IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 February 2024, accepted 29 March 2024, date of publication 4 April 2024, date of current version 11 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3385113

RESEARCH ARTICLE

Polish Word Recognition Based on n-Gram Methods

PIOTR WOJCICKI AND TOMASZ ZIENTARSKI
Department of Computer Science, Lublin University of Technology, 20-618 Lublin, Poland
Corresponding author: Piotr Wojcicki (p.wojcicki@pollub.pl)

This work was supported in part by the Computer Science Discipline, Lublin University of Technology, under Grant FD-20/IT-3/010 and Grant FD-20/IT-3/028.

ABSTRACT Word recognition of Slavic languages is not an easy task due to the complicated declension of words and a variety of diacritical signs. Polish is a representative of West Slavic languages, which are written in Latin characters. Automatic handwritten word recognition in Slavic languages is not easy, due to the poor recognition rate of letters with diacritical signs and lack of good handwritten text corpora for languages with declension. The main aim of the research is to investigate the possibility of correcting typos made in the final phase of recognizing Polish. The method developed is based on letter recognition by means of convolutional neural networks (CNNs) and text matching algorithms for resulting words. At the first stage, we use a designed convolutional neural network for character recognition. At the second stage, after combining letters into words we apply a post-processing error correction method, which improves the efficiency of recognition of the misspelled words. We checked the efficiency of word matching for a few measures of similarity of words, i.e: edit distance (Damerau-Levenshtein), string matching (Sorensen-Dice) and list of candidates. In addition, we examine how word length and the number of misplaced letters affect the behaviour of the algorithms used. The analysis is carried out for bigram and trigram methods. By combining different methods to assess the similarity of words, better selection of lists of proposed words has been achieved. The article proposes an innovative method for correcting post-processing errors in recognizing Polish words with the efficiency of correct word matching ranging from 76% to 99%, depending on the measure and word length used.

INDEX TERMS Error detection, natural language processing, optical character recognition, Slavic languages, text matching.

I. INTRODUCTION

Handwritten word recognition is a task which is sometimes performed with the use of deep learning algorithms such as convolutional neural networks. To improve word recognition efficiency in computer systems, very often matching a dictionary text is made. Both methods are complementing each other in natural language processing tasks.

The letters of West Slavic languages are based on the Latin alphabet, but they also include characters with diacritical signs. For example, in the Polish language there are nine additional lowercase letters generated with diacritical signs and as many additional capital letters. Automatic handwritten word recognition in Slavic languages is not easy, due to the poor recognition rate of letters with diacritical signs and lack of good handwritten text corpora for languages with declension. In the paper [1] convolutional neural networks (CNN) are used to develop an optical character recognition (OCR) system in Latin-based text containing diacritics. As part of the work, a Polish Handwriting Database (PHCD) was created. It consists of 530,000 images of handwritten characters written by over 2,000 people, including students, graduates and lecturers. The created open access database, which can be used free of charge for academic purposes such as further research, is available at: https://cs.pollub.pl/phcd/?lang=en. The CNN used in this project achieved efficiency in the range

The associate editor coordinating the review of this manuscript and approving it for publication was Alicia Fornés.

of 91% to 97%, depending on the length of the word and the type of characters.

The main goal of this work is to propose a method for post-processing error correction, which improves the effectiveness of recognizing misspelled words produced by the CNN for Polish texts [1]. In this method, dictionary text matching is sought after combining letters into words. Construction of proposed spelling lists using the n-gram model was carried out with the string matching algorithm and edit distance metrics algorithm for every detected word. In addition, the relationship between the word length and the number of misplaced letters was checked, as well as the impact on the behavior of the algorithms used. The analysis was carried out for bigram and trigram methods. By combining different methods to assess the similarity of words, we have achieved a better selection of lists of proposed words.

The main aim of our study can be divided into the following steps:

- Composing words from letters.
- Finding a word in the dictionary.
- Construction of proposed lists using the algorithms disscussed.

The rest of this paper is organised as follows. In Section II, the authors refer to related works, Section III contains the research methodology divided into subsections. The research results and discussion are presented in section IV. Conclusions and an outline of future work follow in Section V.

II. RELATED WORK

Character recognition is a task where artificial intelligence algorithms can be used [2]. For handwritten character recognition (HCR) scientists use artificial neural networks (ANNs), the Support Vector Machine (SVM) and the Hidden Markov Model (HMM) [3], [4], [5], [6]. Lately the best results for HCR have been obtained with the type of ANNs called convolutional neural networks. Popko and Weinstein in [7] present a method for recognizing handwritten digits with the use of CNNs, in which the integrated fuzzy logic module based on a structural approach was developed. A high recognition rate of 99.23% was achieved.

However, such good efficiency is not achievable in the case of Slavic language characters. Martinovska et al. in [8] obtained on average a 76% efficiency of recognition of handprinted characters. The idea of recognizing the Slavic characters in manuscripts with the use of CNNs and Transkribus was presented by [9]. Moreover, the authors did not present the efficiency results, which suggests that this approach still needs their verification and improvement. A promising outcome was obtained by Kacalak and Majewski in [10]. Their recognition efficiency of the ANN was about 95%, but for a much easier task, i.e. handprinted Polish characters. Grzelak et al. in [11] used a modified EMNIST database for character recognition. They generated Polish letters with the LeNet5 neural network and added them to the EMNIST database. The efficiency of recognizing the Polish letters ranged from 43% to 87%.

