Handwritten Tifinagh Text Recognition Using Fuzzy K-NN and Bi-gram Language Model

Said Gounane and Mohammad Fakir
Faculty of sciences and technology
University Sultan Moulay Slimane
Beni Mellal, Morocco

Belaid Bouikhalen
Multidisciplinary Faculty
University Sultan Moulay Slimane
Beni Mellal, Morocco

***Abstract*—In this paper we present a new approach in Tifinagh character recognition using a combination of, k-nearest neighbor algorithm and the bigram language model. After the preprocessing of the text image, and the word segmentation, for each image character, the k-NN algorithm proposes candidates weighted of their membership degree. Then we use the bigram language model to choose the most appropriate sequence of characters. Results show that our method increases the recognition rate.**

Keywords—Tifinagh character recognition; fuzzy k-nearest neighbor; features extraction; bigram language model.

I. Introduction

The task of handwritten text recognition is challenging and has been a focus of research for several decades [1][3][7]. A human always use his language knowledge when reading a text. Thus the input image is not enough to estimate the most likely characters (or words) sequence, but also the target language [2]. Thus, modern recognizers use a statistical approach in which the returned characters (words) sequence is the one that maximizes the probability with respect to the input image according to a model trained on handwritten text as well as the language according to a model trained on a language corpus.

This paper is organized as follows: Section 2 presents tow techniques used for features extraction, the gravity center distance method and the pixel density method. The fuzzy k-Nearest neighbor and bi-gram language model are presented in section 3. Our complete handwriting recognition process is outlined in section 4. Experimental results and discussions are in section 5, then the conclusion in section 6.

II. Featurs extraction

Pr-processing techniques like word segmentation, thinning, slant correction and smoothening are applied. To extract the image features, we use a linear combination of the gravity center distance method and the pixel density method. Those two methods require the spatial division of the character image into W × H boxes so that the portions of character will be in some of these boxes. There could be boxes that are empty. The choice of number of boxes is arrived at by experimentation.

A. *Gravity center destance*

For each box B, the gravity center ***G*** of black pixels is computed. The feature is then obtained by dividing the distance ***d*** between ***G*** and bottom left corner of the box by the length of the diagonal of that box D "Fig 1".

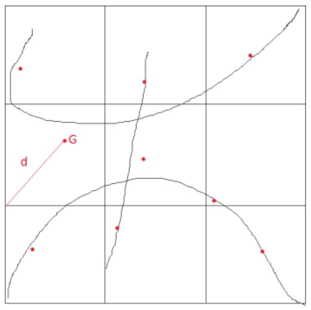


Fig. 1. Gravity center destance method

One can easily see that this characterization is invariant of the character image dimensions. Hence an image of whatever size gets transformed into a vector of W × H predetermined dimensions.

B. *Pixel dencity*

As in the first method, the character Image is divided into W × H boxes. The feature for each box is the number of black pixels divided by the total number of pixels in that box.

The same as the pixel density method, this characterization is invariant of the character image dimensions.

III. Fuzzy k-Nearest Neighbor and Bi-gram language model

In this section, we give a brief presentation of both techniques used in this work, fuzzy k-nearest neighbor and bigram language model.

A. *Fuzzy k-nearest neighbo*

Fuzzy k-Nearest neighbor is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition, image processing and many others[6][8]. Some successful applications are including recognition of handwriting.

It's a very simple technique. It works based on an unsupervised clustering algorithm like fuzzy k-means algorithm to determine prototypes representing each cluster. Then the minimum distance from the query instance to these prototypes are used to determine the K-nearest neighbors. After and basing on membership function, we take the neighbor with the maximum membership degree to be the prediction of the query instance.

From a given labeled examples, and using fuzzy k-means algorithm, we generate a number of prototypes P_{ij} representing each class C_i . In this step, the number of training samples is reduced and the outliers are excluded.

For a given image character X, the k-Nearest neighbor prototypes $N_l \in \{P_{ij}\}, l = 1 \dots k$ are generated (using distance similarity), then the membership degree of X to the class C_i , is calculated as follows [5] :

$$\mu_i(X) = \frac{\sum_{j=1}^{k} \mu_{ij}/d(X,N_j)^{2/(m-1)}}{\sum_{j=1}^{k} 1/d(X,N_j)^{2/(m-1)}} \qquad (1)$$

The term μ_{ij} is the membership degree of N_j to the class C_i . This term is equal to 1 if $N_j \in C_i$, 0 otherwise.

