sciendo

JAISCR, 2021, Vol. 11, No. 3, pp. 229 – 242
10.2478/jaiscr-2021-0014

HANDWRITTEN WORD RECOGNITION USING FUZZY MATCHING DEGREES

Michał Wróbel[1,*], Janusz T. Starczewski[1],
Justyna Fijałkowska[2,4], Agnieszka Siwocha[3,4], Christian Napoli[5]

[1]*Department of Computational Intelligence, Czestochowa University of Technology*
al. Armii Krajowej 36, 42-200 Częstochowa, Poland

[2]*Management Department*
University of Social Sciences, 90-113 Łodź

[3]*Information Technology Institute*
University of Social Sciences, 90-113 Łodź

[4]*Clark University Worcester, MA 01610, USA*

[5]*Department of Computer, Control and Management Engineering*
Sapienza University of Rome, Via Ariosto 25 Roma 00185, Italy

[*]*E-mail: michal.wrobel@pcz.pl*

Submitted: 7th September 2020; Accepted: 16th April 2021

Abstract

Handwritten text recognition systems interpret the scanned script images as text composed of letters. In this paper, efficient offline methods using fuzzy degrees, as well as interval fuzzy degrees of type-2, are proposed to recognize letters beforehand decomposed into strokes. For such strokes, the first stage methods are used to create a set of hypotheses as to whether a group of strokes matches letter or digit patterns. Subsequently, the second-stage methods are employed to select the most promising set of hypotheses with the use of fuzzy degrees. In a primary version of the second-stage system, standard fuzzy memberships are used to measure compatibility between strokes and character patterns. As an extension of the system thus created, interval type-2 fuzzy degrees are employed to perform a selection of hypotheses that fit multiple handwriting typefaces.

Keywords: offline handwriting recognition, handwritten strokes, fuzzy matching degrees, interval type-2 fuzzy sets, decision trees, bigram frequency

1 Introduction

Offline handwriting recognition (HWR) systems are applied to read and interpret a handwritten text from a static image, such as a document scan. Due to the inaccuracy of handwriting, this issue is more complex than printed text recognition. Even considering only the basic handwriting style called a cursive script, the letters are connected and thus there are many possible interpretations [8].

There exist many of approaches to HWR based on Convolutional Neural Networks, which are trained on word images or single letters by tuning the various features of the image through filters via the so-called deep learning algorithm.

However, methods that are not based on whole words and are dedicated to reading proper names, e.g. from archival scans, require single letter recognition. An example of such archival handwritten text is illustrated in Figure 1. Standard approaches are based on the decomposition of plotted curves into the so-called strokes. The stroke is identified as a basic fragment of a letter created by a single move of a hand. Obtaining strokes from handwritten text is challenging due to the fact that the text is not arranged in a straight line, handwriting (especially italics and ligatures) makes it difficult to separate and recognize letters. Stroke extraction is typically the first step in HWR systems.

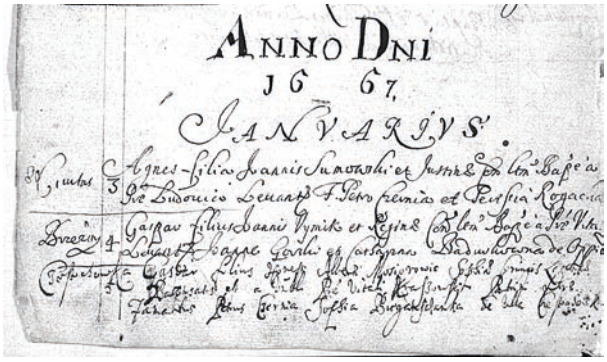

Figure 1. A fragment of the archival handwritten list of births from 1667, the city of Czestochowa.

This paper, basing on the extensive use of our stroke extraction method described in [16], focuses on the problem of identifying the letter or digit patterns within the set of strokes. In this context, we are in need of a method for comparing two small groups of the strokes, the first of which is a group selected from the set of extracted strokes, and the second is a pattern of a letter (or two letters are joined as a single glyph called a ligature).

The similarity of the two groups is not a binary value in general, therefore, fuzzy degrees are proposed to measure and calculate similarity degrees. In our approach, each comparison creates a hypothesis that this group of stokes matches the letter pattern with a certain fuzzy degree, which might be standard fuzzy membership in $[0,1]$ or an interval within the truth range $[0,1]$ describing a membership degree of a fuzzy set of type-2.

The use of type-2 fuzzy sets is advantageous in cases of particular uncertainties as poor scan quality of the source image due to the destruction of the paper, excessive variability and ambiguity of strokes concerning different people or one person over time, and variability of rotation (usually to the right) in cursive text.

To a different degree, a single stroke can be assigned to many hypotheses. Therefore, algorithms for selecting non-exclusive hypotheses, which are likely to be confirmed, are the essential area of research for this work. Actually, this stage transforms sets of strokes into possible words, including many word hypotheses. The selection of the proper one is left to the final decision system.

Summarizing, we present three methods for comparing small (typically limited to three elements) groups of strokes and compatible methods for hypotheses selection, as an extension of our earlier methods provided in [15]. The paper is organized as follows:

- Extraction of strokes from input images, and their approximation are outlined in Section 2.
- Section 3 provides formulas for comparing groups of strokes to letter patterns.
- Section 4 describes a procedure for the preparation of sets of hypotheses.
- Section 5 provides three methods for the selection of hypotheses, one of which is based on interval type-2 fuzzy sets.
- A final selection of a word is the content of Section 6.
- Section 7 presents implementation details and summarizes experimental results.
- Section 8 draws conclusions and ideas for future work.

2 Stroke extraction and approximation

First of all, we need to have a set of extracted strokes. Stroke extraction is not a topic of this paper. We could use, for example, an algorithm proposed in [16] or [9].