Natural language processing (NLP) is an interdisciplinary field, combining issues of artificial intelligence and linguistics, dealing with the automation of analysis, understanding, translating and generating natural language. N-grams have a great potential in language processing. Combined together with other methods they have many various applications, like: spell checking (e.g. in search engines) [12], [13], word correction [14], [15], text categorization [16] or word based sentiment classification [17]. One advantage of the n-gram method is that it is language independent. Moreover, it is stated that implementing n-grams into the auto-correlation approach improves accuracy [18].

A hybrid spell checking methodology for isolated word error correction with high accuracy is presented in [12]. This approach uses unigrams for spellings with less than four characters, bigrams for four to six characters and trigrams for spellings with more than six characters. An algorithm for a language-independent spell checker based on the n-gram model for English and Portuguese is presented in [13].

There are many methods of spell checking errors dedicated to the Bengali language [19]. A system for checking the spelling of an English word transliterated to Bengali, based on Levenshtein distance and unigram is presented in [18]. Another solution for Bengali in the form of smart spell checking is shown in [20]. The algorithm proposed in [21] is a combination of n-gram characters with a neural network. N-grams and recurrent neural network (LSTM) are used for spell checking of the Punjabi language [22] and for the spelling correction process in Turkish [14].

A framework involving word correction for Indian languages with a varying degree of inflections is presented in [15]. It is based on the dictionary. The incorrect words are gathered from the OCR document. The n-gram technology (where n equals 2 or more) and error detection are used. The efficiency of this solution varies for different Indian languages. Error detection is in the range 64% to 85%.

Errors occurring in the Arabic language can also be detected using the system presented in [23]. After detecting an error, the system generates a list of probable corrections. A list of candidates is created on the basis of the Levenshtein distance and occurrence in a given dictionary. In this solution the best score (82.86%) was achieved for the automatic correction using the bigram language model with a candidate cut-off limit of 3.

The performance of the error correction module for the Tamil language is presented in [24]. The authors use n-grams (where n equals 2 and 3), stemming and the hash-table technique in their implementation. Various words were taken into consideration. The described tool generates words with the accuracy of 95 percent.

The comparison between n-grams and syntactic n-grams using three classifiers (SVM, Naive Bayes, and J48) is described in [25]. The work [26] presents a syntactic

TABLE 1. Comparison of selected works on character recognition with error detection.

Reference	Dataset	Model	Performance metrics
Popko et al. [7]	handwritten digits	CNN	recognition rate 99.23%
Martinovska et al. [8]	handprinted characters	Decision tree	recognition efficiency 76%
Kacalak et al. [10]	handprinted Polish characters	ANN	recognition efficiency 95%
Grzelak et al. [11]	handwritten Polish characters	CNN	recognition efficiency from 43% to 87%
Saluja et al. [15]	different Indian language	n-gram	error detection from 64% to 85%
Attia et al. [23]	Arabic characters	Candidate list	error detection 82.86%
Sakuntharaj et al. [24]	Tamil words	n-gram	accuracy 95%

similarity framework that can be used to match a short text. The algorithm for context-sensitive word correction are presented in [27] and [28], and for correcting real errors in [29]. The paper [30] describes a system for automatic recognition of common Arabic handwritten words.

Pang et al. in paper [31] proposes for text matching the approach of image recognition and apply the deep neural network architecture. The tools presented are used for two tasks: firstly checking of the semantic compatibility of two sentences and then for the paper citation matching. In paper [32] a handwriting recognition method based on template matching has been proposed. The knowledge base consists of typed or handwritten numbers, punctuation and the uppercase and lowercase letters of the English alphabet. Single character images in the knowledge base are used to generate correlation values for the input character image and the character output text. The accuracy of the tested system was 90%. It was pointed out that the OCR system performance unit is independent and fixed for handwritten and typed images of various sizes.

Little research has been done on the Polish language by using the n-gram method, because its handwritten characters are potentially more difficult than English ones and there are no sufficient corpora for studying them [33], [34]. Table 1 summarizes related research works on character recognition with error detection.

III. METHODOLOGY

The n-gram model has so far been applied in many linguistic problems such as spelling corrections, speech recognition and word sentence prediction. The n-gram is a probabilistic method originally proposed by Markov [35] and later applied by Shannon [36]. It is a sequence of consecutive units such as words, phonemes, sounds, syllables and letters. Depending on the number of elements, the following names are used: unigram for one-element n-grams, bigram for a sequence of two elements and then a trigram - referring to a sequence of three elements. For the word paper we will get 4 bigrams (pa;ap; pe;er) and 3 trigrams (pap;ape; per).

A. STRING MATCHING ALGORITHMS

In the spelling correction tasks an n-gram is the sequence of n characters in a word and can be used to measure similarity of two strings. The more similar n-grams exist between words, the more similar the words will be. The similarity coefficient (γ) can be defined by the following equation [37]:

$$\gamma = \frac{\alpha \cap \beta}{\alpha \cup \beta} \tag{1}$$

where α and β are n-gram sets of two words a and b. The expression $\alpha \cap \beta$ denotes the number of similar n-grams, and $\alpha \cup \beta$ denotes the number of unique n-grams in α and β.

For a string similarity measure, the Sorensen-Dice (SD) similarity coefficient Γ may be calculated from the equation [37]:

$$\Gamma = \frac{2n_t}{n_x + n_y} \tag{2}$$

where n_t is the number of character n-grams found in both strings, n_x is the number of n-grams in string x and n_y in y, respectively. For example, for the words *computer* and *computations*, the Sorensen-Dice coefficient given for bigrams will be 0.588. The word contains 7 bigrams: *co;om;mp;pu;ut;te;er*, but for *computations* we have 10 bigrams. Both words have 5 common bigrams.

