DEVELOPMENT OF A STEMMING ALGORITHM BASED ON A LINGUISTIC APPROACH FOR WORDS OF THE UZBEK LANGUAGE

Bakayev Ilhom Izatovich
Tashkent University of Information Technologies, 108, Amir Temur ave., Tashkent, 100200, Uzbekistan
bakayev2101@gmail.com

Abstract: The automatic processing of unstructured texts in natural languages is one of the relevant problems of computer analysis and text synthesis. Within this problem, the author singles out a task of text normalization, which usually suggests such processes as tokenization, stemming, and lemmatization. The existing stemming algorithms for the most part are oriented towards the synthetic languages with inflectional morphemes. The Uzbek language represents an example of agglutinative language, characterized by polysemanticity of affixal and auxiliary morphemes. Although the Uzbek language largely differs from, for example, English language, it is successfully processed by stemming algorithms. There are virtually no examples of effective implementation of stemming algorithms for the Uzbek language; therefore, this questions are the subject of scientific interest and defines the goal of this work. In the course of this research, the author solved the task of bringing the given texts in the Uzbek language to normal form, which on the preliminary stage were tokenized and cleared of stop words. To author developed the method of normalization of texts in the Uzbek language based on the stemming algorithm. The development of stemming algorithm employed hybrid approach with application of algorithmic method, lexicon of linguistic rules and database of the normal word forms of the Uzbek language. The precision of the proposed algorithm depends on the precision of tokenization algorithm. At the same time, the article did not explore the question of finding the roots of paired words separated by spaces, as this task is solved at the stage of tokenization. The algorithm can be integrated into various automated systems for machine translation, information extraction, data retrieval, etc.

Introduction

Along with the development of information technologies, the ways of data presentation are evolving just as actively. However, unstructured natural language texts are still the most common data type and, at the same time, one of the most difficult to process automatically. Currently, there are more than 7000 living languages in the world [1], of which a little more than 100 have a written language and only 40 are considered the most widespread - which is spoken by about two-thirds of the world's population. As diverse as natural languages are, their semantic and grammatical properties are just as varied. For this reason, already existing text processing algorithms that are very effective for one language are often completely inapplicable for texts in another language. Moreover, we are talking not only about algorithms for semantic analysis, which is considered an AI-complete task [2], but also about algorithms for solving narrower problems of text normalization. As you know, one and the same, in fact, a search query in a natural language can be composed in many variants. Therefore, an effective search engine must be able to process them all and identify them with each other in order to ultimately return a result relevant to the query.

In this regard, the reduction of the text to the normal (canonical) form significantly simplifies its further processing in any automated systems: search engines, machine translators, auto-abstraction systems, etc. Normalization of the text not only significantly improves the efficiency of search engines, but also has relatively low requirements for time and computational costs. Not to mention the fact that normalized texts make it much easier to solve the problems of their classification and clustering. Text normalization usually involves performing a number of basic processes, including tokenization, stemming, and / or lemmatization. Tokenization is the process of breaking up text down to the smallest atomic units (tokens). Stemming is the process of determining the unchanging stem of a given word (stemma), which does not necessarily coincide with its morphological root. Lemmatization is the process of reducing a given word to a lemma, that is, to its normal (dictionary) form. Stemming and lemmatization pursue the same goal - reducing inflectional forms of words to their normal form. The difference is that stemming algorithms operate without understanding the context and the difference between words, while lemmatization algorithms are based on the use of dictionaries and morphological analysis. Although lemmatization is a subtler and accurate process, and, accordingly, more resource-intensive, nevertheless, stemming algorithms have their own advantages: speed of operation and ease of implementation. In addition, in many cases, low accuracy in finding stemmas may not be critically important.

Among the most popular implementations of stemming algorithms developed to date are the stemmers by D. B. Lovins, M. Porter, K. Pace, and G. Husk (Lancaster's algorithm). Most algorithmic stemmers are, to one degree or another, derived from them or their modifications. In addition, there are stemming algorithms based on statistical, stochastic and hybrid approaches, for example, Stemka, N-gram stemmers, Brute force stemmers, etc. Existing stemming algorithms are overwhelmingly focused on synthetic languages, that is, those in which shaping using morphemes prevails. The authors [3] compare the stemming algorithms of Porter and Lancaster. The evaluation was performed based on the errors of stemming results and visualization of text data. As initial data, the authors used 10 text documents, which were selected at random from Internet newspapers. The results obtained by the authors show that Porter's algorithm is 43% more efficient than Lancaster's. The work [4] describes an algorithm for normalizing words based on the classification of endings and suffixes. The total number of which is 26526 suffixes and 3565 endings. For all parts of speech, a finite state machine is built taking into account the morphological properties of the Kazakh language. Also, the authors have developed a stemming algorithm without a dictionary based on the classification of suffixes and endings. The algorithm has been tested on Kazakh text corpora. The authors of [5] proposed an algorithm based on the morphological rules of the Persian language, which works according to the bottom-up principle. The algorithm consists of three stages: subline tags, rule matching, anti-rule matching. A total of 252 rules and 20 anti-rules are set. The first step is to extract morphological properties for all possible substrings. Morphemes and their clusters, as well as word stems (kernels) are determined. The rule matching step extracts the corresponding rules for each core. Kernels that do not match the rules go to the next anti-rule matching step. The algorithm was tested using the "Hamshahri" text corpus and showed correct extraction of 90.1% of word stems.