We need to represent the stroke as a couple of polynomials

$$\begin{aligned} x(t) &= a_3t^3 + a_2t^2 + a_1t + a_0 \\ y(t) &= b_3t^3 + b_2t^2 + b_1t + b_0, \end{aligned} \tag{1}$$

where

$$t \in [0,1].$$

Usually the extraction stage return strokes given as an ordered set of points:

$$S = (P_1, P_2, ..., P_n). \tag{2}$$

To transform it into polynomial format, let us define t_i as a distance between the beginning of the stroke and the point p_i divided by length of the stroke. We may express it in the following manner:

$$t_i = \frac{\sum_{j=1}^{i-1} \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}}{\sum_{j=1}^{n-1} \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}}. \tag{3}$$

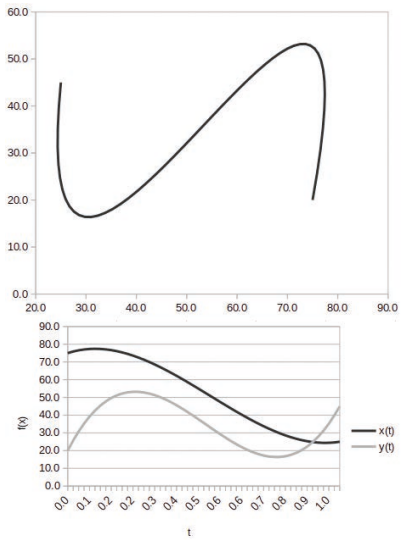

Figure 2. A stroke and its decomposition into two polynomials. The polynomials are:
$x(t) = 175t^3 - 275t^2 + 50t + 75$,
$y(t) = 525t^3 - 800t^2 + 300t + 20$

This approach was mentioned in [5]. Value t_1 is always 0 and value t_n is always 1. Now we may consider a stroke as a couple of functions $x = f_1(t)$ and $y = f_2(t)$. We know the values of these functions in n points. Therefore, we are able to approximate these functions.

We may notice that functions $x = f_1(t)$ and $y = f_2(t)$ have convenient forms to approximate them by polynomials of a degree not greater than 3 – the shape of each stroke is visible. An example of quite a curvy stroke is shown in Figure 2.

Figure 3. An example of the input image

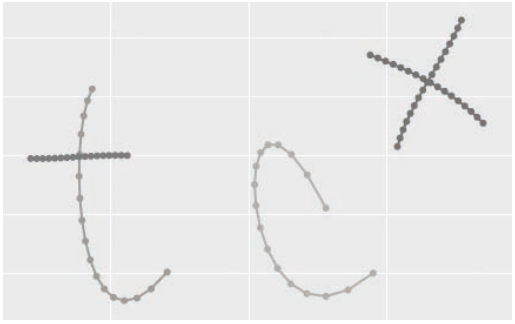

Figure 4. The input image after stroke extraction

In this paper, the set of extracted and approximated strokes set is called *the dataset*. An example of the input image is presented in Figure 3, the strokes extracted from this image and approximated by polynomials are shown in Figure 4.

Using these polynomials we can get the vector of coefficients

$$\vec{v} = (a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0). \tag{4}$$

An algorithm described in this paper requires two input objects: the dataset and the set of letter patterns. In both cases, each stroke should be represented by the vector (4).

As defined in [14], a letter pattern may contain one or more strokes. In this paper, to reduce the complexity, we are analyzing letter patterns containing at most three strokes. If the letter pattern is larger, we can select three strokes that look the most significant (usually the longest ones).

3 Matching the patterns

This Section provides formulae for matching a various number of strokes to the letter pattern.

3.1 Matching one stroke to the letter

The idea of matching a single stroke with the pattern is presented in [13]. Let us briefly describe this.

If we want to measure the similarity between two strokes, we focus just on its shape. Values a_0 and b_0 in the vector (4) represent only localization of the stroke, not a shape, so we can omit these two elements. Hence, the stroke i is described by a vector

$$\vec{s}_i = (a_3, a_2, a_1, b_3, b_2, b_1).$$

Let us notice that the same stroke may be represented by two different vectors since both endpoints can be treated as the beginning of the stroke. Accordingly, the two following functions represent the shape of the same stroke:

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$x'(t) = a_3(1-t)^3 + a_2(1-t)^2 + a_1(1-t) + a_0.$$

If we have vector $\vec{s}_i$, we can calculate its alternative vector $\vec{s}'_i$

$$\vec{s}'_i = (a'_3, a'_2, a'_1, b'_3, b'_2, b'_1),$$

where

$$\begin{aligned} a'_3 &= -a_3, & b'_3 &= -b_3, \\ a'_2 &= 3a_3 + a_2, & b'_2 &= 3b_3 + b_2, \\ a'_1 &= -3a_3 - 2a_2 - a_1, & b'_1 &= -3b_3 - 2b_2 - b_1. \end{aligned} \tag{5}$$

Both vector $\vec{s}_i$ and $\vec{s}'_i$ should be normalized due to the fact that we are not interested in the stroke size at this stage of the algorithm.

$$\hat{\vec{s}}_i = \frac{\vec{s}_i}{\|\vec{s}_i\|}, \hat{\vec{s}}'_i = \frac{\vec{s}'_i}{\|\vec{s}'_i\|}.$$

Let the vector $\vec{s}_i$ represent the stroke in the pattern and the vector $\vec{s}_j$ comes from the dataset. We can express the difference between these vectors just by the euclidean distance between them. Both vectors are units, therefore the distance between them belongs to $[0,2]$. We must remember, that each stroke can be expressed by two vectors. Hence, the shape difference between two strokes can be expressed by

$$D(i,j) = min(\|\hat{\vec{s}}_i - \hat{\vec{s}}_j\|, \|\hat{\vec{s}}_i - \hat{\vec{s}}'_j\|, \|\hat{\vec{s}}'_i - \hat{\vec{s}}_j\|, \|\hat{\vec{s}}'_i - \hat{\vec{s}}'_j\|),$$

and the fuzzy degree for two strokes similarity equals

$$F(i,j) = 1 - \frac{1}{2}D(i,j). \tag{6}$$

3.2 Matching two strokes with the letter

Matching a pattern consisted of two strokes is a bit more complex. We must consider not only the similarity of each stroke pair but also their size and location relative to each other.

