Preprocessing Algorithm for Deciphering Historical Inscriptions Using String Metric

Loránd Lehel Tóth[1], Raymond Eliza Ivan Pardede[1], György András Jeney[1], Ferenc Kovács[2], Gábor Hosszú[1,*]

[1]Department of Electron Devices, Budapest University of Technology and Economics, Budapest, Hungary.

[2]Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, Hungary.

Abstract

The article presents the improvements in the preprocessing part of the deciphering method (shortly preprocessing algorithm) for historical inscriptions of unknown origin. Glyphs used in historical inscriptions changed through time; therefore, various versions of the same script may contain different glyphs for each grapheme. The purpose of the preprocessing algorithm is reducing the running time of the deciphering process by filtering out the less probable interpretations of the examined inscription. However, the first version of the preprocessing algorithm leads incorrect outcome or no result in the output in certain cases. Therefore, its improved version was developed to find the most similar words in the dictionary by relaying the search conditions more accurately, but still computationally effectively. Moreover, a sophisticated similarity metric used to determine the possible meaning of the unknown inscription is introduced. The results of the evaluations are also detailed.

1. Introduction

The writing systems are composed of *graphemes* (in other words *characters*), which are the minimally distinctive units of a writing system [1]. There are different types of the grapheme, e.g., letter, ligature, numerical digit, and punctuation mark. The *script* means a writing system that includes different versions called *alphabets.* E.g., the Latin script has several alphabets, including the French, German, English, Hungarian, etc. alphabets, which are specific sets of graphemes of a script that are used in each orthography. The term "alphabet" is used in a wide sense, not only for the true alphabetic scripts.

Glyph is the shape of the grapheme with topological information. Since the glyphs changed through time, the various versions of the script contain different glyphs for each grapheme. The glyphs of graphemes of a script are realized in the inscriptions as *symbols.* Therefore, the grapheme is the abstraction of a symbol. The symbols are the minimal individual visual units of the inscription. *Typical glyph* is a glyph of a grapheme, which is specific for a certain inscription, but shaped to eliminate the inherent uncertainties of the hand-writing symbols. In such a way, the typical glyph of a grapheme is in accordance with the most significant visual properties of the symbols in a certain inscription or more inscriptions. The *normalized glyph* is one of the typical glyphs of a grapheme that is generally used for representing the grapheme in publications.

---

* Corresponding author. E-mail address: hosszu@eet.bme.hu

Tel.: +36-1-4632724; Fax: +36-1-4632973

The aging of the script identification differs from the Optical Character Recognition (OCR), since in case of the OCR, the normalized glyph of each grapheme (character) belonging to a certain script is known. Therefore, the task of the OCR is to convert the symbols in a certain inscription into the well-known normalized glyphs of a script. In other words, the OCR focuses on the automatic grapheme extraction from a certain inscription [2]. Oppositely, in the computational paleography and more specifically, in the aging identification the right interpretation of the symbols in an inscription is the main problem to be solved. Computational paleography applies computational algorithms in the study of ancient writings and inscriptions such as optimization or mathematical statistical methods [3].

In our research, the majority of the glyph variants of each grapheme can be known a priori from the extant relics, mainly from historical abecedaria used as references; however, several relics may contain unknown glyph variants. We assume that the writing system of the examined inscription is known but the age of that inscription and the actual glyph variant of each grapheme used in that inscription is still unidentified or partly unidentified. This identification problem is difficult, since (*i*) the earlier versions of a script are not known completely, (*ii*) typically several sound values can belong to one grapheme, (*iii*) the orthography of the historical inscriptions sometimes ambiguous, e.g., in several scripts, the vowels were not consequently written, or ligatures of graphemes were frequently applied in certain scripts [4].

The altering approaches of the paleography and the computer science together could improve the paleographic and linguistic researches, deciphering unknown inscriptions, ancient and medieval documents using computer algorithms, machine vision, statistical data analysis and machine learning [3]. The major challenge for computational approaches is to develop and provide a system that will represent paleographical data quickly and easily [5].

Gottfried et al. introduced a document analysis methodology using the interactive computer-aided transcription of handwritings [6]. They especially focused on the glyph separation problem. The combined the automatic and user-machine interactive methods.