B. LIST OF CANDIDATES

Another method based on the string similarity matching is the so-called list of candidates (TOP) [28]. This method produces a list of probable spelling corrections for the misspelled words detected. In the first step, the misspelled word is broken down into ngrams (e.g. floder, original word: flower), giving the bigram $\Gamma_n = fl; lo; od; de; er$.

TABLE 2. List of candidate spelling words. Original word: *flower*. Misspelled word: *floder*, bigram: *fl*, *lo*, *od*, *de*, *er*.

Candidate list	Number of shared bigrams fl, lo, od, de, er
flower	3
floater	3
airflow	2
belonger	2
unicode	2
blowfish	1
colours	1
zodiac	1

Then, for each n-gram from Γ_n, the words in the dictionary containing the selected n-gram are searched. Next, we count the number of shared n-grams contained in words. The result obtained is hardly readable, and finally, after sorting, the record containing the words with the largest number of shared n-grams is returned. For example, Table 2 shows results

for the word *floder* sorted by the largest number of shared n-grams.

C. MINIMUM EDIT DISTANCE ALGORITHMS

Most often, spelling errors entered into the text are errors at the character level. They can be classified into four categories related to insertion, deletion, substitution, or transposition of two adjacent characters. The majority of spelling correction errors made by humans is connected with a single edit operation. Damerau showed that 80% of typographical errors are distance 1 [38]. However, it should be remembered that there are many other factors affecting errors occurring in the text, starting from the errors associated with pre-processing of the analysed text and ending with those generated by the OCR algorithm.

Based on these typical four types of errors, Damerau employs the so-called Damerau-Levenshtein string metric for calculating the minimum number of single character operations necessary to change one word into another. Simultaneously, in 1966, Levenshtein employs a distance measure algorithm which includes only three types of single edit errors, without transposition [39]. Note, that a transpositions can be replaced by a sequence an insertion followed by a deletion, so transposition are still covered by the Levenshtein metric. Both methods are commonly used in text analysis. The Damerau-Levenshtein distance (LD) $d_{a,b}(i,j)$ between two strings a and b is defined by the relation:

$$d_{a,b}(i,j) = \min \begin{cases} 0, & i,j = 0 \\ d_{a,b}(i-1,j) & +1, \quad i > 0 \\ d_{a,b}(i,j-1) & +1, \quad j > 0 \\ d_{a,b}(i-1,j-1) & +C_{sub}, \quad i,j > 0 \\ d_{a,b}(i-2,j-2) & +C_{trans}, \quad i,j > 1 \end{cases} \tag{3}$$

where the cost substitution (C_{sub}) of the i character from the word a with another character j from word b is:

$$C_{sub} = \begin{cases} 0, & a[i] = b[j] \\ 1, & a[i] \neq b[j] \end{cases} \tag{4}$$

and the cost transposition (C_{trans}) of character $a[i]$ into $b[j]$ are expressed as:

$$C_{trans} = 1, \; for \; a[i] = b[j-1] \; and \; a[i-1] = b[j] \tag{5}$$

An short example follows for the word: COMPUTE and its likely transformations:

- an additional letter is inserted *COMBPUTE*,
- a letter is deleted *COMUTE*,
- a letter is substituted by another letter *CAMPUTE*,
- a two adjacent letters are transposed *COMPUET*

D. POST-PROCESSING

The last stage of an OCR system is the post-processing stage, whose task is to detect and correct spelling mistakes. In fact, there are two types of errors: non-word spelling errors (misspelled words) that result in non-existent words, and real-word errors that give real, but grammatically or semantically incorrect words [15], [27], [28].

For non-word spelling error detection, we use two methods, the direct one that matches a word to the dictionary, and the character-based language modelling method when a word is not available in the dictionary. A good spelling correction system needs a balance between three main components: the dictionary, error model and language model. In the error model there is a direct relationship between the number of correction candidates and the likelihood of finding the appropriate corrections.

The system consists of two parts: a Polish language dictionary database system containing over 5 million words and the part responsible for creating a list of proposed words for each misspelled words (Top10Can). The dictionary is the main component of the spell checking system as well as a reference to whether the word is correct or misspelled. It is also a reference in the event of searching for words that require correction. We created a dictionary database system based on the Polish dictionary available at: https://sjp.pl/slownik/odmiany/. This dictionary is published under the GPL 2, LGPL 2.1, Apache 2.0 and Creative Commons Attribution 4.0 International licenses. In the second part, the n-gram model joined with the two different string matching algorithms (SD, TOP) and edit distance metrics respecting various cost algorithm (LD) for misspelled words is used.

Moreover, we divided our work into a test part and evaluation part. In the test part, we needed to prepare a test set file (input file) containing a lot of words with spelling errors. The tests were carried out for a specific number of words (N_t) with a length L_z ranging from 4 to 14 characters and additionally containing from one to three incorrect replaced characters B_z. The lower word length limit is due to the use of n-grams method. For a four character word, we receive three bigrams and two trigrams. This is the absolute minimum. The selected word length range corresponds to the most common word lengths in Polish.

For this purpose, words of a given length were read randomly from the dictionary, then after a random change of characters the set was saved to file. Random placed characters contained letters and numbers. This data set was the input for the test procedure creating the proposed word list with the length equal to 10.

In the evaluation part, we used misspelled words taken from the OCR module, the input was one misspelled word. The method of creating a list of words proposed for a given misspelled word is shown in Algorithm 1. In the test part, Algorithm 1 was used to process words from the Polish dictionary with artificially entered incorrect characters. The algorithm can be summarized in the following steps:

1) Read input test file with the misspelled words.
2) Executing a query to read a record from the database containing all words with the length equal to ± 1 n-gram in relation to the length of the misspelled word.