Let strokes in the pattern be represented by the vectors $\vec{p}_1$, $\vec{p}_2$, and the corresponding strokes in the dataset be represented by the vectors $\vec{s}_1$, $\vec{s}_2$. Using (6) we get similarity value $F(s_1, p_1)$ and $F(s_2, p_2)$.

Let us consider the stroke sizes. To calculate the stroke length we need to transform the polynomial representation of the stroke into a list of points. Each point may be calculated by

$$P_{s_1,i} = (x(T_i), y(T_i)), \tag{7}$$

where $x(t)$, $y(t)$ are defined by (1) and

$$T = [0, 0.02, 0.04..., 0.98, 1].$$

The resolution of the vector t may be different. The stroke length is equal to the sum of distances between successive points

$$L_s = \sum_{i=2}^{|T|} |P_{s,i-1}P_{s,i}|.$$

We expect that the relation between lengths of the strokes $\vec{s}_1$ and $\vec{p}_1$ is similar to the relation between $\vec{s}_2$ a $\vec{p}_2$. Hence, we can calculate this similarity by the formula

$$L(p_1,p_2,s_1,s_2)=\frac{\min(L_1,L_2)}{\max(L_1,L_2)},$$

where

$$L_1=\frac{L_{p_1}}{L_{s_1}},L_2=\frac{L_{p_2}}{L_{s_2}}.$$

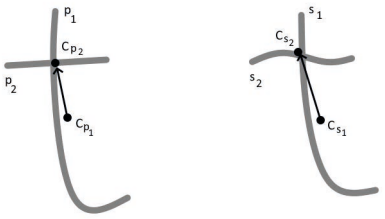

Figure 5. Vectors connecting mass centers of the strokes

Finally, let us define a fuzzy degree for the similarity of the relative location of the strokes. Let C_{s_1} be a center of mass of the stroke $\vec{s_1}$. Coordinates of this point are just average coordinates of all points introduced in (7). It may be expressed by the formula

$$C_{s_1}=\left(\frac{\sum_{i=1}^{|T|}x(T_i)}{|T|},\frac{\sum_{i=1}^{|T|}y(T_i)}{|T|}\right). \tag{8}$$

We assume that the vector $\overrightarrow{C_{s_1}C_{s_2}}$ is similar to the vector $\overrightarrow{C_{p_1}C_{p_2}}$ (taking into account the scale). It is presented in Figure 5. Therefore, the vector

$$\vec{d}=\overrightarrow{C_{p_1}C_{p_2}}-\overrightarrow{C_{s_1}C_{s_2}}\frac{L_1+L_2}{2}.$$

should be very small (its length should be close to zero). Of course, for larger patterns this length could be relatively greater, therefore, a degree for the similarity of the location must depend on the average stroke length. It can be expressed by

$$C(p_1,p_2,s_1,s_2)=1-tanh(\frac{|\vec{d}|}{\frac{1}{2}(L_{p_1}+L_{p_2})}).$$

To sum up, the fuzzy degree for the similarity of two sets containing two strokes is a geometric mean

$$F(s_1,s_2,p_1,p_2)=\sqrt[4]{F(s_1,p_1)F(s_2,p_2)L(p_1,p_2,s_1,s_2)C(p_1,p_2,s_1,s_2)}. \tag{9}$$

3.3 Matching three strokes with the pattern

In order to match a letter pattern consisting of three strokes (p_1,p_2,p_3) with three strokes in the recognized set (s_1,s_2,s_3), a similar formula to (9) can be applied. The proposed formula is symmetrical as long as the strokes are sorted according to their length, i.e.

$$(L_{s_1}+L_{p_1})\geq(L_{s_2}+L_{p_2})\geq(L_{s_3}+L_{p_3}). \tag{10}$$

Next, the fuzzy degree for the similarity of three strokes can be extended to the form of the geometric mean

$$\begin{aligned}&F(s_1,s_2,s_3,p_1,p_2,p_3)=\\&=\sqrt[7]{F(s_1,p_1)F(s_2,p_2)F(s_3,p_3)}*\\&*\sqrt[7]{L(p_1,p_2,s_1,s_2)L(p_1,p_3,s_1,s_3)}*\\&*\sqrt[7]{C(p_1,p_2,s_1,s_2)C(p_1,p_3,s_1,s_3)}.\end{aligned} \tag{11}$$

3.4 A general formula

Observing equations (9) and (11), a general formula for similarity of sets consisting of n strokes can be proposed in the following form

$$F(s_1,...,s_n,p_1,...,p_n)=\sqrt[3n-2]{\prod_{i=1}^{n}F(s_i,p_i)\prod_{i=2}^{n}[L(p_1,p_i,s_1,s_i)C(p_1,p_i,s_1,s_i)]}. \tag{12}$$

However, formulas for the number of strokes greater than three have been not implemented yet. The vast majority of letters (or even simple ligatures i.e. clusters of letters like "le") are recognizable after only three dashes.

4 Preparing the set of hypotheses

If the letter pattern consists of only one stroke, a single hypothesis for each stroke in the dataset can be formulated. The hypothesis is true if the stroke fully matches the letter pattern stroke, which is possible to a degree (6). Since matching one stroke occurs more possibly than matching a complex pattern, the

fuzzy degree for the single stroke comparison can be concentrated with the use of the square operator.

Whenever the letter pattern is composed of two strokes, a single hypothesis for each combination of two elements from the dataset can be derived. Hence, the number of such equals $N(N-1)$, where N is the size of the dataset. The possibility that the hypothesis is true is formulated by (9). Two hypotheses are derived for each combination, since we can assign $\vec{p}_1$ and $\vec{p}_2$ in two directions, therefore, the hypothesis with the greater possibility should be stored.

