The words separation in old Cyrillic texts with fuzzy search method

M N Mokrousov[1]

[1]Kalashnikov Izhevsk State Technical University, Studencheskaya str., 7, Izhevsk, Russia, 426069

e-mail: maxmok@mail.ru

Abstract. The article describes a solution to the problem of word separation in old Cyrillic texts after the stage of graphic character recognition on scanned documents. The article proposes an algorithm for fuzzy text search using the grammatical dictionary of the Old Russian language, with the completeness and accuracy of search results evaluate. To assess the relevance and ranking of the search results, a method for calculating the rank of the symbol recognition variant based on the TF-IDF metric is developed. The article also presents a software system of automated word search, presents the results of experiments that prove the effectiveness of the developed algorithms and programs.

1. Introduction

The urgency of the task of translating ancient Cyrillic manuscripts from graphic representation into text form is due to their exceptional value for historical and linguistic research, which is most effective using the methods of automatic text analysis and recognition of electronic-graphic representation of manuscripts. For maximum process digitization automation of ancient texts requires the involvement pattern recognition methods. A relatively small publications count on recognition systems of handwritten and old-printed Cyrillic texts of the X–XVIII centuries suggests the need to improve methods and technologies to solve this problem.

Almost all researches on the problem of historical documents recognition, says that the existing commercial OCR-system (Optical Character Recognition) do not cope well with the recognition of ancient texts. Researchers from Macedonia [1-2] proposed two ways of recognizing old-printed Cyrillic symbols: on the basis of decision trees and by fuzzy classification. Both methods operate with statistical and simple structural features of symbols. The experimental system recognizes symbols in each method with an average accuracy and completeness of 70-80%. The reducing issue of the recognition result based on information about neighboring characters (the use of a dictionary, etc.) in the work is not affected. In [3] the optical recognition of historical texts is made by whole words, not by individual symbols. At the same time, the recognition process is adaptive; during the work it adjusts itself to the processed historical document. The book of the XVIII century, written in the old German Gothic script, the developed software system correctly recognized 86.6%, which is 4.1% higher than the traditional OCR-system. In [4] the search system in text images produces a pre-selection of words in the archival texts, using such parameters as the minimum possible row height,

margins, relative threshold of the brightness of the background in a separate line, etc. Search in the text based on comparison of the sample image with all images words or symbols that are pre-allocated by the system. The estimation of word images similarity is based on the calculation of the modified Hausdorff distance [5]. It is indicated that the search process takes a long time due to the high computational complexity of the comparison based on the Hausdorff distance.

In [6] describes the automatic text recognition capabilities of neural network models specifically trained to recognize different styles of Church Slavonic handwriting within the software platform Transkribus. Computed character error rates of the models are in the range of 3 to 4 percent; real-life performance shows that specifically trained models, basically, recognize simple (non-superscript) characters correctly most of the time. Error rate is higher with superscript letters, abbreviations, and word separation.

In our opinion, the approaches of word recognition are poorly applicable to Cyrillic texts, because in these texts words are not separated from each other, and there is a relatively large distance between the letters. The individual words selection in such texts is not an easy task. To automatically solve this problem, it is necessary to recognize individual characters, conduct lexical and syntactic analysis of the document.

This paper describes the stage of "Clarify with a dictionary" after graphic recognition of individual characters stage, described in [7]. This step involves the use of fuzzy algorithm to search the dictionary of different combinations of characters variants and the subsequent automated words and symbols selection.

One of the most suitable search methods in this case is the pattern search algorithms, which belong to the group of search methods for substring in a string. For example [8-9] describes and includes implementations the most famous of them: Knuth–Morris–Pratt (KMP) algorithm, Boyer–Moore string-search algorithm, Rabin–Karp algorithm and etc. The use of regular expressions allows expanding the capabilities of the standard text search. In [10] presents two new techniques for regular expression searching, which permit fast searching for regular expressions than any existing algorithm. In this paper the main focus is on the speed of search but not quality.

Almost all fuzzy search algorithms have software implementations in different programming languages. For example, this site [11] provides fuzzy string search tools in text and dictionary written in Java. It contains the most commonly used algorithms and auxiliary utilities for fuzzy string search in large dictionaries. The site includes implementations for Levenshtein Distance, Damerau-Levenshtein Distance, Extension (Spell-checker) Method, N-Gram Method, Signature Hash Method, Bitap, Burkhard-Keller (BK) Trees, Skip algorithm.