This algorithm has better performances versus the classical k-nearest neighbor algorithm. In the classic approach, the vector X is affected to the class with major prototypes in $\{N_{j=1..k}\}$ with no consideration of how close these prototypes are to X. However in the fuzzy approach, the vector X could be affected to a class that have only one prototype in $\{N_{j=1..k}\}$ just because it's closer enough to X and the other k-1 prototypes are not (fig).

A. Bigram language model

An N-gram model is a type of probabilistic language model for predicting the next item in a sequence. N-gram models are now widely used in probability, communication theory, computational biology, data compression and in statistical natural language processing.

Let $P(w)$ be the probability that a word w (sequence of characters $w = \{c_1, c_2, \dots, c_n\}$ appears in a dictionary. By using the chain rule, this probability can be computed as follows:

$$P(w) = P(c_1) \times \prod_{i=2}^{n} P(c_i/c_1, c_2, \dots, c_{i-1}) \qquad (2)$$

By using the Markov assumption, this probability is approximated by:

$$P(w) \approx P(c_1) \times \prod_{i=2}^{n} P(c_i/c_{i-1}) \qquad (3)$$

Each character depends only on the previous one rather than all the previous characters. This assumption is important because it massively simplifies the problem of learning the language model from data.

$$P(c_i/c_{i-1}) = \frac{\text{count}(c_{i-1},c_i)}{\text{count}(c_{i-1})} \qquad (4)$$

In practice, however, one should smooth the probability distributions by also assigning non-zero probabilities to unseen bigrams [4]. The technique used is called the add-one smoothing, and it's given by the following expression:

$$P(c_i/c_{i-1}) = \frac{count(c_{i-1},c_i)+1}{count(c_{i-1})+v} \qquad (5)$$

Where, v is the number of characters in the used language alphabet.

In order to use the bigram context of the first and the last character in a word, we need to augment this word with a special symbols <s> and </s> at the beginning and the end of the word respectively.

For example, to compute P("□□□□"), we first need to augment this word with <s>and </s>. Then we have:

P ("□□□□") =P (□/<s>).P (□/□).P (□/□).P (□/□).P (□/</s >)

IV. Recognition System

Our recognition system is made of three blocks "Fig. 2": the pre-processing block, the isolated character recognition block and the word recognition block.

First, the word image is pre-processed by thinning; slant correction and smoothening. The word image is then segmented to a sequence of character images.