Several researchers focused on the interaction of paleography, the study of ancient and medieval documents, with computerized tools, particularly those developed for analysis of digital images and text mining. Panagopoulos et al. combined two pattern-recognition approaches to identify the writers of ancient documents [7]. The fundamental concept of these methods is to identify a major representative of each inscription. The first algorithm estimates a prototype for the inscription, and its goal is to suppress noise from different realizations of the same grapheme produced by the same writer. The second approach initially estimates the curvature at each pixel of grapheme contour. It achieves the fitting of the grapheme realizations (symbols) by means of curvature since the distortion of each glyph contour corresponds to tractable deformation of it. The combination of the two approaches was applied in order to classify the documents to the corresponding writers.

Kavallieratou et al. developed an integrated document analysis system, which has one module of the handwritten word recognition algorithm based on structural characteristics and lexical support [8]. In general, their grapheme recognition procedure consists of two steps. The first step is the feature extraction where each grapheme is represented as a feature vector and the second step is classification of the vectors into a number of classes. They present the newly introduced radial histogram, out-in radial and in-out radial profiles near the well-known horizontal and vertical histograms, in combination with for representing 32x32 matrices of graphemes, as 280-dimensional vectors, which are structural approach for feature extraction. The k-means clustering algorithm is used for classification. On word recognition level, the system has been supported by a lexical component.

Singh described a new method called String Distance Measurement (SDM) for recognizing handwritten graphemes and estimates the error rate performing a cross-validation study with neural networks using SDM pattern recognition [9]. The advantage of the technique is that it can be applied in a generic manner to different applications which involve shape

recognition and may be successfully modified for individual applications. The algorithm is based on the measurement of gradient change and using neural networks to evaluate them. The technique is expected to perform better in uncertain and noisy environments compared to the existing methods.

Märgner et al. presented an approach to recognize handwritten Arabic words [10]. The system uses Hidden Markov Model (HMM) for word recognition, and is based on grapheme recognition without explicit segmentation. The most important requirement for the development and comparison of recognition systems is a large database combined with ground truth information. Compared to English text, where handwritten words and numbers have been publicly available for a long time, in the case of Arabic handwritten words many papers use a specific, small data set of their own, or they present results on large databases that are not available to the public, it follows that it is really hard to compare different results which would be important for improving existent methods. In recent years, methods based on HMM particularly, have been successfully used for recognizing cursively handwritten words. The great difference in the shape of handwritten graphemes between Latin and Arabic requires not only a modification and adaptation of the preprocessing and feature extraction process to the characteristics of the Arabic writing, but also that the HMM must be adopted to Arabic handwriting style, along with post-processing that uses language dependent syntax and semantics.

Khémiri et al. also proposed a method for the recognition of handwritten Arabic words that is based on horizontal-vertical HMM and Dynamic Bayesian Network Model [11]. Their goal was to reach an automatic offline recognition system of Arabic handwritten words based on Probabilistic Graphical Models (PGMs), where a PGM is a diagrammatic representation of probability distribution. In this graphical model, nodes express random variables and links represent probabilistic relationships between variables.

Kurniawan et al. compared contour based and non-contours based techniques for extracting words from unconstrained handwritten text lines [12]. Their approach is based on contours of the words rather only considering threshold for inter-word gaps. Contour of each word is examined along with threshold for inter-word gaps to extract words with high confidence. Preprocessing technique, normalization is not applied, that enhance the speed significantly unlike other techniques. Another simple technique for punctuation detection is proposed to increase accuracy of word extraction.

Gomathi Rohini et al. proposed a method for unconstrained handwritten word segmentation [13]. The input text line image was preprocessed and the text lines were segmented into words. Vertical projection profile was integrated with the system to find distance metrics. Following this, gaps were classified by threshold estimation. The proposed system located accurately the words in text lines.

Chatelain et al. designed a system dedicated to the extraction of numerical data in unconstrained handwritten documents. They proposed a method for discriminating the connected components, using different families of features and a combination of classifiers [14]. Heutte et al. also developed handwritten text recognition architecture, and showed how to endow the reading system with learning faculties necessary to adapt the recognition to each writer's handwriting [15].