However, full borrowing of software solutions to solve the problem is difficult for several reasons, which were identified by the author personally as the analysis result scanned manuscripts on the project *Manuscripts.ru* [21].

First, old-printed Cyrillic texts do not have delimiters (punctuation marks), which complicates the process of singling out individual words in the source text. The maximum length of a word in the dictionary is 26 characters, the minimum is 1 character, which can mean a number. The number of variants for each symbol after graphic recognition can be in the range from 1 to 6. Taking into account the maximum length of the word, the number of possible characters chains, taking into account the maximum length of 26 characters, is determined by the combinatory product rule of probability theory. In this case, the maximum number of search queries for one potentially long word only will be 6^{26}, which is unnecessarily much.

Secondly, after graphic recognition process of low quality images, with various spots, scuffs, page breaks, followed by gluing, the consequences of printed documents restoration, the result of recognition may have the following errors:

1) one character is recognized as two or three;
2) two or three consecutive characters are recognized as one;
3) omission of characters when the recognition program could not pick up the options; most often this case occurs in the presence of a dark spot or scratches;
4) there is no correct variant in the sequence of character recognition probabilities;
5) the word is not in the dictionary.

At the moment, these "noise" problems can be solved only with the use of manual editing the graphic recognition results.

Thirdly In Slavonic-Russian manuscripts simple or alphabetic titles abbreviations are used and such abbreviations are difficult for analysis and search. Such abbreviations were most often used to refer to words related to God and the Church (God, Amen, Trinity, etc.), as well as to other words (maiden, wisdom, memory, etc.).

Fourth, it is necessary to take into account the morphological rules of character conversion and replacement. These are different rules for equating individual letters and their variants or letter combinations at the end of the word form, after certain vowels, consonants or taking into account the position of the characters in the text.

Fifth, a separate item that makes it difficult to fully automate the character recognition process is the presence of a drop cap (initial) – the first capital letter of the text or Chapter, depicted as the miniature.

Author decided to use a recursive search method based on regular expressions, which would take into account the symbol position in the word when evaluating the search relevance. Such fuzzy *positional* search will allow to abstract from possible graphic recognition *errors* and give more freedom to the expert who manages the search.

2. Fuzzy "positional" dictionary search algorithm

To reduce the number of symbols variants obtained after the graphic recognition stage, it is necessary to search for words in the grammatical dictionary of the Old Russian language created within the framework of the project *Manuscripts.ru* [12]. For this purpose, a recursive search algorithm based on regular expressions is used.

2.1. Pre-processing of the input text

The input text is a sequence of characters with variants after graphical recognition. Each character is represented by the following structure:

<Symbol>::=‘[’[<Coordinates>’,’{<Value Variant>’;’}]’]’

<Coordinates>::=<X>’,’<Y>’,’<Width>’,’<Height> *<Variant Value>::=<Value>’,’<Relevance>[’,’<Number_diactric>],*

where *<Coordinates>* are the coordinates of the rectangular area in which the symbol is enclosed in the original image; *<X>,<Y>* are the upper left corner coordinates of the symbol rectangular area in the image; *<Width>,<Height>* are the width and height of the rectangular area of the symbol in the image; *<variant Value>* is the variant of the symbol value; *<Value>* is the percentage of the variant relevance after the graphic recognition stage; *<Number_diactrika>* is the number of previous characters, which is covered by the diactric symbol.

Example:

[(422,1281,64,71),в,89;б,92;г,98;н,81;п,80][(506,1280,50,70),л,97;а,94;в,83;,82]

[(473,1316,72,18)◌̀,100,2][(583,1268,17,18),,78][(0,0,0,0),]

As you can see, the text may contain diactric symbol:

[(473,1316,72,18)◌̀,100,2]

In this case, after the relevancy percentage indicates how much the previous characters covered by this accented character. Also, there may be omissions of characters that could not be determined after graphic recognition stage.

The source text is converted to a table form, where each table column contains variants of the symbol value. The columns number is equal to the characters number.

2.2. Generation of regular expressions and a recursive search in the dictionary

The next is the regular expressions generation and recursive search in the dictionary. This step analyzes variant characters combinations from 1 to 10 (configured in the search options). Each combination is a regular expression that is inserted into the SQL-query:

*SELECT * FROM Dictionary WHERE word like N'[нн][сс][к][сс][ннп]'*

The result of the query is a words set found by this expression. If the words count is greater than the threshold value (the default is 100), this result is not considered.

The input to the search algorithm

$Table is an characters variants array after the graphic recognition that are received from the original text;

|$Table| is the number of characters in the input text.

