(M_j^i|\mathrm{Pa}(M_j^i)), \tag{6}$$

where the following definitions hold:

$$\begin{aligned} P(M_j^i|\mathrm{Pa}(M_j^i)) =& P(M_j^i|(\boldsymbol{X}_{\mathrm{M}})_j^i)P(M_j^i|D_j^i)P(D_j^i|(\boldsymbol{X}_{\mathrm{D}})_j^i) \\ &\bullet P(M_j^i|A_j^i)P(A_j^i|(\boldsymbol{X}_{\mathrm{A}})_j^i) \\ =& \prod_{G=D,A} P(M_j^i|G_j^i) \prod_{G=M,D,A} P(G_j^i|(\boldsymbol{X}_{\mathrm{G}})_j^i),\ i = 1 \text{ or } j \neq 1 \end{aligned} \tag{7}$$

$$\begin{aligned} P(M_j^i|\mathrm{Pa}(M_j^i)) =& P(M_j^i|M_{k_{i-1}}^{i-1})P(M_{k_{i-1}}^{i-1}|\mathrm{Pa}(M_{k_{i-1}}^{i-1})) \\ &\bullet \prod_{G=D,A} P(M_j^i|G_j^i) \prod_{G=M,D,A} P(G_j^i|(\boldsymbol{X}_{\mathrm{G}})_j^i),\ i \neq 1 \text{ and } j = 1, \end{aligned} \tag{8}$$

where $G = M, D, A$ represents the three categories of graphemes, and $P(G_j^i|(\boldsymbol{X}_{\mathrm{G}})_j^i)$ is the grapheme recognition confidence. Therefore, for a test sample with grapheme features $\boldsymbol{X}$, the preferred recognition result (i.e., the first choice) is the word category corresponding to the maximum posterior probability, which is given as follows:

$$I = \arg\max\{P(W_I|\boldsymbol{X}), I\}. \tag{9}$$

Experiments

The Off-line Uyghur Handwriting Database

Since no benchmark database is available for Uyghur text recognition, this article introduces an off-line Uyghur handwriting database denoted as OUHD (Off-line Uyghur Handwriting Database). The database is used to evaluate the proposed algorithm, as well as for training. It contains a total of 12500 off-line Uyghur handwritten words, included 500 categories. The data collected were written by 25 different native Uyghur writers without restrictions regarding form so as to ensure authentic and practical samples. We divided the data samples in the database into 25 groups according to the writers, each with 500 word samples. Some samples of the OUHD are shown in Fig. 8.

The performance of the proposed grapheme recognition algorithm was evaluated using an Uyghur grapheme

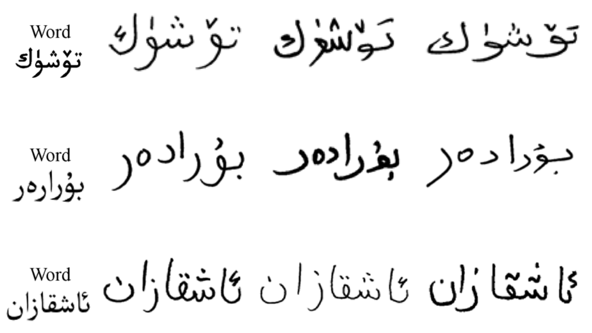


Fig. 8 Samples of words from the OUHD Uyghur handwriting database

database obtained by manually segmenting the data samples in the OUHD. The grapheme database was used for both training and grapheme recognition. The database contains 25 groups, for a total of 113275 grapheme data samples, including 76125 MGs, 9525 AGs, and 27625 DGs. Some data samples are shown in Fig. 9.

The following subsections present analyses of the application of the proposed algorithm to experimental data, and also include a comparison of the results with those obtained by other existing algorithms in the field. However, owing to a lack of existing algorithms for the recognition and segmentation of handwritten Uyghur words, we employed some established algorithms developed for the recognition of handwritten Arabic text and applied them to the OUHD. This is possible because Arabic and Uygur characters are similar in shape, and the 32 basic letters of Uygur and the 28 basic letters of Arabic share 21 equivalent letters. The programming code is single threaded, and was implemented as C++ scripts. All experiments were conducted on a PC with a 2.6 GHz Intel i5-4300M CPU and 4.0 GB of memory.