Our deciphering method [16] is composed of two main parts, the first is a preprocessing algorithm that filters out the less probable interpretations of the inscription, and the second part is the deciphering algorithm. The deciphering algorithm determines the sound values of the right age-related versions of the graphemes in the well-known historical abecedaries from the visualization of the analyzed graphemes. The deciphering algorithm is based on the extraction of topological parameters of the graphemes [17, 18]. In other words, the deciphering algorithm identifies the topological components of the examined inscription and evaluates them using the well-known SHR abecedaries with their canonical forms. It is noteworthy that besides our topological parameter-based method there are several approaches to describe the topology of the graphical objects, including the contour-based shape representation techniques [19].

The developed deciphering method was implemented in the Software for Inscription Deciphering (SID). The SID was applied for the *Székely-Hungarian Rovash* (SHR) script that is a traditional writing system of the Hungarians, used by mainly the Székelys, a Hungarian-speaking ethnic group in Romania. SHR is part of the Rovash (pronounced "rove-ash") script family [20]. The rovash scripts are closely related writing systems that are used in the Eurasian Steppe by several nations and tribes up to the 11th or 12th century AD, and in the Carpathian Basin mainly by Hungarians up to the present time. However, our algorithm is universal and can be used for other languages by changing the topological parameters of the examined symbols and using the appropriate dictionary of the implied language.

The article presents a new preprocessing algorithm that improves the performance of the whole deciphering method. Section 2 introduces the earlier ('normal') preprocessing algorithm, then the Section 3 details the new preprocessing algorithm, and Section 4 presents the results of the improvements implemented in the software SID. Section 5 summarizes the conclusions.

2. The normal preprocessing algorithm of the deciphering method

The flowchart of our deciphering method is presented by Fig. 1, where the steps of the normal preprocessing algorithm separated with dotted line. Each grapheme is handled as individual object that has so-called grapheme name, which is used as ID. For the simplicity, we restrict our description to the one word long undeciphered inscription that is called *Word under Test* (WuT). The type of the input of the algorithm should be a picture or a script document; however, the type of the input document is not significant, because the topology parameters of the WuT were extracted and uploaded them into the database manually. In the presented approach, OCR tools were not used.

The notations in the flowchart of Fig. 1 are corresponding to the notations introduced in the article.
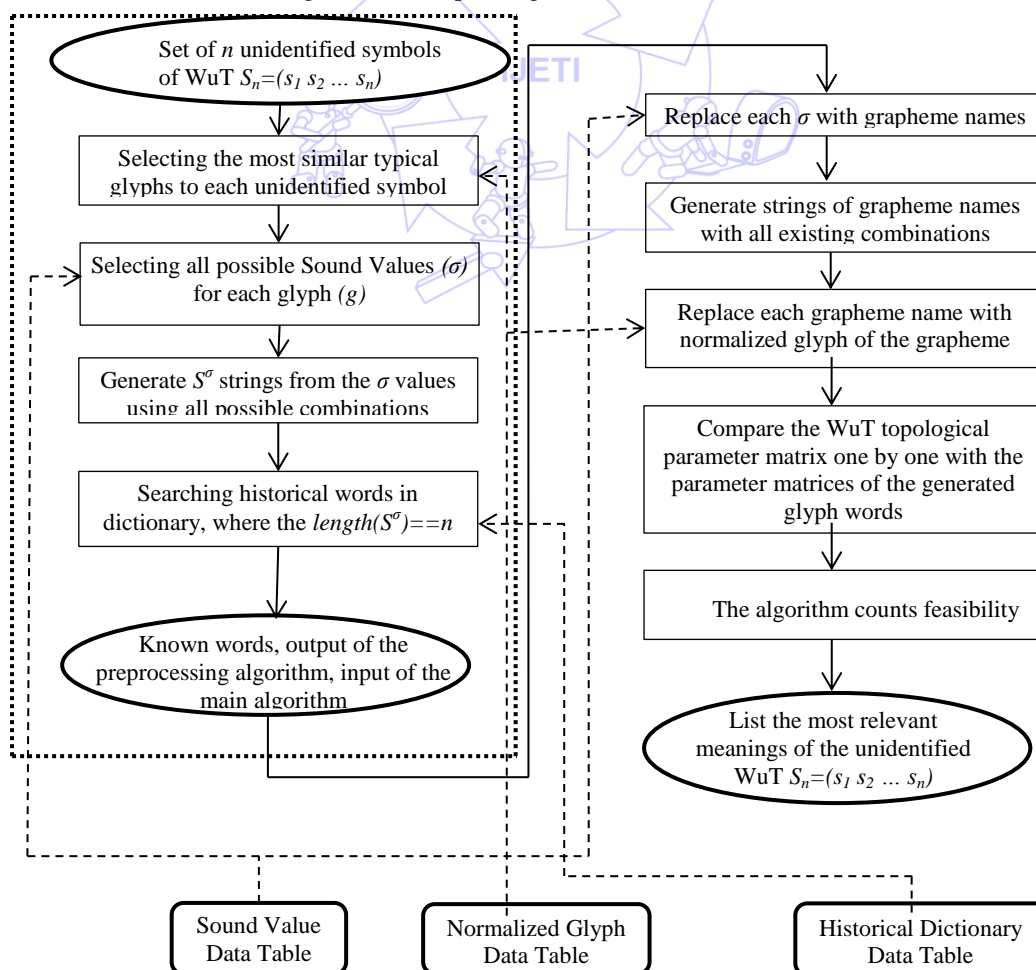