Analogically, in cases of letter patterns containing three strokes, a group of strokes can be matched to the letter pattern in six different manners – (s_1,s_2,s_3), (s_1,s_3,s_2), (s_2,s_1,s_3), (s_2,s_3,s_1), (s_3,s_1,s_2), and (s_3,s_2,s_1). Again, one of these hypotheses with the highest possibility have to be stored.

To conclude, a fuzzy membership degree of pattern matching determines the fuzzy matching degree of a hypothesis and is defined by the following formula

$$f_i = \begin{cases} F^2(s,p) & : \{s\} \\ F(s_1,s_2,p_1,p_2) & : \{s_1,s_2\} \\ F(s_1,s_2,s_3,p_1,p_2,p_3) & : \{s_1,s_2,s_3\}. \end{cases} \tag{13}$$

One of such received hypotheses is represented by the set of strokes associated with the letter pattern with the degree indicated by the fuzzy membership value.

Before the recognition stage, very short lines, possibly noises, should be removed from the set. To do this, we need to count the median lengths of the strokes in the input set and remove small strokes, e.g. smaller than 1/4 of the median. Then the lines have to be sorted by the x coordinate of their centers of gravity and the hypothesis should be constructed from one, two ore three consecutive lines depending on the number of lines in the matched pattern.

This part of the algorithm has time complexity $O(NP)$, where N is the number of strokes in the dataset and P is the number of patterns. In order to reduce the total number of received hypotheses, we can limit the number of hypotheses created for a single pattern, e.g. on $\frac{1}{2}N$. For each pattern class, the list of received hypotheses has to be sorted by the fuzzy degree F in descending order and may have deleted the poorest hypotheses, if necessary.

5 Selection of hypotheses

Let $A = \{h_1, h_2, \ldots, h_{|A|}\}$ be the set of all hypotheses derived by the algorithm presented in the previous Section. Each hypothesis h_i has the following values assigned:

- a set S_i containing such strokes from the dataset that are covered by the letter pattern,
- a fuzzy degree f_i defined by (13),
- a target letter (or, generally, a small fragment of text).

Our goal is to create a subset $R \subset A$ that each stroke belongs to at least one hypothesis. Such subsets represent suggested words. The degree of matching such propositions can be introduced as an average fuzzy degree of all hypotheses for R.

$$Q(R) = \frac{1}{|R|} \sum_{h_i \in R} f_i. \tag{14}$$

We can assume that $Q(\emptyset) = 1$.

5.1 A basic greedy method

One of the simplest and the most intuitive approaches to obtain a subset R is a greedy algorithm, which may be listed in the following steps:

1. Set $R = \emptyset$.
2. Sort A by f_i in descending order.
3. Pop the hypothesis h_0 with the highest f_i.
4. Add h_0 into R.
5. Remove (from A) the hypotheses having any common strokes with h_0.
6. If $A \neq \emptyset$, go back to step 3.

Greedy algorithms are quite successful in problems as data compression (e.g. Huffman encoding) or finding the shortest paths between nodes in graphs (as Dijkstra's algorithm). The selection method based on the greedy algorithm makes the optimal choice only at each iteration; however, it is not guaranteed that a greedy strategy leads to an optimal solution since decisions are based only on the local graph information without regard to the overall graph.

5.2 A selection method using fuzzy degrees

Since the greedy method does not guarantee to receive the result with the highest degree of matching $Q(R)$, we propose a fuzzy tree method that generates more variants of R with the use of fuzzy degrees of matching. As a result, we obtain a family of sets $\mathcal{R}$. In each node of the tree, a decision which hypothesis should be added to R is made with the following options: the first option is the one with the highest f_i, similarly to the greedy method, and the second option is formed by the hypotheses that have at least one common stroke with h_i acquired in the first option.

Therefore, the recursive procedure for the construction of the decision tree can be summarized as follows:

Initially, $R = \emptyset$ and A includes all the hypotheses.

1. Pop the hypothesis h_0 with the highest f_i.
2. Create a subset $T \subset A$ where each element $h_t \in T$ has at least one common stroke with h_0.
3. For each $h_t \in T$:
   (a) Create a set $R_t = R \cup \{h_t\}$.
   (b) Create a set $A_t \subset A$ by removing from A all the hypotheses having any common strokes with h_t.
   (c) If $A_t = \emptyset$, add the set R_t into $\mathcal{R}$. Otherwise, invoke this procedure recursively with $R = R_t, A = A_t$

This algorithm generates excessively many variants $R_t \in \mathcal{R}$, hence, it is practically hard to use. Nevertheless, its modification presented below can reduce the time complexity significantly.

Let L be the list of pairs (A_t, R_t) sorted by $Q(R_t)$ in descending order, which initially contains only one pair $(A, \emptyset)$, and let the maximum number of pairs in L be denoted as l_{max} and the maximum number of variants in $\mathcal{R}$ be denoted as r_{max}. The modified algorithm for expanding the hypothesis selection tree can be organized as follows:

1. Pop the pair (A_0, R_0) from L with the highest $Q(R_i)$, remove it from the list.
2. Pop the hypothesis h_0 from A_0 with the highest f_i.
3. Create a subset $T \subset A_0$ where each element $h_t \in T$ has at least one common stroke with h_0.
4. For each $h_t \in T$:
   (a) Create a set $R_t = R \cup \{h_t\}$.
   (b) Create a set $A_t \subset A$ by removing from A all the hypotheses having any common strokes with h_t.
   (c) If $A_t = \emptyset$, add the set R_t into $\mathcal{R}$. Otherwise, add a pair (R_t, A_t) into sorted list L.
5. If the number of stored pairs $|L|$ is greater than l_{max}, remove the poorest ones.
6. If the number of sets R_t is lower than r_{max} and L is not empty, return to step 1.