Grapheme Segmentation Results

The effect of the grapheme segmentation algorithm was quantitatively evaluated based on three criteria: accuracy rate, recall rate, and false detection rate. The accuracy rate refers to the ratio of correct segmentation results to the total number of segmentation points obtained by the algorithm. The recall rate refers to the ratio of the number of segmentation results that have been correctly detected by the algorithm in the correct segmentation position. Finally, the false detection rate is the difference between 1 and the calculated

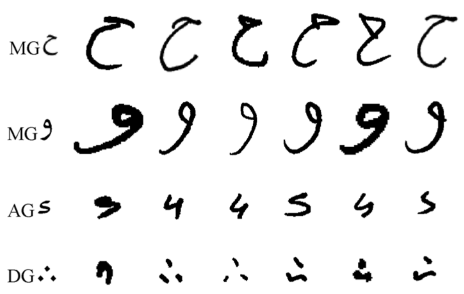


Fig. 9 Samples of graphemes manually extracted from the OUHD

accuracy rate, which includes over-segmentation and bad segmentation, where over-segmentation occurs when a grapheme is divided into more than one grapheme, and bad segmentation occurs when segmentation points incorrectly segment two adjacent graphemes.

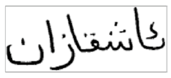
We compared the segmentation results obtained using three grapheme segmentation algorithms on the OUHD. Algorithm 1 is the proposed grapheme segmentation algorithm for handwritten Uyghur text. Algorithm 2 is a segmentation algorithm for handwritten Arabic text based on least-pixel localization and optimal topological structure filtering [19]. Algorithm 3 is a segmentation algorithm for handwritten Arabic text that locates segmentation points by combining template matching and local minima of the contour [20]. The total number of grapheme segmentations for the 500 word categories in the OUHD was 4031, including 25 groups for a total of 100775. Table 2 shows the overall segmentation performance results and the computation times of these three segmentation techniques.

As may be seen from Table 2, our grapheme segmentation algorithm performed very well with an accuracy rate of 98.44%. The false detection rate was only 1.56%, which included 0.84% over-segmentation and 0.72% bad segmentation errors. Comparing the three algorithms, we can see that the over-segmentation error of the proposed algorithm is much less than those of algorithms 2 and 3 owing to the use efficient heuristic techniques, and more importantly, the use of grapheme analysis to cluster dot strokes after stroke separation, which greatly reduces the over-segmentation of DGs. These results are very promising, because correct segmentation is essential for supporting a high grapheme recognition rate.

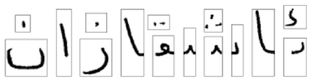
Figure 10 presents an example to illustrate the segmentation results of a word sample using the different segmentation algorithms. The word sample ئاشقازان shown in Fig. 10a consists of 15 graphemes, including the MG sequence ﺪ, ﺎ, ﺪ, ﺴ, ﺴ, ﻌ, ﺎ, ﺮ, ا and ں, the DG sequence ∴, ¨, ˙ and ˙, and the AG ء. The successfully segmented word using the proposed algorithm is presented in Fig. 10b. In addition, the segmentation result of algorithm2 is presented in Fig. 10c for comparison. It can be seen that some graphemes are over-segmented into two graphemes, such as the seventh MG ﺎ, the tenth MG ں, and the first DG ∴. We also present the segmentation result of algorithm 3 in Fig. 10d, where

Table 2 Comparison of the grapheme segmentation performances of three algorithms

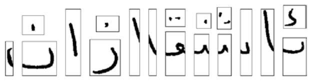
	Accuracy rate (%)	Recall rate (%)	False detection rate (%)			Runtime (ms/word)
			Over-segmentation	Bad segmentation	Total	
Algorithm 1 (proposed)	98.44	99.98	0.84	0.72	1.56	97
Algorithm 2 [19]	84.78	99.08	13.12	2.10	15.22	146
Algorithm 3 [20]	95.33	98.69	3.64	1.03	4.67	85