Fig. 1 The deciphering algorithm with the normal preprocessing procedure

Step 0 is making a drawing of the undeciphered inscription that is usually written from the right to the left; however, in certain cases the writing direction is from the left to the right.

Step 1 is the segmentation of the inscription into symbols described by the string of symbols S_n, as Eq. (1) presents.

Step 2 is optional, changing the writing direction from the most common "right to the left" to "left to the right" direction. In general we can determine the writing direction according to the orientation of the symbols/graphemes.

Step 3 extracts the topological parameters of S_n. All topological features belonging to the known glyphs will be extracted with this method.

Step 4 finds the k closest known glyphs (G_n^k) to S_n based on the similarity metrics of topological parameters as Eq. (1) presents. This step is using the Euclidean distance that compares two topological parameter vectors as shown in Fig. 2 [21].

$$S_n = \begin{pmatrix} s_1 & s_2 & \cdots & s_n \end{pmatrix},$$

$$G_n^k = \begin{bmatrix} g_{11} & g_{21} & \cdots & g_{n1} \\ g_{12} & g_{22} & \cdots & g_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ g_{1k} & g_{2k} & \cdots & g_{nk} \end{bmatrix} \tag{1}$$

Topological features	Unidentified symbol (s_n):	Normalized known glyph (g_{nk}):
Number of loops	0	0
Number of vertical lines	1	1
Number of horizontal lines	0	0
Number of endpoints	2	2
etc.	…	…

Fig. 2 Topological parameters of an unknown symbol and a normalized known glyph

Step 5 assigns *Sound Values* (σ) that belongs for each g_{ij} glyph in G_n^k, represented with International Phonetic Alphabet (IPA) value, see Eq. (2). When the algorithm finds the same σ belonging to the different glyphs, it will take into consideration only one of them.

$$g_{ij} = \left\{ \sigma_1, \sigma_2, \ldots, \sigma_{q_{max}} \right\}, i \leq n, \ j \leq k; \tag{2}$$

Step 6 generates strings (S^σ) from the possible combinations of σ values, see Eq. (4). N^σ is the number of the strings generated from σ values, see Eq. (3).

$$N^\sigma = \left(q \cdot k \right)^n, q \leq q_{max}, k \geq k_{min}, n \leq n_{max}, \textit{ where } q, k, n \in \mathbb{N}^+; \tag{3}$$

where q is the set of the σ numbers that belong to the actual g_{ij}, and n is the number of graphemes of WuT, and $\mathbb{N}^+$ is the set of the positive integers. The maximum values of the numbers $q_{max}=3$ and $n_{max}=14$ are specific Hungarian language features; obviously for other languages these numbers can be different. According to Eq. (3), the N^σ is a power function, where the exponent is the n, which is determined by the length of WuT. The base comes from ($q\,k$), which is the variable. As a result, if the k is large, the running time of the algorithm becomes longer and the algorithm needs too much resource (memory,

CPU), besides there is a chance that it does not give positive answer on the output in case the algorithm cannot find the closest known glyphs for symbols. The normal preprocessing algorithm gives positive hits only if the generated S^σ strings are exactly the same as the historical words W^σ stored in the dictionary. This causes why we need to set the k to be large enough to get positive results; therefore, $k_{min}=5$.