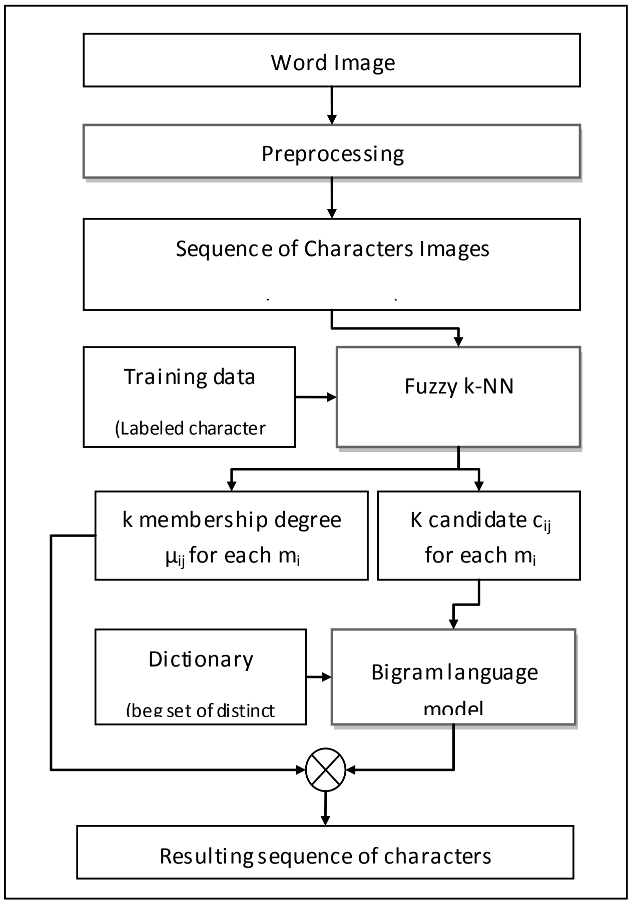


Fig. 2. Recognition system

For each character image the fuzzy k-nearest neighbor proposes candidates, each one with a membership degree $\mu(c_{ij})$ (the j th candidate proposed for the i th character image). From these candidates we generate all m possible combinations $w_{i=1..m}$. The membership degree of each word is given by:

$$\mu(w_i) = \prod_{c \in w_i} \mu(c), \; i = 1 \dots m \qquad (6)$$

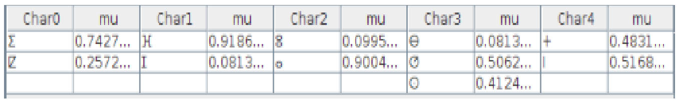
For example, in the word image "□□□□□" the k-NN algorithm propose a set of candidates for each character in the word image "Fig. 3". For the first character "□" tow candidates are proposed, "□" with membership degree 0.742 and "□" with a membership degree 0.258, the decision is clear with no ambiguity "□" because "□" has the higher membership degree. The same case for the second and third character. For the forth character "□" it proposes three candidates "□": 0.081, "□":0.506 and "□":0.412 the decision become ambiguous. The same problem with the last character, it proposes "□": 0.484 and "□":0.516 for the character "□". Using these membership degrees, the fuzzy k-nearest neighbor algorithm gives the character sequence "□□□□□" as result, instead of "□□□□□".

In the previews stage of our recognition system, the k-nearest neighbor algorithm misrecognizes some ambiguous characters. To avoid this problem we use the bigrams to select the most appropriate word as follows:

$$\widetilde{w} = argmax_{w\epsilon\{w_{i=1..m}\}}(P(w).\mu(w)) \qquad (7)$$

A. Algorithm

```
For each word image w
    For each c in w
        Using fuzzy k-NN algorithm, Generate
        candidates and their membership degrees
    End for
    Generate all m possible combinations $w_{i=1..m}$.
    For each word $w_i$
        Compute $\mu(w_i)$ using (1) and (6)
        Compute $P(w_i)$ using (3)
    End for
    The selected sequence of characters
    $\widetilde{w}$ is given by (7)
End for
```



Char0	mu	Char1	mu	Char2	mu	Char3	mu	Char4	mu
ⵎ	0.7427...	ⵅ	0.9186...	ⵓ	0.0995...	ⵟ	0.0813...	ⵜ	0.4831...
ⵣ	0.2572...	ⵊ	0.0813...	ⴰ	0.9004...	ⵕ	0.5062...	ⵏ	0.5168...
						ⵔ	0.4124...		

Fig. 3. Proposed candidates for each character image and thier membershep degrees.

V. Experimental Results

In order to test our approach, we made up a java desktop application. The data base used to train the fuzzy k-nearest neighbor was made up of 1940 handwritten characters image. The dictionary used in the bigram model was a set of 1059 different Amazigh words written in Tifinagh. In addition to the character images database, and the dictionary size, Results obtained depends on many other parameters: the features extraction method, the size of the grid used in the features extraction $W \times H$, the number of neighbors k and the fuzziness parameter m.

A. Features extraction

In the features extraction stage we divide the character image into $W \times H$ boxes. Since most Tifinagh characters are symmetric with respect to x or y axes or both (□, □), it's more appropriate to use an odd numbers for both W and H. In addition, if W and H are too high we lose generality and we do more computation, and if they are too low we lose precision and many deferent characters will be seen as one. Table 1 shows the recognition rate for deferent choice of W and H.