Algorithm 1 Create Top10Can List for One Misspelled Word

Input: Single misspelled word
Output: Ten element list of suggested words

$N_G \leftarrow$ Create a list of n-grams from misspelled word
$L_z \leftarrow$ Calculate length of N_G list
$N_w \leftarrow$ Number of words from database with length $L_z \pm 1$
for $i = 1$ **to** N_w **do**
    $SD[i] \leftarrow$ store word and Sorensen-Dice value using equation 2
    $LD[i] \leftarrow$ store word and Damerau-Levenshtein value using equation 3
end for
TOP $\leftarrow$ Read from database list of words containing shared n-grams N_G and its number
SD $\leftarrow$ Sort list in descending order of Sorensen-Dice value
LD $\leftarrow$ Sort list in ascending order of Damerau-Levenshtein value
TOP $\leftarrow$ Sort list in descending order of shared n-grams number
for all words in TOP **do**
    TOPSD $\leftarrow$ calculate Sorensen-Dice value
    TOPLD $\leftarrow$ calculate Damerau-Levenshtein value
end for
for all words in TOPSD **do**
    TOPSDLD $\leftarrow$ Calculate Damerau-Levenshtein value
end for
for all words in TOPLD **do**
    TOPLDSD $\leftarrow$ Calculate Sorensen-Dice value
end for
TOPSD $\leftarrow$ Sort list in descending order of Sorensen-Dice value
TOPLD $\leftarrow$ Sort list in ascending order of Damerau-Levenshtein value
TOPLDSD $\leftarrow$ Sort list in descending order of Sorensen-Dice value
TOPSDLD $\leftarrow$ Sort list in ascending order of Damerau-Levenshtein value
return The first 10 elements of the *SD*, *LD*, *TOP*, *TOPSD*, *TOPLD*, *TOPSDLD*, *TOPLDSD* lists

3) Calculation of SD, LD values for each randomly selected misspelled words. Repetition of calculations for other B_z values.
4) Performing the next query to calculate the TOP value for different values of B_z for the same misspelled words. The searching query is executed for words with the length equal to ± 1 n-gram in relation to the length of the misspelled word.
5) For each misspelled word and calculated SD, LD, TOP values we create a Top10Can list with the most favourable values of these parameters at the beginning of the list. We also create lists containing various combinations of calculated values: TOP + SD, TOP + LD, TOP + SD + LD, and TOP+LD+SD. We finally have 7 lists of suggested words.
6) Finally, we check in which position on the list is the originally found word. The position values of the words found are in the range of 0 to 10, a value of 0 meaning that the word is not on the list of the proposed 10 words.
7) Save the obtained results: list of proposed words and word positions.
8) Repeat steps from 2 to 7 for each length of misspelled words error, L_z.

IV. RESULTS AND DISCUSSIONS

The research conducted was focused on combining the above mentioned research methods with the aim of examining their efficiency in the case of the presence of Polish characters in recognized words. In the experimental part two sets of data were prepared. The first contained data necessary to test the methods used to assess the quality of the proposed word lists. The second set contained words after the OCR process with letters with the lowest degree of recognition [1]. For all tests, we divided the words into bigram and trigram.

The first test set contained words with a length of 4 to 14 characters including one, two and three randomly entered character errors. The total set contained 9000 words for each length. The second test set contained 4000 words with letters incorrectly recognized by the CNN. The words contained only one misspelled character.

A. TESTING

The above algorithm allows calculation of SD, LD and TOP values being a measure of word similarity. A list of the most favourable values of these parameters at the beginning of the list is created. It is assumed that the test is passed if the searched word is among the first ten words on the list.

TABLE 3. The proposed list of suggested words for the misspelled word: *mate9iał*, correct word *materiał*.

Position	SD	LD	TOP
1	materiał	materiał	matematyk
2	małpiatek	materiały	małpiatek
3	materiały	materiału	materiału
4	materiału	materia	materiał
5	ałmatek	matowiał	materiały
6	materia	bateria	małolatek
7	ałmaatek	macerał	atentatem
8	diamatem	majenia	matowiał
9	matowiał	malenia	kwiaciate
10	adiaterma	maliniał	małpiatko

For example (see Table 3), for the misspelled word *mate9iał* (*material* with misspelled *r*), the correct word *materiał* (Eng. *material*) is located in the first position of the SD and LD lists, and the fourth of the TOP list. We also tried to combine several methods together to improve the results. In this case, first sorting takes place based on the calculated successive coefficient. The results obtained using algorithms