In the article [6], a new method was developed for obtaining the basis of words for different languages. Although the experiment was carried out by the authors only for English and Persian. It is argued that the method does not depend on the morphological rules of the language. The method uses a bilingual dictionary to define the stems of words. The proposed method is divided into several stages. At the first stage, the clustering of structural and semantic similarities of words from the dictionary is carried out. Next, a candidate word is selected from each cluster. These candidates are used to identify new words. The experiments carried out show that for the English language the method works with an accuracy of 69.52%, and for the Persian language - 70.32%. The main disadvantage of the stemming algorithms discussed above is that stemmas found with their help do not always correspond to the morphological root of words. Typically, such problems arise in cases where the type of a word is not known in advance, or the word form is formed in accordance with several rules. Thus, a situation occurs, the highlighted stems of words by themselves do not make sense. The Uzbek language is an agglutinative language, distinguished by the polysemanticity of affix and service morphemes. Despite the fact that the Uzbek language has many differences from the English language, which is also considered agglutinative, however, it fully allows the use of text normalization techniques based on stemming algorithms. The purpose of this work is to develop a method for normalizing texts in the Uzbek language based on a stemming algorithm for processing words, the structural type of which is known in advance.

Formulation of the problem.

The morphological composition of words in the Uzbek language includes two main parts: root and affixal morphemes. The root (o‘zak) is a morpheme that has a lexical meaning and does not have affixes in its composition. An affix is a morpheme that can have several functions. By joining the root, the affix forms a new word or grammatical meaning. The normal form (stem) of words in the Uzbek language (negiz) can consist only of the root morpheme or of the root and affixal morphemes at the same time. The stem of words can be derivative and non-derivative. A derivative stem consists of a root and a word-forming affix. For example, gul (flower) is a non-derivative stem, and guldon (vase) is a derivative stem. Word formation can occur in various ways, including by joining an affix to the root "gul + chi" (florist), by concatenating two simple words through a hyphen "baxt-saodat" (happiness), "tez-tez" (often). Thus, in the modern Uzbek language, words in their structure are: simple, complex, paired, repetitive (Fig. 1).

Simple	Complex	Paired	Repetitive
•kitob •daftar •guldon	•sotib olmoq •hurmat qildi •2021-yil	•baxt-saodat •asta-sekin •sixat-salomatlik	•tez-tez •katta-katta •yor-yor

Note that the corresponding software tool for determining the types of words of the Uzbek language, based on the use of regular expressions, was created by us earlier [7, 8]. The task is to bring to normal form all words of the given text, which after the preliminary stage of tokenization was marked by word types and subjected to cleaning from stop words.

Methods

The developed algorithm varies depending on the types of words, which correspond to the following designations: w_s - simple words (with a derivative or non-derivative stem); w_c - compound words; w_p - paired words; w_r - repeating words.

Of greatest interest is the problem of normalizing words like w_c and w_p, for example, "2021-yillar" and "sixat-salomatlik", respectively. The essence of the algorithm is that the affixes that occur after the hyphen are removed from the word form. If the original word has no affixes, then it is considered a stemma. That is, “2021-yillar” is truncated to “2021-yil” (2021), “sixat-salomatlik” is truncated to “sixat-salomat” (live) (Fig. 2).