5.3 An interval type-2 fuzzy selection

Until now, the patterns are composed of letters or ligatures from a single handwriting style. In order to extend this base to multiple styles, their aggregate can be implemented in the form of interval type-2 fuzzy sets. Formally, a fuzzy set of type-2, denoted by $\tilde{A}$, is a vague collection of elements characterized by membership function $\mu_{\tilde{A}} \colon \mathbb{X} \to \mathcal{F}([0,1])$, where $\mathcal{F}([0,1])$ is a set of all classical fuzzy sets in the unit interval $[0,1]$ and $\mathbb{X} \subset \mathbb{R}$. Practically, each $x \in \mathbb{X}$ is associated with a secondary membership function $f_x \in \mathcal{F}([0,1])$ introduced by a mapping $f_x \colon [0,1] \to [0,1]$. If f_x is a closed subinterval of the real unit interval $[0,1]$, denoted by $f_x \in I([0,1])$, then $\tilde{A}$ defines the interval type-2 fuzzy set. Practically, interval type-2 fuzzy sets are characterized with their bounds as $\tilde{A}$ and $\tilde{B}$ can be characterized by $\mu_{\tilde{A}}(x) = [\underline{u}_x, \overline{u}_x] \in I([0,1])$and $\mu_{\tilde{B}}(x) = [\underline{v}_x, \overline{v}_x] \in I([0,1])$. Therefore, the set operations, as intersection $\tilde{A} \cap \tilde{B}$, union $\tilde{A} \cup \tilde{B}$ and complement $\neg\tilde{A}$, can be realized for each $x \in \mathbb{R}$ by the so-called extended operations, i.e.,

$$\widetilde{\min}(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)) = [\min(\underline{u}_x, \underline{v}_x), \min(\overline{u}_x, \overline{v}_x)] \ , \tag{15}$$

$$\widetilde{\max}(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)) = [\max(\underline{u}_x, \underline{v}_x), \max(\overline{u}_x, \overline{v}_x)] \ , \tag{16}$$

$$\widetilde{\mathrm{neg}}(\mu_{\tilde{A}}(x)) = [1 - \overline{u}_x, 1 - \underline{u}_x] \ . \tag{17}$$

The set of truth intervals $I([0,1])$ is only a partially ordered set under $\widetilde{\max}$ and $\widetilde{\min}$, [2], while the ordering relation for truth intervals is defined as

$$[\underline{u}, \overline{u}] \subseteq [\underline{v}, \overline{v}] \iff \underline{u} \leq \underline{v} \wedge \overline{u} \leq \overline{v} \ . \tag{18}$$

The use of interval type-2 sets has been confronted to many sources of uncertainty as rounding computational errors, quantization errors, measurement non-statistical errors, component tolerances, and knowledge uncertainties about exact values of physical parameters [6, 7, 12]. They as well can be used to model different handwriting typefaces.

Let the stroke of the letter pattern stroke be described by a vector of intervals

$$\vec{p} = ([\underline{a}_3, \overline{a}_3], [\underline{a}_2, \overline{a}_2], [\underline{a}_1, \overline{a}_1], [\underline{b}_3, \overline{b}_3], [\underline{b}_2, \overline{b}_2], [\underline{b}_1, \overline{b}_1]) .$$

Each parameter can be aggregated from multiple handwriting typefaces according to the following formulas:

$$\underline{a}_i = \min_{j=1,\ldots,J} a_j \tag{19}$$

$$\overline{a}_i = \max_{j=1,\ldots,J} a_j \tag{20}$$

$$\underline{b}_i = \min_{j=1,\ldots,J} b_j \tag{21}$$

$$\overline{b}_i = \max_{j=1,\ldots,J} b_j , \tag{22}$$

where $i \in 1,2,3$ and J is the number of letter patterns to be aggregated.

The fuzzy degree for the similarity between interval pattern (of a letter or a ligature) and non-interval stroke can be proposed as interval type-2 fuzzy memberships:

$$\underline{F}_\alpha(i,j) = \max\left(\min\left(2\frac{\alpha_j - \underline{\alpha}_i}{\overline{\alpha}_i - \underline{\alpha}_i}, 2\frac{\overline{\alpha}_i - \alpha_j}{\overline{\alpha}_i - \underline{\alpha}_i}\right), 0\right) \tag{23}$$

$$\overline{F}_\alpha(i,j) = \max\left(\min\left(\begin{array}{c}\frac{\alpha_j - 2\underline{\alpha}_i + \overline{\alpha}_i}{\overline{\alpha}_i - \underline{\alpha}_i}, \\ \frac{2\overline{\alpha}_i - \underline{\alpha}_i - \underline{\alpha}_j}{\overline{\alpha}_i - \underline{\alpha}_i}, 1\end{array}\right), 0\right) , \tag{24}$$

where a universal parameter α stands for a_n or b_n with $n = 1,2,3$. Figure 6 illustrates the construction of the type-2 fuzzy similarity set. In terms of the possibility theory, the upper membership degrees form a trapezoidal function, which might be interpreted as the upper limit of possible stroke-to-pattern matches, while the lower degrees constitute a triangular function of certain matches.

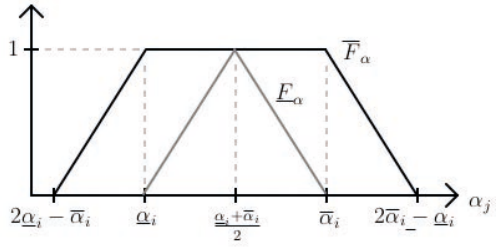

Figure 6. Fuzzy grade for the similarity between the uncertain letter pattern and the precise stroke

The obtained interval type-2 fuzzy grade is multidimensional, henceforth, the standard Cartesian product is able to reduce dimensionality, i.e.,

$$\underline{F}(i,j) = \min_{\alpha \in \{a_1,a_2,a_3,b_1,b_2,b_3\}} (\underline{F}_\alpha(i,j)) \tag{25}$$

$$\overline{F}(i,j) = \min_{\alpha \in \{a_1,a_2,a_3,b_1,b_2,b_3\}} \left(\overline{F}_\alpha(i,j)\right) . \tag{26}$$

5.4 Decision trees examples

Following an example provided in [14], let the exemplary dataset be composed of three strokes as illustrated in Figure 7. For a given set of strokes, a set of hypotheses have been generated, which is listed in Table 2. Each hypothesis included one or two strokes matching the letter pattern. Fuzzy grade of similarity between strokes and the pattern is assessed qualitatively using (14).