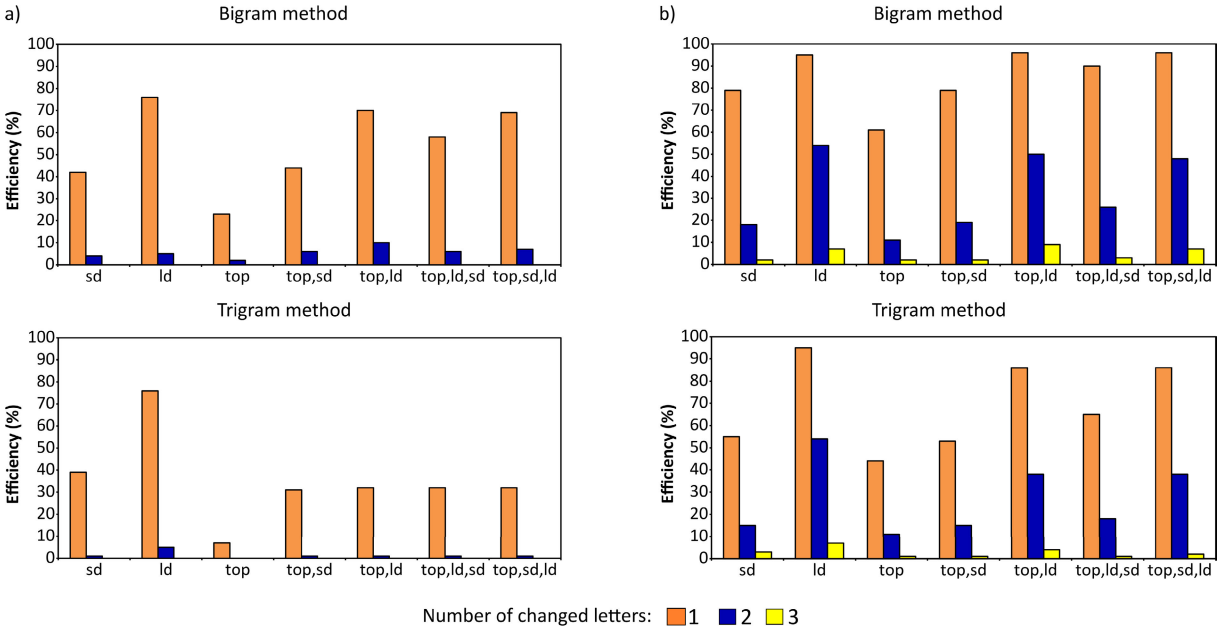

FIGURE 1. Efficiency of finding correct words for a selected combination of algorithms for a misspelled word length: a) 4, b) 6 characters respectively. The numbers of misspelled characters are given in the legend.

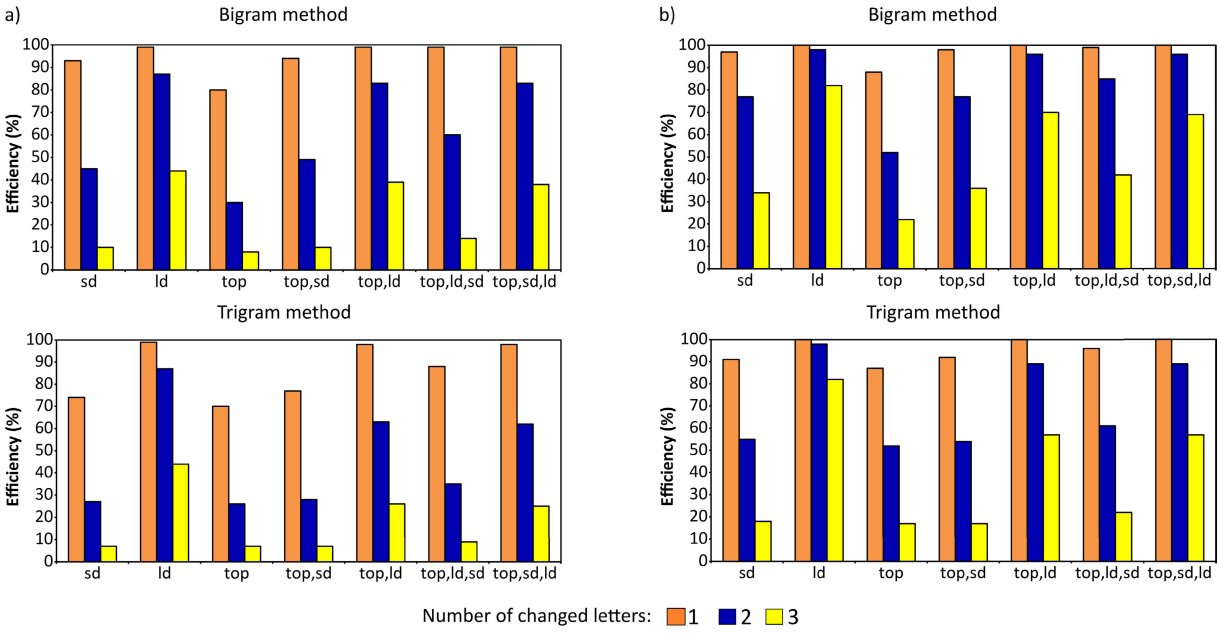

FIGURE 2. Efficiency of finding correct words for a selected combination of algorithms for a misspelled word length: a) 8, b) 10 characters respectively. The numbers of misspelled characters are given in the legend.

or their combination are considered satisfactory when their efficiency exceeds 80%.

Figures 1 – 3 show the efficiency of finding similar words for different methods and different lengths of misspelled words with a different number of misspelled characters. Each figure contains part a - word sequences acquired using the bigram method and part b - word sequences acquired using the trigram method. The efficiency of finding the correct words increases as the length of the analysed word increases, even when words contain more than one wrong character.

The results for short words based on 4- and 6-character sequences are shown in Figure 1. For this word length, using a bigram sequence will be noticeably more effective in finding the correct word than a trigram sequence with one letter changed. Changing two or three letters causes half of the characters in the word are incorrect and the efficiency of finding properly words is low. The results for words with