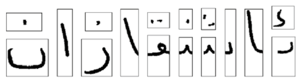
(a) sample word ئاشقازان



(b) segmentation result for algorithm 1



(c) segmentation result for algorithm 2



(d) segmentation result for algorithm 3

Fig. 10 Illustration of grapheme segmentation obtained by the different algorithms

we note the over-segmentation of the DG ۛ. A conjunction of the dot strokes and natural variations in the locations of strokes tend to induce over-segmentation for algorithms 2 and 3 because these algorithms match the dot strokes, rather than the DGs, with the MGs. In addition, algorithm 2 suffers from a high proportion of over-segmentation errors for the MGs because it only considers the topological structure.

Grapheme Recognition Results

The leave-one-out method was adopted for cross validation. Here, one group was selected from the 25 groups of data samples for testing, and the other 24 groups were used for training. Each group was selected for testing in turn, and the average recognition rate was calculated based on the combined results. The performance of the proposed algorithm was evaluated by comparing the experimental results of two recognition algorithms for three types of graphemes. Algorithm 1 was the proposed grapheme recognition algorithm, while algorithm 2 was an off-line handwritten Arabic character recognition method that combines statistical and structural features with an artificial neural network [21]. Table 3 lists the grapheme recognition results and the computation times of the two algorithms.

The results in Table 3 indicate that the proposed algorithm provided better recognition results than algorithm 2. This is because our algorithm uses different recognizers for different types of graphemes. While the recognition rates of the proposed algorithm for MGs and AGs are comparable to those of algorithm 2, the computation times were substantially decreased because the distance classifier has a shorter operation time than the artificial neural network classifier employed in algorithm 2. Moreover, the recognition rate of the proposed algorithm for DGs was 99.99%, which is 1.86% greater than that obtained by algorithm 2 because dot strokes that are conjoined in writing can be effectively recognized when extracting structural features and considering the information of grapheme segmentation, which also greatly decreases the required recognition time.

Word Recognition Results

In the experiments, the recognition performances of five algorithms on the OUHD were compared. Algorithm 1 was the off-line handwritten Uyghur word algorithm proposed in the present study. Algorithm 2 [11] and algorithm 3 [12] are based on segmentation-driven recognition. Here, algorithm 2 is an off-line Arabic handwriting recognition algorithm using structural techniques, which first integrates a segmentation algorithm into the recognition phase and utilizes a polygonal approximation algorithm. Then the segmented character is modeled by fuzzy polygons and later recognized using a fuzzy polygon matching algorithm. Finally, the best hypothesis of a sequence of recognized characters for each word is selected by dynamic programming. Algorithm 3 recognizes an off-line handwritten Arabic word using a set of stages that include segmentation, feature extraction, character classification, and post-treatment. In the character classification stage, a support vector machine (SVM) classifier is used based on two horizontal and vertical scanning masks. In addition, a puzzle algorithm is added as a post-processor. Algorithm 4 [7] and algorithm 5 [8] are based on holistic recognition. Algorithm 4 proposes an

Table 3 Comparison of grapheme recognition performances for two algorithms

	MGs		AGs		DGs	
	Recognition rate (%)	Runtime (ms/grapheme)	Recognition rate (%)	Runtime (ms/grapheme)	Recognition rate (%)	Runtime (ms/grapheme)
Algorithm 1(proposed)	98.28	122	99.87	121	99.99	37
Algorithm 2 [21]	98.27	732	99.82	731	98.13	729

off-line handwritten Arabic recognition algorithm based on asynchronous multi-stream hidden Markov model (HMM), which models the interaction between multiple features composed of a combination of statistical and structural features, which are extracted over the columns and rows of the word image using a sliding window approach. Algorithm 5 is a handwritten Arabic/Persian word recognition algorithm based on combined features and an SVM classifier. Here, the word image is broken into an $m \times n$ window and the features include the angle, number, location, and size of straight lines extracted from each window.