$maxWordLength is the maximum length of the considered symbols combinations.

$maxWordCount is the maximum words count in the query result.

The output

$Result is the resulting array with words sets for each character.

The main dictionary search cycle

```
$b ← 0
$e ← 0
Repeat
{
        RecognizeNextWord( $b, output $e);
        $b ← $e;
} While ($b < |$Table|);
```

Recursive search function

```
RecognizeNextWord( $b, output $e)
{
        $d ← $e – $b + 1;
        While (($d <= $maxWordLength) AND ($e < |$Table|))
        {
                $reg ← GenerateRegularExpression($b, $e);
                $symbol ← $Table[$b];
                if (IsQueryResultAlreadyExists($reg, $symbol) = FALSE)
                {
                        $res ← ExecuteQuery($reg, $symbol);
                        SaveWordSet($res)
                        $Result ← $Result ∪ $res;
                        if (|$res| > 0) AND (|$res| <= $MaxWordCount))
                        {
                                $e2 ← $e +1;
                                RecognizeNextWord($e + 1, output $e2);
                        }
                }
                $e++;
                $d ← $e - $b + 1;
        }
}
```

where *$b* is the character index from which the scan starts;

$e is the character index that ends the scan;

$d is the current length of the scanned character combination;

$reg is the regular expression for the current character combination;

$symbol is the first character of the current combination;

$res is the words set of the current symbols combinations;

GenerateRegularExpression($b, $e) is a function that returns a regular expression for the current character combination;

IsQueryResultAlreadyExists ($reg, $symbol) is a function that returns *True* if for the regular expression *$reg* and the symbol *$symbol* already exists a words set, and *False* otherwise;

ExecuteQuery($reg, $symbol) is a function that searches words in the dictionary by the regular expression *$reg* and the first character *$symbol* and returns a words set;

SaveWordSet($res) is a function that saves a words set *$res* in working memory.

After executing the main algorithm, an array with sets of words for each character combination not exceeding the set threshold will be formed.

Thus, the fuzzy search algorithm consists of two main steps:

1) the original text transformation into table form;

2) the recursive search in the dictionary for character variants combinations using regular expressions.

3. Reducing the number of character recognition options based on the fuzzy search algorithm

The variants number reducing method after graphic character recognition based on the fuzzy search algorithm results is to assess the relevance of the symbols variants by calculating the weights of each symbol variant on such data as:

- frequency of the symbol occurrence in the fuzzy dictionary search results;
- frequency of the symbol occurrence in the fuzzy dictionary search results, taking into account the position in the word and the length of the word;
- frequency of the symbol occurrence in the dictionary;
- frequency of the symbol occurrence in the dictionary, taking into account the position in the word and the length of the word.

The metric TF-IDF [4] adapted for this task is used to calculate the weight of the variant. It was proposed to use a symbol variant (hereinafter the symbol) as a word in this metric, and sets of words as documents. Thus, SF (symbol frequency) – the symbol importance, in the context of the word sets in which the symbol participated, is estimated by the formula:

$$SF(s,W) = \frac{\left|(s \in w_i)\right|}{\left|W\right|},$$

where $|s \in w_i|$ is the occurrences count of a character s in a set of words derived from regular expressions involving this character; $|W|$ is the total words count in the sets obtained from regular expressions involving this character. IDF (*inverse dictionary frequency – inverse frequency of the dictionary*) is the inversion of the frequency which some character s occurs in the dictionary D. There is only one *IDF* value for each character within the dictionary:

$$IDF(s,D) = \log\frac{\left|D\right|}{\left|(s \in d_i)\right|},$$

where $|D|$ is the words count in the dictionary (1846721); $|s \in d_i|$ is the dictionary words count in which the symbol s occurs.

In order to take into account the position p of the symbol s and the length of the words l in the sets W, in which the symbol in question is present, the calculation formulas *SF* and *IDF* can be written as follows:

$$SF(s,W,l,p) = \frac{\left|(w_i(l,p,s) \in W)\right|}{\left|(w_i(p,s) \in W)\right|},$$

where $w_i(l,p,s)$ is the *i-th* word in sets W of length l at position p of which stands the symbol s; $w_i(l,p,s) \in W$ is the words count $w_i(l,p,s)$ in the sets W; $w_i(p,s)$ is the *i-th* word in sets W at position p of which stands the symbol s; $w_i(p,s) \in W$ is the words count $w_i(p,s)$ in the sets W.

$$IDF(s,D,l,p) = \log\frac{\left|(d_i(s,p) \in D)\right|}{\left|(d_i(s,p,l) \in D)\right|},$$

where $d_i(s,p)$ the *i-th* word of the dictionary *D* at position *p* of which stands the symbol *s*; $|d_i(s,p) \in D|$ is the words count $d_i(s,p)$ in the dictionary *D*; $d_i(s,p,l)$ is the *i-th* word of the dictionary D of length *l* at position *p* of which stands the symbol *s*; $|d_i(s,p,l) \in D|$ is the words count $d_i(s,p,l)$ in the dictionary *D*.