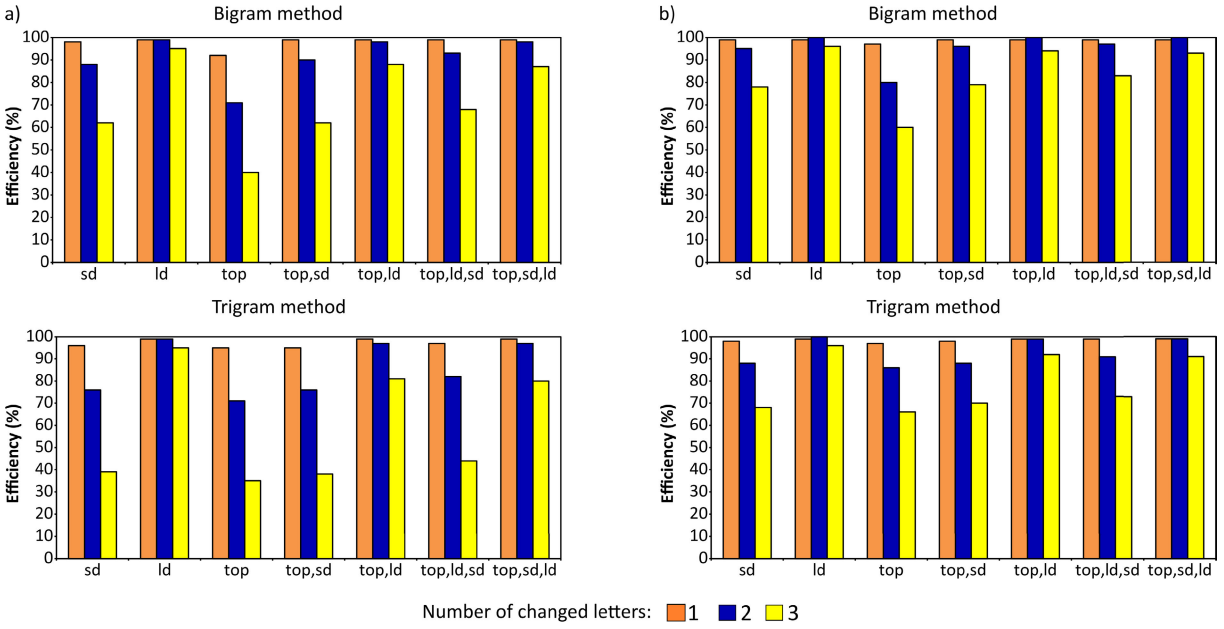

FIGURE 3. Efficiency of finding correct words for a selected combination of algorithms for a misspelled word length: a) 12, b) 14 characters respectively. The numbers of misspelled characters are given in the legend.

medium length (8- and 10-character) based on bi- and trigram sequences are shown in Figure 2. In the case of this word length, there is a clear increase in efficiency for two and three changed characters compared to short words. Dividing a word into shorter parts (bigrams) is more effective, even if altering three letters in the word, as shown in the chart. Application of selected methods (e.g. TOP, SD) after dividing a word into bigrams gives an efficiency of almost 50% when changing two characters, while the trigram method allows for less than 30% efficiency. The results for long words (12- and 14-character) based on bi- and tri-gram sequences are shown in Figure 3. The number of characters changed in relation to the total word length is smaller than in the previously considered cases. Therefore, the effectiveness of improving one or two characters is much higher. Moreover, some methods, i.e. LD, are characterized by visibly higher efficiency compared to other methods (e.g. SD, TOP or their combination) both in the case of bi- and trigrams.

The Damerau-Levenshtein method proved to be the most effective method for all cases. The algorithm gives good results even with more incorrect characters placed in the word. Moreover, the effectiveness of this algorithm does not depend on the type of a word's division into n-grams. In the second place in terms of the efficiency of the algorithm is Sorensen-Dice. Its performance also increases with increasing word length; unfortunately, it achieves much weaker performance for short words ($\approxeq 40\%$ for $B_z = 1, L_z = 4$). We observed a significant decrease of the performance for the SD method for words divided into trigrams compared to the same word divided into bigrams. For words longer than about 10 characters, the division of words into bigrams and trigrams gives similar algorithm performance. Moreover, the same incorrect character put in the word affects the SD value calculated for trigrams more significantly than bigrams. The third of these algorithms (TOP) has the lowest performance.

TABLE 4. The efficiency and time of finding Top10Can for different misspelled word lengths containing one misspelled character. Calculations of the SD and LD values were done for misspelled word length, $L_z = 4, 10, 14$ characters.

Bigram method						
Word length	4		10		14	
Method	SD	LD	SD	LD	SD	LD
Efficiency (%)	41	58	96	99	99	100
Execution time (s)	0.39	1.66	12.81	143.02	13.58	236.47
Trigram method						
Word length	4		10		14	
Method	SD	LD	SD	LD	SD	LD
Efficiency (%)	39	58	89	99	99	100
Execution time (s)	0.34	1.72	11.96	145.76	12.81	236.34

Combining several algorithms with each other did not give the expected effect, only slight increases of the total performance. On the other hand, it improved the quality of the received list of proposed words, adding the next algorithm shifting the position of the correct word towards the high position on the Top10Can list.

B. EVALUATION

In the Evaluation part of our work, we tested approximately 4000 misspelled words taken from the output OCR system. The tests were done using the LD and SD with division of words into bigrams and trigrams for word lengths equal to 4, 10 and 14 characters.

The result of this calculation is presented in Table 4. The results show the advantage of the LD algorithm over

TABLE 5. The Polish word recognition efficiency for CNN before and after the post-processing error correction method. The words contain only one misspelled character.

Word length	Word recognition efficiency for CNN	Word recognition after post-processing correction	
4	91.55%	97.97%*	94.93%**
10	96.42%	99.96%*	99.82%**
14	97.34%	99.97%*	99.97%**

*corresponds to the LD method, and ** to the SD method

the SD for short analysed words. Both algorithms achieve similar performance for long-enough words. In addition, the method based on the division into n-grams gives a better final performance, resulting in shorter execution times.

V. CONCLUSION

We proposed a post-processing error correction method used to improve the quality of misspelled words created by a neural network system. This method based on the Polish language dictionary and a character-based error n-gram model, together with the text matching algorithms, is used to construct the proposed lists of suggested words.