We evaluated the performance of the proposed word recognition algorithm on the OUHD database described in "The Off-line Uyghur Handwriting Database", and adopt the leave-one-out method for conducting cross validation. Here, each group from the 25 groups of data samples was selected in turn for testing with the other groups employed for training, and the average recognition rate was calculated. Table 4 summarizes the word recognition performance of the five algorithms. It can be seen that the algorithm proposed in this study (algorithm 1) provided the highest first choice word average recognition rate of 91.65%, which verifies the effectiveness of the algorithm.

Detailed analysis was conducted by firstly comparing the performances of segmentation-driven algorithms (algorithms 1, 2, and 3) and holistic algorithms (algorithms 4 and 5) based on the results listed in Table 4. The results indicate that the proposed algorithm has the highest first candidate recognition rate, and its recognition rate of the top five candidates is slightly greater than that of algorithm 4. It can be concluded that this is because the proposed algorithm improves the recognition accuracy of similar words by extracting features at the grapheme level to achieve localized recognition of subtle differences between similar words. We verified this conclusion by selecting two groups of two similar words from the 500 categories of the OUHD. The first group of similar words was ئېغىز and ئېغىر, and the second group was ئاپا and ئاچا. We counted the number of samples that were correctly identified as the first choice by the five different algorithms from the 25 samples of each of the above words, and the comparison results are listed in Table 5. We note that the difference in the first group of similar words is the occurrence of DG ـ˙ـ, and the difference in the second group is the occurrence of MG ـد or ـحـ. However, algorithms 2 and 3 must distinguish characters ـز and ـر for the first group and characters ـپ and ـچ for the second group. The fact that algorithms 4 and 5 must identify the entire word clearly illustrates the stronger discrimination ability of the proposed algorithm based on the recognition of graphemes. In addition, the proposed algorithm considers six types of connectedly written dot strokes during the segmentation process, and can, therefore, correctly recognize connectedly written forms of ∵, such as ʔ and v. However, algorithm 3 extracts dot strokes as a pre-classification, and does not consider conjoined dot strokes. Therefore, the recognition rate of algorithm 3 is low for words containing the graphemes ∴ and ∵, such as in the second group of similar words.

Second, a comparison of the word recognition rates obtained by the segmentation-based algorithms 1, 2, and 3 suggests that the proposed modeling based on grapheme decomposition can avoid the over-segmentation of multi-segment characters during the segmentation process, and achieve better recognition performance. Here, multi-segment characters refer to those characters that are easily segmented into multiple graphic segments, such as ـس, which can be segmented into ـد, ـٮـ, and ـٮـ. The over-segmentation of these

Table 5 Comparison of the recognition performances of the five algorithms for similar words

	Number of correct 1st choice identifications in 25 samples			
	Group 1		Group 2	
	ئېغىز	ئېغىر	ئاپا	ئاچا
Algorithm 1(proposed)	25	24	24	24
Algorithm 2 [11]	24	23	23	24
Algorithm 3 [12]	23	23	19	18
Algorithm 4 [7]	21	20	21	22
Algorithm 5 [8]	21	21	18	20

Table 4 Comparison of the Uyghur word recognition performances of five algorithms

	Recognition rate (%)			Number of training categories (recognition strategy)	Runtime (ms/word)
	1st choice	Top 2	Top 5		
Algorithm 1 (proposed)	91.65	93.77	95.12	58 (Segmentation-driven recognition)	624
Algorithm 2 [11]	80.32	83.85	86.34	128 (Segmentation-driven recognition)	922
Algorithm 3 [12]	75.56	78.11	82.61	128 (Segmentation-driven recognition)	783
Algorithm 4 [7]	87.94	90.25	95.08	500 (Holistic recognition)	301
Algorithm 5 [8]	85.03	88.96	93.37	500 (Holistic recognition)	557

multi-segment characters seriously detracts from the performance of the character segmentation algorithm, which then reduces the accuracy of word recognition. Algorithms 2 and 3 determine the over-segmentation of multi-segment characters based in part on the character recognition results. However, this strategy is invalidated for Uyghur characters such as ﺲ. This is because ـٮـ is a graphic segment in Arabic not a character, but, in Uyghur, ـٮـ is not only a graphic segment of the character ﺲ, but is also a middle form of the character ى. Therefore, a character recognizer cannot determine whether ـٮـ is a character or some other graphic element. Therefore, our algorithm reduces over-segmentation errors through grapheme decomposition and modeling (see the results in Table 1), which further increases the word recognition rate.