Thus, the weight (relevance) of a symbol is estimated in two ways:

1) based on the character occurrences count in word sets derived from regular expressions involving that character:

$$Weight(s) = SF(s,W) * IDF(s,D)$$

2) taking into account the symbol position and the words length in the sets in which the analyzed symbol is present:

$$Weight(s,l,p) = IDF(s,D,l,p) * SF(s,W,l,p)$$

So, in word sets *W*, the symbol *s* can occur at different positions *p* in words of different lengths *l*, the total weight of the symbol, taking into account these parameters, is calculated as the sum of the weights:

$$WeightSum(s,l,p) = \sum_{i=1}^{n} Weight_i(s,l,p),$$

where *n* is the number of weights of the symbol *s*, taking into account the different positions and the word length in the word sets in which the symbol participates.

The total symbol variant weight is calculated as the sum of the weights, taking into account the symbol occurrences count in the word sets and taking into account the position and length of words:

$$WeightTotal(s) = Weight(s) + WeightSum(s,l,p)$$

The next, the $Rank(s_i)$ of each s_i variant of the symbol *s* is calculated as the ordinal number in the variant weights array sorted in descending order. For each symbol variant, there are tree ranks calculated for each weight: $Rank(Weight(s_i))$, $Rank(WeightSum(s_i,l,p))$, $Rank(WeightTotal(s_i))$. Total rank *RankTotal* for the symbol variant is computed as the harmonic mean of the character variant ranks. Each character variant is ranked by accounting *RankTotal*. As the rank was calculated by the variant symbol position in the sorted weight arrays, the lower the rank, the higher the relevance.

4. Experiment description

The graphic recognition variants reduction of the text "Ostromir gospel" was carried out by the following method:

1) selected fragment of the text;
2) a fuzzy dictionary search was performed for the selected text fragment;
3) calculated weight and the symbols variants ranking on the basis of search results;
4) correct the characters variants were set by manual word confirmation found from the dictionary search results;
5) Precision, Recall and completeness were calculated and recorded in the table.

Figure 1. The "Ostromir gospel" fragment with graphic recognized symbols.

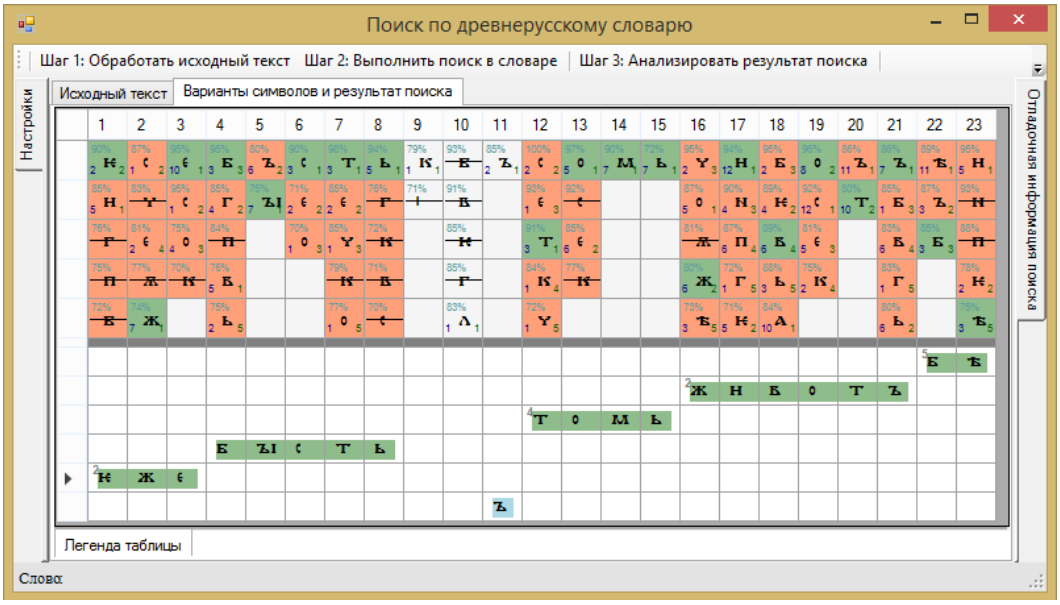

Figure 2. Analysis results of the text 9-10 lines "Ostromir gospel".