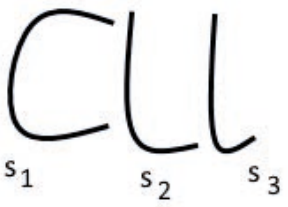

Figure 7. An example of input dataset

Table 1. The set of hypotheses

Hypothesis	Strokes	Letter	Quality Q
h_1	s_3	i	0.9
h_2	s_1	c	0.8
h_3	s_1, s_2	a	0.7
h_4	s_2, s_3	u	0.6
h_5	s_2	i	0.5

A full decision tree, presented in Figure 8, has been constructed along with the methods using the greedy search as well as fuzzy grades.

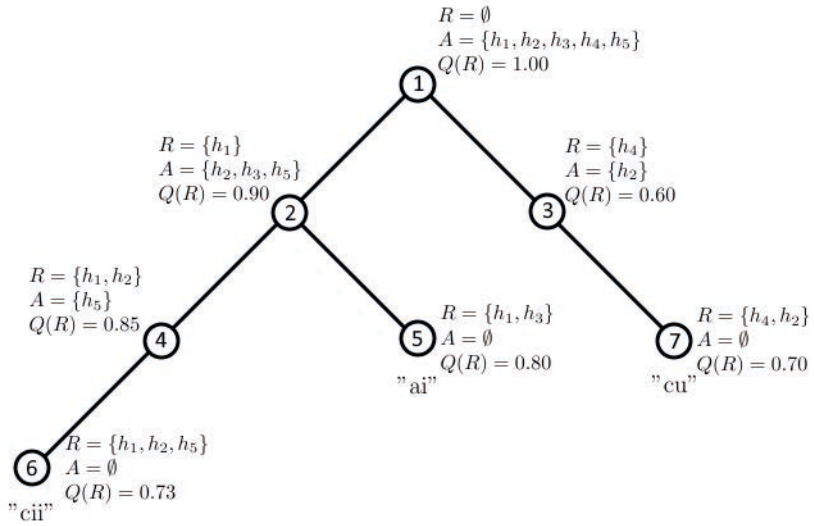

Figure 8. Decision tree using fuzzy degrees of matching

The numbers inside the nodes mean the order in which the nodes were created. Three different text variants of the text have been generated: "ai" ($Q = 0.80$), "cii" ($Q = 0.73$), and "cu" ($Q = 0.80$). The greedy method has selected a subset $R = \{h_1, h_2, h_5\}$, which average fuzzy membership is $Q = 0.73$. The optimized fuzzy version of the algorithm has skipped some nodes to avoid poorly promising paths. It can be easily seen that the algorithm using fuzzy grades is able to generate a variant with a greater membership grade for strokes-to-class matching than the greedy method does.

An analogical decision tree is presented in Figure 9. The upper bounds of the degree of matching have been determined without changes using the arithmetic mean of partial membership grades. They play a role of a limit for possible matches between strokes and classes that resembles terms in the possibility theory. However, the lower membership grades act as certain matching, hence the lower degree of matching of the path should be aggregated with an operator that preserves the nature of the lower limit such as the minimum operator.

6 Word selection

As the final decision, a single text variant may be chosen making use of the two following criteria for evaluation of hypotheses:

- the grade of pattern matching of the particular text variant,
- the chance that the given string of letters would make sense in the given language.

To apply the second criterion, we can use statistical data provided in the paper [4], which contains information on how often a given bigram (understood as a pair of consecutive letters) appears in the English language corpus. The most common bigram has a degree of matching equal to 1.0, and in other cases, the fuzzy degree is equal to the number of occurrences of a given bigram divided by the number of occurrences of the most frequent one. Since the data in the source table are stored in logarithmic form, the final fuzzy degree of matching shall be described by the following formula:

$$Q'_{ij} = \frac{e^{x_{ij}}}{e^{x_{max}}} = e^{x_{ij} - x_{max}}. \tag{27}$$

Since the median and mean of these values are very small (0.003 and 0.049 respectively), they

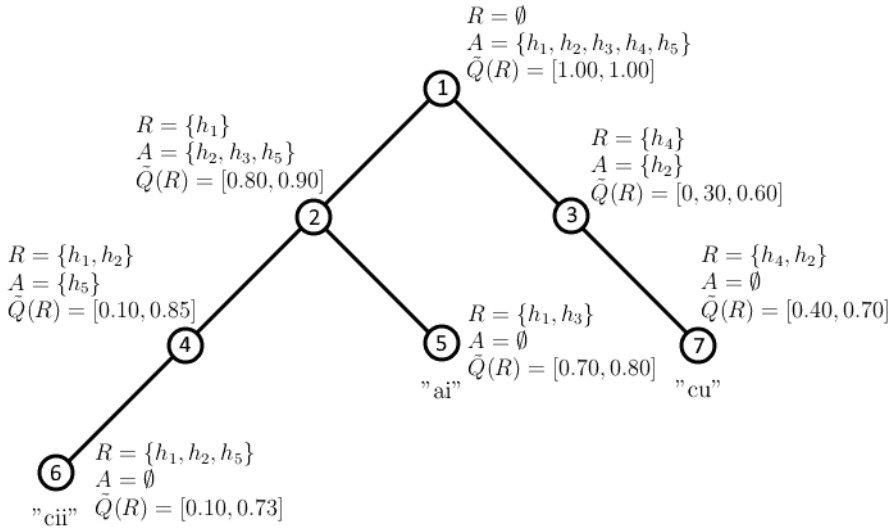

Figure 9. Decision tree using interval type-2 fuzzy memberships

should be additionally adjusted using the fourth-degree root. After this operation, the median is 0.242 and the average is 0.247, so the best big frames do not overwhelm the rest. In summary, the formula (27) finally takes the form:

$$Q_{ij} = \sqrt[4]{\frac{e^{x_{ij}}}{e^{x_{max}}}} = e^{\frac{1}{4}(x_{ij} - x_{max})}. \tag{28}$$

Operating on such values, we can define *meaningfulness* of a text variant as the arithmetic mean of fuzzy matching degrees of successive bigrams, i.e.,

$$B(S) = \frac{1}{|S| - 1} \sum_{i=1}^{|S|-1} Q_{s_i s_{i+1}}. \tag{29}$$

The final fuzzy matching degree of the string may be determined as the geometric mean of the two degrees of matchings: one achieved during recognition and the *meaningfulness* grade taking into account dictionary dependencies between letters. It needs to be noted that using a different corpus, the method can be tuned to a different language.