Third, we note from Table 4 that the numbers of training categories required by segmentation-driven algorithms are small and fixed, which means that these algorithms are readily expanded to the task of large-scale vocabulary recognition. In contrast, holistic algorithms require a relatively large number of categories, and this number increases as the size of the vocabulary increases. Specifically, the number of training categories required by the proposed algorithm are quite small because the algorithm conducts word recognition at the grapheme level, and these categories consist of 46 MGs, 6 AGs (see Table 1), and 6 conjoined dot strokes ∽(··), |(:), ᓚ or ∧(∴), and ʔ or ∨(⸪), for a total of only 58 categories. Algorithms 2 and 3 segment words at the character level; therefore, these require training categories composed of 128 characters. However, algorithms 4 and 5 are based on holistic recognition strategies, such that their required number of training categories are equal to the entire lexicon of the experimental database, which is 500 in this paper.

Finally, the runtime results in Table 4 indicate that the recognition algorithms based on segmentation require more computation time than the holistic recognition algorithms. This is because the segmentation modules employed in segmentation-driven algorithms increase the algorithm complexity to some extent.

Conclusion

The complicated strokes and connected writing employed in handwritten Uyghur text make it difficult to accurately extract the characteristics of the individual characters employed in the writing. Therefore, the algorithm proposed in this study decomposed Uyghur script at the grapheme level, and designed different feature extractors and classifiers for the different graphemes. As a result, the algorithm not only can detect and recognize slight differences between similar words, but is also robust for the detection of complex handwriting features such as the connected writing of dots and stroke deformations. The algorithm calculated the posterior probabilities of the candidate categories using a BN model. The BN model not only accommodated the complex structure and graphic description of Uyghur script, but also integrated semantic information with the grapheme identification process, grapheme compositional structure, and conjoined segments, and thereby effectively increased the word recognition rate relative to presently available word recognition algorithms applicable to similar cursive scripts. In addition, the decomposition of graphemes into a small and fully predictable number of categories greatly reduces the number of categories needed for algorithm training, and the algorithm is readily expandable to Uyghur text recognition tasks involving a large vocabulary.

Funding This study was funded by National Natural Science Foundation of China, NSFC (Grant numbers 61562058 and 61761028).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Impedovo S. More than twenty years of advancements on frontiers in handwriting recognition. Pattern Recognit. 2014;47(1):916–28.
2. Kaur H, Kumar M. A comprehensive survey on word recognition for non-Indic and Indic scripts. Pattern Anal Appl. 2018;21:897–929.
3. Tanzila S, Abdulaziz SA, Amjad R. Online versus offline Arabic script classification. Neural Comput Appl. 2016;27(7):1797–804.
4. Mohammad TP, Sabri AM. Offline Arabic handwritten text recognition: a survey. ACM Comput Surv. 2013;45(2):1–35.
5. Xu Y, Lu Z, Li J. Handwritten Uyghur character recognition based on radical dictionary and time division direction feature. J Jilin Univ (Eng Technol Ed). 2013;43(3):740–6.
6. Tanzila S, Amjad R, Mohamed EB. Methods and strategies on offline cursive touched characters segmentation: a directional review. Artif Intell Rev. 2014;42(4):1047–66.
7. Jayech K, Mahjoub MA, Amara NEB. Synchronous multi stream hidden Markov model for offline Arabic handwriting recognition without explicit segmentation. Neurocomputing. 2016;214(19):958–71.
8. Tavoli R, Keyvanpour M, Mozaffari S. Statistical geometric components of straight lines (SGCSL) feature extraction method for offline Arabic Persian handwritten words recognition. IET Image Process. 2018;12(9):1606–16.
9. Tamen Z, Drias H, Boughaci D. An efficient multiple classifier system for Arabic handwritten words recognition. Pattern Recognit Lett. 2017;93(7):123–32.
10. Ghanim TM, Khalil MI, Abbas HM. Comparative study on deep convolution neural networks DCNN-based offline Arabic handwriting recognition. IEEE Access. 2020;8:95465–82.
11. Parvez MT, Mahmoud SA. Arabic handwriting recognition using structural and syntactic pattern attributes. Pattern Recognit. 2013;46(1):141–54.