If we calculate manually the all possible values of the formula (3), we get an inconceivably and impractically large number of combinations. Although in the reality we cannot reach the q_{max} theoretical maximum, because usually less than q_{max} different sound values belong to one g_{ij} and the most cases we try to decipher shorter than n_{max} words. In practice we are working with manageable levels, and the probability that we run into this theoretical worst case number is very low, however in some cases we got long running times.

Since we applied our method for the SHR script, which was almost exclusively used for Hungarian language; and we examine historical inscriptions; therefore, a historical Hungarian dictionary is used. Each word in the historical dictionary is transformed into a phonetic form that is represented with the sound values of each grapheme of the word expressed with IPA values. We have one set of historical Hungarian words in dictionary (W^σ) where the number of the stored words indicates y see Eq. (4).

$$S^\sigma = \begin{bmatrix} S_1^\sigma \\ S_2^\sigma \\ \vdots \\ S_{N^\sigma}^\sigma \end{bmatrix}, W^\sigma = \begin{bmatrix} W_1^\sigma \\ W_2^\sigma \\ \vdots \\ W_y^\sigma \end{bmatrix} \tag{4}$$

In the Step 7 the results (R^σ) of the preprocessing algorithm returns the p intersection of the set of the generated sound value strings (S^σ) and the dictionary (W^σ). Eq. (5) presents the result of the Step 7 that is the final result of the preprocessing algorithm.

$$R^\sigma = S^\sigma \cap W^\sigma, R^\sigma = \begin{bmatrix} R_1^\sigma \\ R_2^\sigma \\ \vdots \\ R_p^\sigma \end{bmatrix} \tag{5}$$

3. The improved preprocessing algorithm

The improved preprocessing algorithm contains the following steps. Steps 0, …, 4 are identical to the normal preprocessing algorithm described above, see in Fig. 1. Step 5 collects the *Transcription Value* (τ, denoted typically one letter, sometimes a composition of more letters) that belongs to each glyph g_{ij}, see Eq. (6).

$$g_{ij} = \{\tau_1, \tau_2\}, i \le n, j \le k; \tag{6}$$

Step 6 generates strings of the τ values of the combinations of WuT symbols (S_n), see Eq. (7).

$$S^{\tau} = \begin{bmatrix} S_1^{\tau} \\ S_2^{\tau} \\ \vdots \\ S_{N^{\tau}}^{\tau} \end{bmatrix}, W^{\tau} = \begin{bmatrix} W_1^{\tau} \\ W_2^{\tau} \\ \vdots \\ W_y^{\tau} \end{bmatrix} \tag{7}$$

The improved preprocessing algorithm implemented in the software SID uses a database management system to store the necessary information in following data tables. The *Sound Value Data Table* is the set of σ values that belongs to each grapheme. The *Normalized Glyph Data Table* stores the set of glyphs with topology attributes. The *Historical Dictionary Data Table* contains the set of words recorded with phonetic symbols and transcription values as well.

The improved algorithm uses the same dictionary as the normal preprocessing algorithm described in previous section. The only difference is that in the normal preprocessing algorithm the historical words were denoted by sound values (W^{σ}), and the improved algorithm uses the same set of historical words represented with transcription values (W^{τ}).

The run time of the preprocessing step highly depends on the actual input values; however, the improved preprocessing algorithm is at least three-time faster than the normal one. As a practical consequence of the decreased run-time of the software SID with using the improved preprocessing algorithm, the value of k_{min} can be decreased, cf. Eq. (3).

Now q is the set of the τ numbers that belong to the actual glyph. n is the number of symbols of WuT. N^{τ} is the number of the strings generated from τ values, see Eq. (8).

$$N^{\tau} = (q \cdot k)^n, q \leq q_{\max}, k \leq k_{\min}, n \leq n_{\max}, where\ q,k,n \in \mathbb{N}^*; \tag{8}$$

where $q_{max}=2$, $k_{min}=3$. We successfully decreased the multiplication ($q\,k$), what is the base of the power function, the result of which is that the N^{τ} is lower than the N^{σ}.