The words searching result in the dictionary after graphic recognized text (9-10 lines of "Ostromir gospel" [14], Figure 1), followed by confirmation of the words is shown in Figure 2.

The symbol variant rank is indicated in the lower right corner of the cell with the symbol variant, the variant occurrence frequency in the search results is shown in the lower left corner, and the graphic recognition relevance percentage is shown in the upper left corner. If the symbol variant does not occur in the found word sets, it is crossed out (frequency of occurrence – 0).

In the lower part of the table the words confirmed by the expert are highlighted in green. The number in the upper left corner of the word shows how many words were found in the dictionary by the combination of options (the words count in the set).

Figure 3 shows the process of assembling a sentence from words obtained by searching the dictionary and then confirming the correct word from the search results set. Each symbol and each word in the process of the software system operation can be viewed on the original image. During the sentence assemble the expert can manually add the missing characters and words.

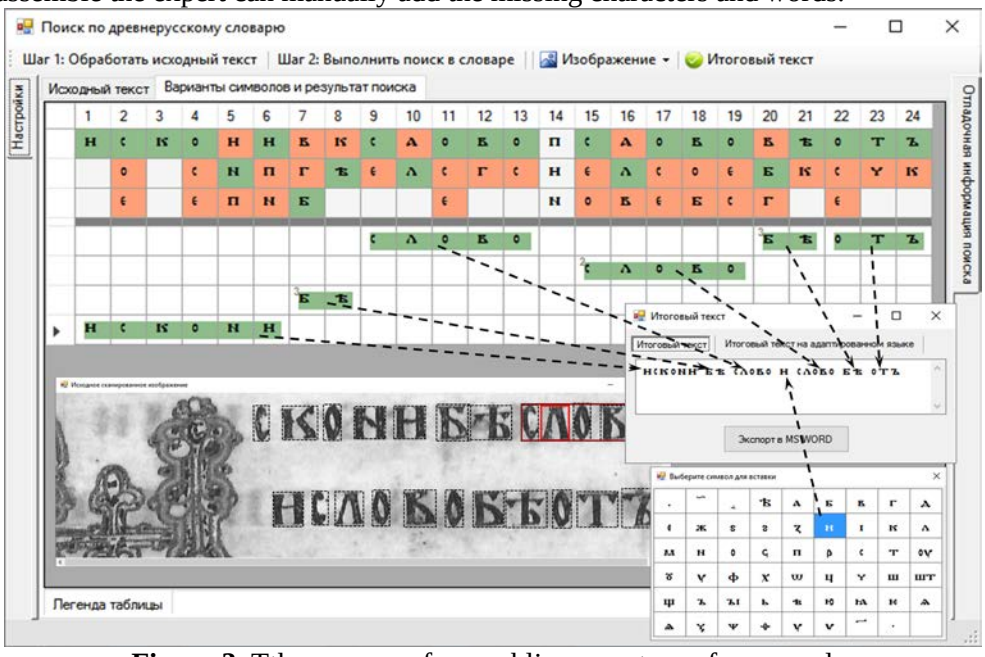

Figure 3. Tthe process of assembling a sentence from words.

The Precision and Recall of the reduction results were evaluated for the highest ranked options (1 and 2).

Let ***S*** be a set of options of characters from the input text; $S^+ \subset S$ is a subset of the symbols that have the highest rank; $S^+ \subset S$ is a subset of the symbols that were used in the formation of words sets (not the strikethrough options); $S^E \subset S$ is a subset of the characters that have been confirmed by the expert (highlighted in green); S' is a lot of characters in the input text (number of columns). In addition to Precision and Recall, the percentage of Reduction the number of symbol variants that did not participate in search queries was estimated.

$$Precision = \frac{\left|S^+ \cap S^E\right|}{\left|S'\right|},\ Recall = \frac{\left|S^+ \cap S^E\right|}{\left|S^E\right|},\ Reduction = \frac{\left|S\right| - \left|S^\Sigma\right|}{\left|S\right|}$$

Table 1. Precision and Recall of the results of reducing the variants character count.