The effectiveness of word recognition by CNN depends on the number of misspelled characters in the word and the type of character. The artificial neural network used in the project achieved efficiency in the range from 91% to 97% depending on the length of the word (see Table 5).

The use of the Damerau-Levenshtein or Sorensen-Dice algorithm significantly increases the final recognition efficiency. It should be noted that the list of suggested words is created only when a misspelled word does not exist in the dictionary. In summary, by using the proposed method the final recognition efficiency of the entire system reaches 99.97% (see Table 5). The most important results from this study include the following:

1) The proposed algorithms were used to analyse Polish words, but they are linguistically independent and can be used for any language.
2) The quality of the list of suggested words depends strongly on the quality of the dictionary used.
3) The editing distance algorithm used to construct the list of proposed words was the most effective. It constructed good lists even for short words with a large number of incorrect characters. The Damerau-Levenshtein algorithm is n-gram length independent.
4) The Sorensen-Dice algorithm matched the performance of the Damerau-Levenshtein algorithm only for word lengths greater than 8 characters. In addition, increasing the number of errors in a word strongly affected the algorithm's efficiency. The use of the division of words into n-grams in the Damerau-Levenshtein algorithm made this method sensitive to low word length and word division width (bigram, trigram).
5) The effectiveness of both algorithms decreases as the width of the word division into n-grams increases. For words longer than about 10 characters, their division of words into bigrams and trigrams gives similar algorithm performance.
6) The computational complexity of the Damerau-Levenshtein algorithm is much higher than that of Sorensen-Dice. The average Top10Can list search time for the LD algorithm is over 200 seconds, while the SD algorithm reaches spells of 15 seconds.
7) Combining several methods slightly affected the total performance. On the other hand, it improved the quality of the received list by shifting the position of the correct word towards the top of the list.

REFERENCES

[1] E. Lukasik, M. Charytanowicz, M. Milosz, M. Tokovarov, M. Kaczorowska, D. Czerwinski, and T. Zientarski, "Recognition of handwritten Latin characters with diacritics using CNN," *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 69, Jan. 2021, Art. no. 136210.
[2] J. Tait and Y. Wilks, "Anniversary article: Then and now: 25 years of progress in natural language engineering," *Natural Lang. Eng.*, vol. 25, no. 3, pp. 405–418, May 2019.
[3] S. Kundu, H. S. Chhabra, S. S. Ara, and R. P. Mishra, "Optical character recognition using 26-point feature extraction and ANN," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 5, pp. 156–162, May 2017.
[4] J. Mahajan and R. Mahajan, "Designing an intelligent system for optical handwritten character recognition using ANN," *Int. J. Comput. Appl.*, vol. 91, no. 13, pp. 1–4, Apr. 2014.
[5] D. Singh, Mohd. A. Khan, A. Bansal, and N. Bansal, "An application of SVM in character recognition with chain code," in *Proc. Commun., Control Intell. Syst. (CCIS)*, Mathura, India, Nov. 2015, pp. 167–171.
[6] K. Verma and R. K. Sharma, "Comparison of HMM- and SVM-based stroke classifiers for gurmukhi script," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 51–63, Dec. 2017.
[7] E. A. Popko and I. A. Weinstein, "Fuzzy logic module of convolutional neural network for handwritten digits recognition," *J. Phys., Conf.*, vol. 738, Aug. 2016, Art. no. 012123.
[8] C. M. Bande, M. Klekovska, I. Nedelkovski, and D. Kaevski, "Recognition features for old slavic letters: Macedonian versus Bosnian alphabet," *Int. J. Sci. Res. Publications*, vol. 5, no. 12, pp. 145–153, 2015.
[9] A. Rabus, "Recognizing handwritten text in Slavic manuscripts: A neural-network approach using transkribus," *Scripta e-Scripta*, vol. 19, pp. 9–32, Jan. 2019.
[10] W. Kacalak and M. Majewski, "Handwriting recognition methods using artificial neural networks," *Proc. Artif. Neural Netw. Eng. (ANNIE)*, vol. 16, 2016, pp. 1–9.
[11] D. Grzelak, K. Podlaski, and G. Wiatrowski, "Analyze the effectiveness of an algorithm for identifying Polish characters in handwriting based on neural machine learning technologies," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 10, pp. 1258–1264, Dec. 2021.
[12] V. J. Hodge and J. Austin, "A comparison of standard spell checking algorithms and a novel binary neural approach," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 5, pp. 1073–1081, Sep. 2003.
[13] F. Ahmed, E. W. De Luca, and A. Nürnberger, "Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness," *Polibits*, vol. 40, pp. 39–48, Dec. 2009.
[14] A. C. Kinaci, "Spelling correction using recurrent neural networks and character level N-gram," in *Proc. Int. Conf. Artif. Intell. Data Process. (IDAP)*, Sep. 2018, pp. 1–4.
[15] R. Saluja, D. Adiga, G. Ramakrishnan, P. Chaudhuri, and M. Carman, "A framework for document specific error detection and corrections in indic OCR," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Kyoto, Japan, vol. 4, Nov. 2017, pp. 25–30.
[16] K. Ogada, W. Mwangi, and W. Cheruiyot, "N-gram based text categorization method for improved data mining," *J. Inf. Eng. Appl.*, vol. 5, no. 8, pp. 35–43, 2015.
[17] K. S. Kalaivani and S. Kuppuswami, "Exploring the use of syntactic dependency features for document-level sentiment classification," *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 67, pp. 339–347, Apr. 2019.