The normal preprocessing algorithm was able to identify any word of historical dictionary in case of exact match only. However, in certain cases, any generated sound value string of the set (S^{σ}) is not identical to any of the equally long words of the historical dictionary (W^{σ}), but there is at least one member of S^{σ} that is similar to one member of W^{σ}. To handle such situation, a further improvement was implemented in the improved preprocessing algorithm. We redesigned the Step 7 of the normal preprocessing algorithm where at this time we calculates the similarity of the generated S^{τ} strings and the W^{τ} words from dictionary. If this value reaches a predefined similarity threshold (*sth*), the actual words will be selected from the dictionary. This *sth* defined the allowed number of differences between the generated S^{τ} string and the W^{τ} words of dictionary. The final result of the improved preprocessing algorithm is in Eq. (9).

$$\mathrm{R}^{\tau} = similarity(S^{\tau}, W^{\tau}) \tag{9}$$

The similar text function we called *similarity(S^{τ},W^{τ})* is used for comparison of two strings and is coming from the text similarity algorithm from [22]. Both parameters of the function are required. Namely, the first parameter specifies the generated S^{τ}, and the second parameter specifies W^{τ} to be compared. The return value of the function is the number of matching τ values of two strings.

The number of the results from the improved preprocessing algorithm depends from the predefined *sth*, what the user can change in the user interface of the software SID before running. If this *sth* is quite large the preprocessing algorithm gives many hits, even if they are not enough relevant; otherwise, few precise hits will return in the output. As we see in the

section Results, the improved preprocessing algorithm gives positive hits in case the generated S^τ are not exactly the same as the W^τ word found in the dictionary.

4.   Results

In this section we provide the difference between the normal algorithm and the improved one using the software SID, where both preprocessing algorithms were implemented. The example for a WuT is presented in Fig. 3. This inscription is part of a two-page long Székely-Hungarian Rovash text that was found in a Bible. Samuel Patakfalvi got this bible in 1941; the inscription should be made between 1776 and 1785. The two-page long text was deciphered; however, it contained some symbols that were unknown before. In the Step 0 of the preprocessing algorithm we made a drawing of the inscription.

Fig. 3 An example SHR inscription [17]

In the Step 1, the segmentation of the symbols of the inscription was prepared manually, see in Fig. 4.

Fig. 4 Separation of the symbols

In the Step 2 the writing direction was changed from right to the left to left to the right direction, see Fig. 5.

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
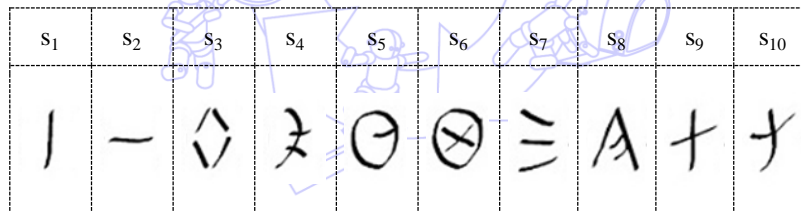
Fig. 5 Changing the writing direction

In the Step 3 the topological parameters of the symbols are extracted; the example is presented in Fig. 6.

s_1 **Topological features of s_1**
Number of loops: 0
Number of vertical lines: 1
Number of horizontal lines: 0
Number of endpoints: 2
…

s_5 **Topological features of s_5**
Number of loops: 1
Number of vertical lines: 0
Number of horizontal lines: 1
Number of endpoints: 1
…
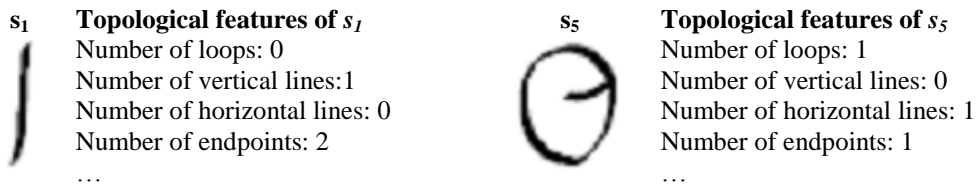
Fig. 6 Definition of the topological parameters of each symbol

After the preparation of the input of the preprocessing algorithm, the topological features will be loaded for each glyph through the graphical user interface of the SID.