7 Implementation and experimental results

The algorithm has been implemented in Python 3. A stroke extraction stage has been performed by the method presented in [16]. The set of input strokes and the set of letter patterns have been stored in CSV files. Since a stroke can be represented by a specific class, some computations, like (5) and (7), have been calculated once during loading the input files. The experiment has consisted of recognizing the written word *tex*, whose image is shown in Figure 3). The set of patterns, which has been used in the simulation, is collected in Figure 10.

Table 2 summarizes fuzzy degrees of matching obtained for the whole set of hypotheses in recognition of the handwritten word *tex*. The second column of the table represents the fuzzy matching degree from the recognition stage calculated by (14). In the third column, the fuzzy matching is calculated by (29) based on the bigrams is listed. The first column is the overall matching grade determined as the geometric mean of recognition and bigram fuzzy degrees.

It can be observed that the bigram analysis strongly promoted *pronounceable* solutions (tiv, tre, tev) at the expense of random character strings, which had quite high indexes after the recognition stage (tlv, ilix, clix, cllx).

Figure 10. The set of letter patterns used in simulation

Table 2. The set of hypotheses for the handwritten text *tex*.

Overall matching	Recognition *Q*	*Meaning–fulness B*	Text
0.735	0.842	0.641	tex
0.686	0.632	0.744	tre
0.676	0.855	0.534	tix
0.642	0.629	0.656	tro
0.638	0.678	0.600	olex
0.620	0.692	0.556	clex
0.618	0.663	0.577	siex
0.618	0.605	0.631	trs
0.617	0.660	0.577	siex
0.602	0.674	0.537	ciex
0.601	0.680	0.531	slex
0.600	0.677	0.531	slex
0.600	0.688	0.523	olix
0.580	0.701	0.479	clix
0.575	0.661	0.501	oiex
0.573	0.608	0.539	trc
0.560	0.700	0.449	ollx
0.560	0.690	0.455	slix
0.560	0.684	0.458	silx
0.559	0.687	0.455	slix
0.559	0.682	0.458	silx
0.540	0.696	0.419	cilx
0.538	0.690	0.420	jlex
0.461	0.643	0.331	cux
0.537	0.713	0.405	cllx
0.534	0.625	0.457	oux
0.530	0.673	0.417	jiex
0.517	0.702	0.380	sllx
0.516	0.699	0.380	sllx
0.511	0.682	0.383	oilx
0.500	0.673	0.371	siix
0.499	0.670	0.371	siix
0.490	0.700	0.343	jlix
0.477	0.628	0.363	sux
0.476	0.684	0.331	ciix
0.476	0.624	0.363	sux
0.468	0.607	0.361	trj
0.459	0.871	0.242	tlx
0.456	0.694	0.299	jilx
0.445	0.671	0.295	oiix
0.437	0.712	0.269	jllx
0.433	0.651	0.288	olcx
0.419	0.641	0.274	jux
0.403	0.664	0.244	clcx
0.385	0.805	0.184	tcx
0.380	0.683	0.212	jiix
0.378	0.653	0.219	slcx
0.378	0.650	0.219	slcx
0.267	0.663	0.108	jlcx
0.203	0.751	0.055	tjx

8 Conclusions and future works

In this paper, we have presented stroke-based methods for the recognition of handwritten text operating in the offline mode. The methods have been intended to involve creating and evaluating hypotheses as to whether a group of strokes matches a pattern of a certain letter or ligature. Fuzzy grades have been employed for the evaluation of hypotheses, i.e. simply stroke-letter matches, in the first stage methods, while in the second-stage methods, the hypotheses have been verified using a decision tree equipped with either fuzzy grades, or interval fuzzy sets of type-2.

Fuzzy grades have allowed for an efficient traversal of the tree expanding only reasonable sequences of consecutive hypotheses. The experiment has confirmed the effectiveness of using fuzzy membership degrees in ordering and assessing the most possible hypotheses in the case of a simple base of letter patterns.

Nevertheless, HWR systems should support multiple ways of writing letters by different writers and be resistant to changes in handwriting due to the passage of time with different tilt angles, aspect ratios, varying proportions, ambiguities, missing or corrupted lines, etc. For handling such uncertainties, an interval type-2 fuzzy decision model has been proposed. Interval type-2 fuzzy sets have been demonstrated to be capable of handling and processing uncertainties of such a type when a data model has to be aggregated from different sources and all sources are equally reliable.

Future research will focus on the development of the final decision step, taking into account the uncertainties modeled by type-2 fuzzy sets, as has been presented in this paper. The database of letter patterns and ligatures should be automatically extended with many different typefaces so that the interval fuzzy sets could describe the widest possible range of possibilities in the written text. As an extension, the HWR system should be able to learn and interpret new handwriting styles during recognition in online interaction with the system operator (such as [17]). This stage should combine fast dictionary techniques for searching text with a fixed structure with the mechanisms for reading names, surnames, places, proper names, rare and unique words, which we deal with in the analysis of archival records.

Acknowledgments

The project has been financed under the program of the Minister of Science and Higher Education "Regional Initiative of Excellence" in the years 2O19–2022 — project number 020/RID/2018/19, the amount of financing PLN 12,000,000.