[18] Md. M. Hossain, Md. F. Labib, A. S. Rifat, A. K. Das, and M. Mukta, "Auto-correction of English to Bengali transliteration system using Levenshtein distance," in *Proc. 7th Int. Conf. Smart Comput. Commun. (ICSCC)*, Sarawak, Malaysia, Jun. 2019, pp. 1–5.
[19] P. Mandal and B. M. M. Hossain, "A systematic literature review on spell checkers for Bangla language," *Int. J. Modern Educ. Comput. Sci.*, vol. 9, no. 6, pp. 40–47, Jun. 2017.
[20] T. Mittra, S. Nowrin, L. Islam, and D. C. Roy, "A Bangla spell checking technique to facilitate error correction in text entry environment," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, Dhaka, Bangladesh, May 2019, pp. 1–6.
[21] B. C. Gencosman, H. C. Ozmutlu, and S. Ozmutlu, "Character n-gram application for automatic new topic identification," *Inf. Process. Manag.*, vol. 50, no. 6, pp. 821–856, Nov. 2014.
[22] G. Kaur, K. Kaur, and P. Singh, "Spell checker for Punjabi language using deep neural network," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Coimbatore, India, Mar. 2019, pp. 147–151.
[23] M. Attia, P. Pecina, Y. Samih, K. Shaalan, and J. Van Genabith, "Improved spelling error detection and correction for Arabic," in *Proc. COLING*, 2012, pp. 103–112.
[24] R. Sakuntharaj and S. Mahesan, "Use of a novel hash-table for speeding-up suggestions for misspelt Tamil words," in *Proc. IEEE Int. Conf. Ind. Inf. Syst. (ICIIS)*, Dec. 2017, pp. 1–5.
[25] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic n-grams as machine learning features for natural language processing," *Exp. Syst. Appl.*, vol. 41, no. 3, pp. 853–860, Feb. 2014.
[26] N. Gali, R. Mariescu-Istodor, D. Hostettler, and P. Fränti, "Framework for syntactic string similarity measures," *Exp. Syst. Appl.*, vol. 129, pp. 169–185, Sep. 2019.
[27] Y. Bassil and M. Alwani, "Context-sensitive spelling correction using Google web 1T 5-gram information," *Comput. Inf. Sci.*, vol. 5, no. 3, pp. 37–48, Apr. 2012.
[28] Y. Bassil and M. Alwani, "OCR context-sensitive error correction based on Google web 1T 5-gram data set," *Amer. J. Sci. Res.*, vol. 50, Feb. 2012, doi: 10.48550/arXiv.1204.0188.
[29] S. Sharma and S. Gupta, "A correction model for real-word errors," *Proc. Comput. Sci.*, vol. 70, pp. 99–106, Jan. 2015.
[30] L. Dinges, A. Al-Hamadi, M. Elzobi, and A. Nürnberger, "Automatic recognition of common Arabic handwritten words based on OCR and N-GRAMS," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3625–3629.
[31] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2016, pp. 2793–2799.
[32] O. O. Oladayo, "Optical character recognition of off-line typed and handwritten English text using morphological and template matching techniques," *IAES Int. J. Artif. Intell. (IJ-AI)*, vol. 3, no. 3, p. 121, Sep. 2014.
[33] B. Ziolko and D. Skurzok, "N-grams model for Polish," in *Speech and Language Technologies*, vol. 107, Jun. 2011, pp. 107–126.
[34] B. Ziolko, D. Skurzok, and M. Michalska, "Polish n-grams and their correction process," in *Proc. 4th Int. Conf. Multimedia Ubiquitous Eng.*, Aug. 2010, pp. 1–5.
[35] G. A. Fink, "N-gram models," in *Markov Models for Pattern Recognition* (Advances in Computer Vision and Pattern Recognition). London, U.K.: Springer, 2014, pp. 107–127, doi: 10.1007/978-1-4471-6308-4_6.
[36] F. Wang, T.-T. Quach, J. Wheeler, J. B. Aimone, and C. D. James, "Sparse coding for n-gram feature extraction and training for file fragment classification," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2553–2562, Oct. 2018.
[37] B. Ilijoski and Z. Popeska, "N-gram measure for syntactical similarity of the words," in *Proc. 10th ICT Innovations Conf., Eng. Life Sci. Web*, 2018, pp. 37–45.
[38] G. V. Bard, "Spelling-error tolerant, order-independent pass-phrases via the Damerau–Levenshtein string-edit distance metric," in *Proc. 5th Australasian Symp. ACSW Frontiers*, vol. 68, 2007, pp. 117–124.
[39] A. Al-Bakry and M. Al-Rikaby, "Enhanced Levenshtein edit distance method functioning as a string-to-string similarity measure," *Iraqi J. Comput. Informat.*, vol. 42, no. 1, pp. 48–54, Dec. 2016.

PIOTR WOJCICKI received the M.Eng. degree in mechatronics from Lublin University of Technology, Lublin, Poland, in 2014. From 2014 to 2016, he was an Engineer with the Division of Silicon Microsystem and Nanostructure Technology, Institute of Electron Technology, Piaseczno. Since 2016, he has been a Research Assistant with the Department of Computer Science, Lublin University of Technology. His main research interests include pattern recognition, machine learning, voice synthesis, deepfake recognition, robotics, the IoT, and applied computer science.

TOMASZ ZIENTARSKI received the M.S. and Ph.D. degrees from the Faculty of Chemistry, Maria Curie-Skłodowska University, Lublin, Poland, in 1994 and 2000, respectively. Since 2015, he has been an Associate Professor with Lublin University of Technology, where he is currently the Head of the Department of Computer Science and Computer Modelling. His main scientific research interests include computer simulation and modeling of physicochemical processes, microelectronics, the Internet of Things, natural language processing, wireless transmission, and neural networks.