TABLE I. RECONITION RATE FOR DIFERENT GRIDE SIZES

	Recognition rate					
	3x3	*3x5*	*5x5*	*5x7*	*7x5*	*9x11*
FKNN	80.87	85.24	88.73	86.99	88.26	81.88
FKNN & Bigrams	84.20	86.64	91.05	89.31	89.66	85.48

The gravity center distance method gives information about the position of the black pixels in the box. Therefore, tow boxes with deferent amount of black pixels but with the same gravity center will have the same feature value. In the other hand, the pixel density method gives information about the amount of the black pixels in a box. The weakness of this method is that it gives no information about the position of black pixels in that box. So, tow boxes with same amount of black pixels will have the same feature value, even if the black pixel distribution is not the same.

In order to take advantage of both methods we used a linear combination of these two methods. So if v_d is the feature given by the pixel density method and $v_g$the one given by the gravity center distance, the used feature is given by the following expression:

$$v = \gamma v_g + (1-\gamma)v_d \quad 0 \leq \gamma \leq 1 \qquad (8)$$

"Fig. 4" shows that the combination of pixel density and gravity center distance method ($\gamma = 0.02$) gives better results than using only one method ($\gamma = 1 or\ 0$). We can also see that, for all values of γ, the k-nearest neighbor combined to the bigram model (solid curve), produces higher performances than the only use of the k-nearest neighbors algorithm (dashed curve).

B. Neighbors number k and the fuzziness factom m

The k-nearest neighbor algorithm is based on the equation (1). This equation contains tow parameters that affects the accuracy of the classifier, these parameters are the number of nearest neighbor k and the fuzziness factor m. High values of k will produce large number of candidates with small membership degrees. This big number of candidates leads to a huge number of combinations, and then, some of these combinations could have a higher probability even if they are far away from the real word image "Fig. 5". In the other hand if k is too low, the k-nearest neighbor will not propose enough candidates to be used by the bigram model "Fig. 5".

"Fig. 6" shows that, if the m parameter is too large all proposed candidate will have approximately the same membership degree and the decision will be made by the bigrams.

An example that illustrates an inappropriate choice of k and m is shown in “Fig. 7”. Where k=11, m=3 and the word image “□□□□□□□”, for each character image, the membership degrees, of proposed candidates are all between 0.08 and 0.38, the number of possible words is 72576. The result given by the fuzzy k-nearest neighbor algorithm was “□□□□□□□” and “□□□□□□□” by the combination of fuzzy k-NN and bigram model. In both cases tow characters were misrecognized, the difference is that the second one “sounds” to be a real Amazigh word “□□□□□□□”.

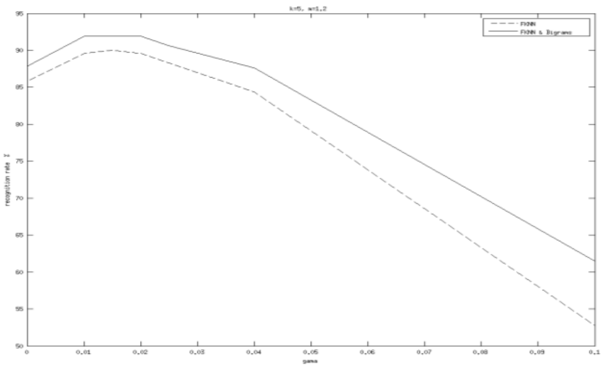


Fig. 4. Recognition rate function of γ, Fuzzy KNN & bigrams in solide line, FKNN in dashed line

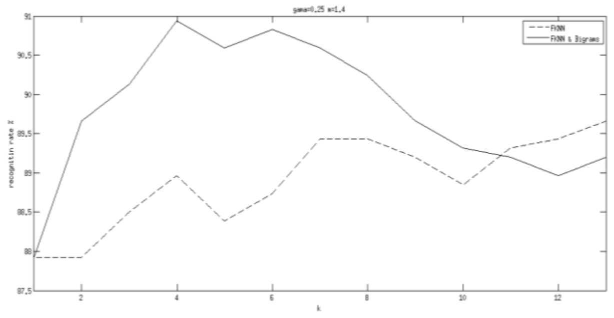


Fig. 5. Recognition rate function of k, Fuzzy KNN & bigrams in solide line, FKNN in dashed line