Experiment №	Precision of up to 1 variant	Recall of up to 1 variant	Precision of up to 2 variants	Recall of up to 2 variants	Reduction
1	0.67	0.67	0.9	0.9	0.15
2	0.57	0.67	0.86	1	0.18
3	0.52	0.79	0.62	0.93	0.27
4	0.52	0.52	0.83	0.83	0.17
5	0.52	0.6	0.7	0.8	0.26
6	0.77	0.77	0.86	0.86	0.31
7	0.55	0.55	0.83	0.83	0.26
8	0.36	0.45	0.64	0.8	0.14
9	0.24	0.67	0.32	0.89	0.44
10	0.5	0.59	0.62	0.73	0.14
Average	0.52	0.63	0.72	0.86	0.23

The calculated parameters of Precision and Recall were summarized in a table (Table 1). The Precision and Recall of up to one variant means that only one symbol variant, whose rank is 1 and which is marked as correct by the expert advisor, was taken into account in the calculation. The Precision and Recall of up to two variants means that the calculation took into account two symbol variants with the highest rank, among which there is a variant marked as correct by the expert advisor.

5. Conclusion

The experiments conducted to reduce the symbols variants count based on the fuzzy search algorithm show that the proposed variants reducing method to one variant gives an accuracy rate of 52%, and completeness – 63%, to two variants – 72% and 86%, respectively. Given that one of the symbol variants was graphically recognized correctly, the completeness of reducing variants method in combination with graphic recognition results tends to 100%.

In the future, it is planned to conduct experiments with the use of N-grams, built according to the Old Russian dictionary and the results of expert search using the algorithm described in this article. Author hopes to obtain statistical data about frequency of the letter combinations in old Cyrillic texts, to obtain regularities of language models for old Cyrillic texts, which will improve the efficiency of words search and words selection in such texts.

6. References

[1] Klekovska M, Martinovska C, Nedelkovski I and Kaevski D 2012 Comparison of Models for Recognition of Old Slavic Letters *ICT Innovations* 129-139

[2] Bande C M, Klekovska M, Nedelkovski I and Kaevski D 2014 Feature Selection for Classification of Old Slavic Letters *Control Engineering and Applied Informatics* 16(4) 81-90

[3] Kluzner V, Tzadok A, Shimony Y and Walach E 2009 Antonacopoulos. Word-Based Adaptive OCR for Historical Books *Proceedings of the 10th International Conference on Document Analysis and Recognition* 501-505
[4] Kirov N 2008 A software tool for searching in binary text images *Pregled NCD* **13** 9-16
[5] Dubuisson M-P, Jain A 1994 A Modified Hausdorff Distance for Object Matching *Proc. 12th Int. Conf. Pattern Recognition* 566-568
[6] Rabus A *Recognizing handwritten text in Slavic manuscripts: A neural-network approach using Transkribus* URL : https://www.academia.edu/38835297
[7] Kuchuganov A V and Kasimov D R 2016 Automation of recognition of old-printed characters with the help of descriptive logic *Rašytinis palikimas ir skaitmeninės technologijos: VI tarptautinė mokslinė konferencija* (Vilnius; Iževskas) 104-109
[8] Borivoj M, Holub J and Polcar J 2005 Text Searching Algorithms. Volume I: Forward String Matching **1(2)** URL: http://stringology.org/athens/TextSearchingAlgorithms
[9] Introduction to Pattern Searching Algorithms *Tutorialspoint. The Biggest Online Tutorials Library* URL : https://www.tutorialspoint.com/introduction-to-pattern-searching-algorithms
[10] Navarro G, Raffinot M 2005 New Techniques for Regular Expression Searching *Algorithmica* **41(2)** 89-116 DOI: 10.1007/s00453-004-1120-3
[11] Fuzzy search tools *Google code archive* URL: https://code.google.com/archive/p/fuzzy-search-tools/
[12] The Portal "Manuscript" URL: http://manuscripts.ru/
[13] Jurafsky D and Martin J 2000 *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (New Jersey: Prentice Hall) p 934
[14] Ostromir gospel. Manuscript sheets *Russian national library* URL: http:// www.nlr.ru/exib/Gospel/ostr/ill.html

Acknowledgments
The author expresses gratitude to the Department of the information processing system staff from Kalashnikov Izhevsk State Technical University: Kasimov Denis Rashidovich, Kuchuganov Alexander Valerievich, Kuchuganov Valeriy Nikonorovich for valuable advice and recommendations, and preparation of experimental data.