In the Step 4 the parameter k is set to *2*. The *sth* is set to *0* in the first trial, which represents the normal version of the preprocessing algorithm. As we found out, the output of the preprocessing algorithm could not find the exact match between the generated $N^\sigma=5184$ element of S^σ and the $y=142$ pieces of W^σ from dictionary, which contained $n=10$ graphemes.

For example, the symbol "-" used in the SHR relic "Patakfalvi Bible" was unknown until the "Patakfalvi Bible" was discovered. If this symbol is given as an input with unknown sound value of our algorithm, we can find that the order number of the symbol "-" is greater than k. Therefore, the normal preprocessing algorithm will not find similar glyphs in the similarity queue that means, in case k is set to 2, it will not be able to find the appropriate Hungarian word in the dictionary, because the normal preprocessing algorithm was prepared only to find the exact matches in the set of words.

Fig. 7 Result of the improved preprocessing algorithm when $sth = 2$

Another case we show the improved preprocessing algorithm, in Fig. 7 the sth is set to *2*, and in Fig. 8 is set to *3*. In the first case the method found *2* matches and the last case it found *6* matches in the dictionary. By using the improved preprocessing algorithm the deciphering software SID found the proper decipherment of the input inscription with both of these two settings.

The *Hit Rate* of the deciphering algorithm is different in the examples in Fig. 7 and Fig. 8 47.7% and 60.8%, respectively. Its reason is that the *sth* value is different (2 and 3, respectively) in each case.

Fig. 8 Result of the improved preprocessing algorithm when *sth* = 3

As the above example presented, in this case, the software SID running the normal preprocessing algorithm did not find any match in the Hungarian dictionary; however, by using the improved preprocessing algorithm, the SID found the expected word from the dictionary and gave a probabilistic estimate for the meaning and interpretation goodness of the inscription.

The improved preprocessing algorithm became more robust than the normal one. The test runs show that we do not need to set the *k* to a large value; we get positive results even if the *k* is set to 1. This caused that the running time of the

preprocessing algorithm became shorter and the algorithm does not need to generate large set of strings of transcription values (S^τ). The preprocessing algorithm produces results with the most relevant hits on the output of the software SID.

5. Conclusions

The article presented the different preprocessing algorithms of the deciphering method for historical inscriptions of unknown origin, namely the normal preprocessing algorithm and the improved one with two novel accelerator methods to increase the efficiency. The most outstanding advantage of the normal preprocessing algorithm is its accuracy. However, it has two weaknesses *(i)* in some cases it can be very slow and *(ii)* it is not robust enough, in case it interpreted poorly one of the symbols. We accelerated the preprocessing phase of the deciphering method in such way that at first we initiated the transcription values instead of sound values and then strengthened the comparison method to find the relevant hits from the dictionary even if some of them had wrong interpretation of the symbols. Test runs demonstrated that the improved preprocessing algorithm is significantly more effective than the normal one. Although the featured software was optimized for Székely-Hungarian Rovash script, the developed algorithm is universal and can be used for other scripts and languages by changing the Hungarian dictionary to a different language type and the topology coordinates of the another script' graphemes have to be set in the database.

References

[1] G. Hosszú, "The Rovas: A special script family of the central and eastern European languages," Acta Philologica (Wydział Neofilologii Uniwersytet Warszawski, Warszawa), vol. 44, pp. 91-102, 2013.

[2] L. Eikvil, Optical Character Recognition, Oslo: Norsk Regnesentral, 1993. (online) Access date June 21, 2015, http://bkreaders.ru/books/OCR.pdf.

[3] G. Hosszú, "Mathematical statistical examinations on script relics," in Data Mining and Analysis in the Engineering Field, 1st ed., V. Bhatnagar, Ed. Hershey, New York: Information Science Reference, 2014, pp. 142-158.

[4] G. Hosszú, "A novel computerized paleographical method for determining the evolution of graphemes," in Encyclopedia of Information Science and Technology, 3rd ed., M. Khosrow-Pour, Ed. Hershey, New York: Information Science Reference, 2015, pp. 2017-2031.

[5] T. Hassner, M. Rehbein, P. A. Stokes, and L. Wolf, "Computation and palaeography: potentials and limits (Dagstuhl Perspectives Workshop 12382)," Dagstuhl Reports, vol. 2, no. 9, pp. 184-199, 2012.