References

[1] D. Dubois and H. Prade. Fuzzy sets and systems: Theory and applications. Academic Press, Inc., New York, 1980.

[2] M. Jones and D. Mewhort. Case-sensitive letter and bigram frequency counts from large-scale english corpora. Behavior Research Methods, Instruments, & Computers, 36:388–396, 2004.

[3] E. L'Homer. Extraction of strokes in handwritten characters. Pattern Recognition, 33:1147–1160, 2000.

[4] J. M. Mendel. Computing with words and its relationships with fuzzistics. Information Sciences, 177(4):988–1006, 2007.

[5] R. K. Nowicki and J. T. Starczewski. A new method for classification of imprecise data using fuzzy rough fuzzification. Inf. Sci., 414:33–52, 2017.

[6] D. J. Ostrowski and P. Y. K. Cheung. A Fuzzy Logic Approach to Handwriting Recognition, pages 299–314. Vieweg+Teubner Verlag, Wiesbaden, 1996.

[7] D. Phan, I.-S. Na, S.-H. Kim, G.-S. Lee, and H.-J. Yang. Triangulation based skeletonization and trajectory recovery for handwritten character patterns. KSII Transactions on Internet and Information Systems, 9:358–377, 2015.

[8] J. T. Starczewski. Advanced Concepts in Fuzzy Logic and Systems with Membership Uncertainty, volume 284 of Studies in Fuzziness and Soft Computing. Springer, 2013.

[9] M. Wróbel, K. Nieszporek, J. T. Starczewski, and A. Cader. A fuzzy measure for recognition of handwritten letter strokes. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, editors, Artificial Intelligence and Soft Computing, pages 761–770, Cham, 2018. Springer International Publishing.

[10] M. Wróbel, J. T. Starczewski, and C. Napoli. Handwriting recognition with extraction of letter fragments. In L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M.

Zurada, editors, Artificial Intelligence and Soft Computing, pages 183–192, Cham, 2017. Springer International Publishing.

[11] M. Wróbel, J. T. Starczewski, and C. Napoli. Grouping handwritten letter strokes using a fuzzy decision tree. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, editors, Artificial Intelligence and Soft Computing, pages 103–113, Cham, 2020. Springer International Publishing.

[12] M. Wróbel, J. T. Starczewski, K. Nieszporek, P. Opiełka, and A. Kaźmierczak. A greedy algorithm for extraction of handwritten strokes. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, editors, Artificial Intelligence and Soft Computing, pages 464–473, Cham, 2019. Springer International Publishing.

[13] M. Zalasiński, K. Łapa, K. Cpałka, K. Przybyszewski, and G. G. Yen. On-line signature partitioning using a population based algorithm. Journal of Artificial Intelligence and Soft Computing Research, 10(1):5–13, 2019.

Michał Wróbel is a Ph.D. student with the Department of Intelligent Computer Systems, Cze¸stochowa University of Technology, since 2016. He received the B.Sc. degree in Computer Science from the Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Gliwice, in 2015, where he also got the M.Sc. degree in 2016. His master's thesis concerned on greedy algorithms. Several times he attended Iternational Conference on Artificial Intelligence and Soft Computing. Since 2019 he cooperates with ConnectPoint Sp. z.o.o. where he participates in the project in the field of data science. His current research focuses on off-line HWR. His interests include artificial intelligence, heuristics, and image processing.

Janusz T. Starczewski is an Associate Professor with the Department of Intelligent Computer Systems at Cze¸stochowa University of Technology, where he is Head of education for discipline of information and communication technology. He holds the Ph.D and D.Sc. degrees in Computer Science and an M.Sc. in Electrical Engineering. He is an experienced project contributor in artificial intelligence and IT systems. The mainstream of his scientific achievements comprise studies on advanced concepts of fuzzy logic, including type-2 fuzzy logic systems and their combinations with the rough set theory. He has authored more than 50 publications. His book "Advanced Concepts in Fuzzy Logic and Systems with Membership Uncertainty" has been granted by the Polish Minister of Science and Higher Education.

Justyna Fijałkowska received the M.Sc. degree from the University of Lodz, Poland, in 2001 and her Ph.D. degree in 2009 from the University of Social Sciences, Lodz, Poland. She is currently a professor at the University of Social Sciences in Lodz, Poland, and at Clark University Worcester USA, a branch campus in Lodz and Warsaw. She is also an Assistant Professor at the Faculty of Economy and Sociology, University of Lodz. Her current research interests include Sustainability Reporting and Integrated Reporting, textual analysis, machine learning content analysis, topic modeling, and various applications of artificial intelligence.

Agnieszka Siwocha received M.Sc. the degree from Lodz University of Technology, Faculty of Technical Physics, Computer Science and Applied Mathematics, and her a Ph.D. in 2015 in the field of computer science, computer graphics at the University of Social Sciences, Łódź, Poland. She is currently an assistant professor at the Social Academy of Sciences in Łódź. She has a title of Adobe Certified Expert, and provides training on Adobe software and computer graphics. Author of over 20 publications related to various problems of computer science, computer graphics and IT applications. Her present research interests include fractal coding, compression, and the quality of digital images, computer graphics, machine learning, and multifractal analysis.

Christian Napoli is Associate Professor with the Department of Computer, Control, and Management Engineering ”Antonio Ruberti”, Sapienza University of Rome, since 2019, where he also collaborates with the department of Physics and the Faculty of Medicine and Psychology, as well as holding the office of Scientific Director of the International School of Advanced and Applied Computing (ISAAC). He received the B.Sc. degree in Physics from the Department of Physics and Astronomy, University of Catania, in 2010, where he also got the M.Sc. degree in Astrophysics in 2012 and the Ph.D. in Computer Science in 2016 at the Department of Mathematics and Computer Science, he obtained the National Scientific Abilitation as associate professor in Computer Engineering (2017) and computer science (2019). He is involved in several international research projects, serves as reviewer and member of the board program committee for major international journals and international conferences. His current research interests include neural networks, artificial intelligence, human-computer interation and computational neuropsychology.