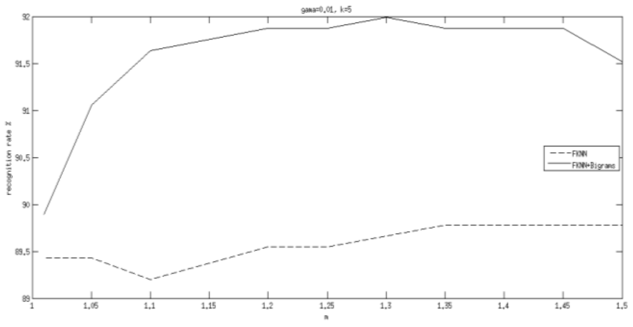
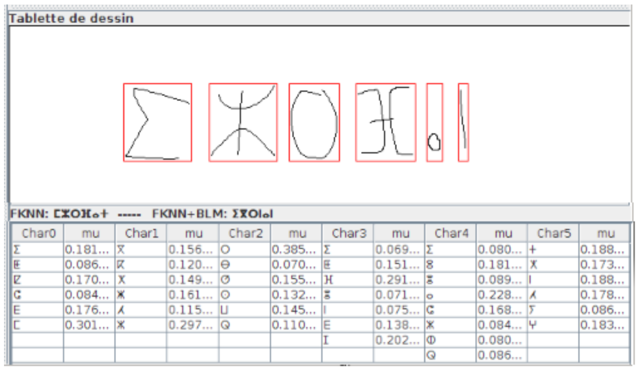


Fig. 6. Recognition rate function of m, Fuzzy KNN & bigrams in solide line, FKNN in dashed line



FKNN: ⊏ⵅOⵀⴰ+ ----- FKNN+BLM: ΣⵅOIⴰI

Char0	mu	Char1	mu	Char2	mu	Char3	mu	Char4	mu	Char5	mu
Σ	0.181...	ⵅ	0.156...	O	0.385...	Σ	0.069...	Σ	0.080...	+	0.188...
ⴹ	0.086...	ⴽ	0.120...	Θ	0.070...	ⴹ	0.151...	8	0.181...	X	0.173...
ⴽ	0.170...	X	0.149...	Ø	0.155...	ⵀ	0.291...	ⵓ	0.089...	I	0.188...
ⵛ	0.084...	ⵣ	0.161...	ⵙ	0.132...	ⵓ	0.071...	ⴰ	0.228...	ʎ	0.178...
E	0.176...	ʎ	0.115...	U	0.145...	I	0.075...	ⵛ	0.168...	ƹ	0.086...
⊏	0.301...	ⵣ	0.297...	Q	0.110...	E	0.138...	ⵣ	0.084...	ⵖ	0.183...
						I	0.202...	Φ	0.080...		
								Q	0.086...		

Fig. 7. Proposed candidate with an inapropriat values of k=11 and m=3.

VI. Conclusion

In this work we presented a new technique for handwritten Tifinagh text recognition. In the features extraction stage we used a linear combination of the gravity center distance and pixel density. Then the fuzzy k nearest neighbor proposes candidates for each character image with its membership degree, the bigram model assigns a probability for each characters sequence. The selected characters sequence is the one that has the higher product. With a good choice of the number of neighbors k, and the fuzziness factor m, bigram model can improved significantly the recognition rate.

References

[1] A. L. Koerich, R. S. (2003). Large Vocabulary Of f-Line Handwriting Recognition: A Survey.Pattern Analysis and Applications. 97–121.
[2] Bunke, U.-V. M. (2001). Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. Int'l Journal of Pattern Recognition and Artificial Intelligence , 15:65–90.
[3] D.Slezak, J. B. Image processing and Pattern recognition. New York: Springer.
[4] Martin H, Jurafsky. D. (2000). Speech and language processing. Prentice Hall.
[5] J.Sagau. (n.d.). Logique flou en classification. Techniques de l'Ingénieur , H3638.
[6] M.Hanmandlu, A. A. Fuzzy Model Based Recognition of Handwritten Hindi Numerals using bacterial foraging. 6th IEEE/ACIS ICIS.
[7] S.Theodoridis, &. K. (2009). Pattern recognation (fort edition). Elsevier.
[8] Zheng, F. (n.d.). K-Means-based fuzzy classifier.