12. Zaiz F, Babahenini MC, Djeffal A. Puzzle based system for improving Arabic handwriting recognition. Eng Appl Artif Intell. 2016;56(11):222–9.
13. Ahmad I, Fink GA. Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs. Int J Doc Anal Recognit. 2019;22:329–49.
14. Xu L, Wang Y, Li X, et al. Recognition of handwritten Chinese characters based on concept learning. IEEE Access. 2019;7:102039–53.
15. Roy PP, Rayar F, Ramel J-Y. Word spotting in historical documents using primitive codebook and dynamic programming. Image Vision Comput. 2015;44:15–28.
16. Chherawala Y, Roy PP, Cheriet M. Feature design for offline Arabic handwriting recognition: handcrafted vs automated? In: The 12th International Conference on Document Analysis and Recognition (ICDAR), IEEE, Washington, 2013;290–294
17. Chherawala Y, Roy PP, Cheriet M. Feature set evaluation for offline handwriting recognition systems: application to the recurrent neural network model. IEEE Trans Cybern. 2015;46(12):2825–36.
18. Al Hamad HA, Zitar RA. Development of an efficient neural-based segmentation technique for Arabic handwriting recognition. Pattern Recognit. 2010;43(8):2773–98.
19. Elzobi M, Al-Hamadi A, Al-Aghbari Z, et al. IESK-ArDB: a database for handwritten Arabic and an optimized topological segmentation approach. Int J Doc Anal Recognit. 2013;16(3):295–308.
20. Zoizou A, Zarghili A, Chaker I. A new hybrid method for Arabic multi-font text segmentation, and a reference corpus construction. J King Saud Univ Comput Inf Sci. 2018;. https://doi.org/10.1016/j.jksuci.2018.07.003.
21. Lamghari N, Charaf MEH, Raghay S. Hybrid feature vector for the recognition of Arabic handwritten characters using feed-forward neural network. Arab J Sci Eng. 2018;43:7031–9.
22. Boufenar C, Kerboua A, Batouche M. Investigation on deep learning for off-line handwritten Arabic character recognition. Cognit Syst Res. 2018;50:180–95.
23. Xu Y. A study of key techniques for Uighur handwriting recognition. PhD. thesis, 2013; Xidian University, Chain, Xi'an.
24. Juan A, Vidal E. Comparison of four initialization techniques for the K-medians clustering algorithm. In: Advances in Pattern Recognition, Lecture Notes in Computer Science, Springer, Berlin, 2000;842–852.
25. Jin L, Wei G. Handwritten Chinese character recognition with directional decomposition cellular features. Circuits Syst Comput. 1998;8(4):517–24.
26. Wei X, Lu S, Lu Y. Compact MQDF classifiers using sparse coding for handwritten Chinese character recognition. Pattern Recognit. 2018;76(4):679–90.
27. Wang Q, Yin F, Liu C. Handwritten Chinese text recognition by integrating multiple contexts. IEEE Trans Pattern Anal Mach Intell. 2012;34(8):1469–81.
28. Liu L, Wang S, Su G, et al. Towards complex activity recognition using a Bayesian network-based probabilistic generative framework. Pattern Recognit. 2017;68(8):295–309.
29. Wu Z, Zhang J, Chen K, et al. Yoga posture recognition and quantitative evaluation with wearable sensors based on two-stage classifier and prior Bayesian network. Sensors. 2019;19(23):1–19.