[6] B. Gottfried, M. Wegner, and M. Lawo, "Towards the interactive transcription of handwritings: anytime anywhere document analysis," Int. J. on Document Analysis and Recognition (IJDAR), vol. 18, no. 1, pp. 31-45, March 2015.

[7] M. Panagopoulos, P. Rousopoulos, D. Arabajis, M. Exarhos, and C. Papaodysseus, "Methods and algorithms for the automatic identification of writer of ancient documents," Proc. 1st Conf. on Computer Applications and Quantitative Methods in Archaeology Greek Chapter (CAA-GR), Rethymno, Crete, Greece, 2012, pp. 153-158, March 2014.

[8] E. Kavallieratou, K. Sgarbas, N. Fakotakis, and G. Kokkinakis, "Handwritten word recognition based on structural characteristics and lexical support," Proc. Int. Conf. on Document Analysis and Recognition, IEEE Press, Aug. 2003, vol. 1, pp. 562-566.

[9] S. Singh, "Shape detection using gradient features for handwritten character recognition," Proc. 13th Int. Conf. on Pattern Recognition, IEEE Press, August 1996, vol. 3, pp. 145-149.

[10] V. Märgner, H. El Abed, and M. Pechwitz, "Offline handwritten Arabic word recognition using HMM – a character based approach without explicit segmentation," Actes du 9ème Colloque International Francophone sur l'Ecrit et le Document, pp. 259-264, Sept. 2006.

[11] A. Khémiri, A. Kacem, and A. Belaïd, "Towards Arabic handwritten word recognition via probabilistic graphical models," Proc. 14th Int. Conf. on Frontiers in Handwriting Recognition, Heraklion, IEEE Press, Sept. 2014, pp. 678-683.

[12] F. Kurniawan, A. R. Khan, and D. Mohamad, "Contour vs. non-contour based word segmentation from handwritten text lines: an experimental analysis," International Journal of Digital Content Technology and its Applications vol. 3, no. 2, pp. 127-131, Jan. 2009.

[13] S. Gomathi Rohini, R. S. Umadevi, and S. Mohanavel, "Statistical approach for segmenting unconstrained handwritten text lines." IJCA Proc. Amrita Int. Conf. of Women in Computing (AICWIC'13). IJCA Journal, Jan. 2013, pp. AICWIC(1):21-24.

[14] C. Chatelain, L. Heutte and T. Paquet, "A syntax-directed method for numerical field extraction using classifier combination," Proc. Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), 2004, Tokyo, Japan, 26-29 Oct. 2004, pp. 93-98.

[15] L. Heutte, A. Nosary, T. Paquet, "A multiple agent architecture for handwritten text recognition," Pattern Recognition, vol. 37, no. 4, pp. 665-674, 2004.

[16] L. L. Tóth, R. Pardede, and G. Hosszú, "Novel algorithmic approach to deciphering rovash inscriptions," in Encyclopedia of Information Science and Technology, 3rd ed., M. Khosrow-Pour, Ed. Hershey: Information Science Reference, 2015, pp. 7222-7233.

[17] L. L. Tóth, R. E. I. Pardede, G. A. Jeney, F. Kovács, and G. Hosszú, "Application of the cluster analysis in computational paleography," in Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering, 1st ed., P. Samui, Ed. Hershey: Engineering Science Reference, 2016, pp. 525-543.

[18] N. A. Khan, "A shape analysis model with application to character and word recognition," Ph.D. Dissertation, Technische Universiteit Eindhoven, Eindhoven, 2000.

[19] R. Rashli, Z. Zulkoffli, E. A. Bakar, and M. S. Soaid, "A study of 3D CAD model and feature analysis for casting object," International Journal of Engineering and Technology Innovation, vol. 2, no. 2, pp. 138-149, 2012.

[20] G. Hosszú, Heritage of Scribes. The relation of rovas scripts to Eurasian writing systems, 2nd ed. Budapest: Rovas Foundation, 2012.

[21] S. Theodoridis and K. Koutroumbas, Pattern recognition, 2nd ed. San Diego: Elsevier, 2003.

[22] I. Oliver. Programming classics - implementing the world's best algorithms. Prentice Hall, 1994.