Next, we will describe the variables and functions of the "StemWCWP" algorithm for the w_c and w_p types:

- "S" - string; - "Spart" - a string containing a part of the string S; - "b" - logical label; - "i" - Iterative variable; - "HyphenAfterStr (S)" - user-defined function that cuts out the part of the string after the hyphen; - "Length (string S)" - standard function for returning the length of a string; - "dbStemShow (string S)" - custom function that searches the database; - "SubStr (firstindex: int, lastindex: int)" is a standard function that extracts characters from firstindex to lastindex;

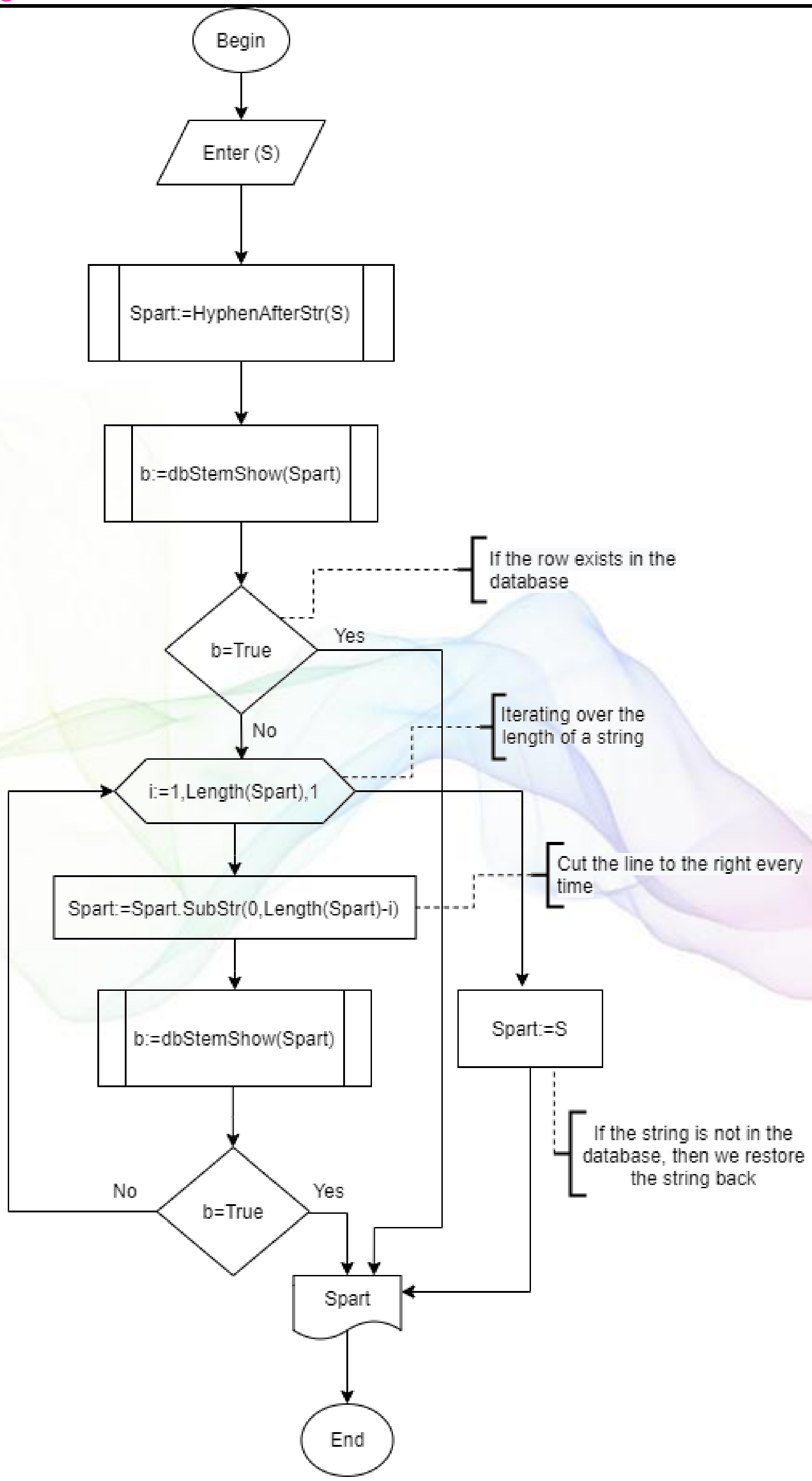
Begin
Enter (S)
Spart:=HyphenAfterStr(S)
b:=dbStemShow(Spart)
If the row exists in the database
b=True
Yes
No
Iterating over the length of a string
i:=1,Length(Spart),1
Cut the line to the right every time
Spart:=Spart.SubStr(0,Length(Spart)-i)
b:=dbStemShow(Spart)
Spart:=S
If the string is not in the database, then we restore the string back
No
b=True
Yes
Spart
End

Fig. 2. StemWCWP Algorithm Block Diagram

Words like w_p - "a sequence of alphabet characters + a hyphen + a sequence of alphabet characters", where words separated by a hyphen can be identical, or the first word can be a substring of the second word. In the second case, any affixes are cut off from the second word (Fig. 3).

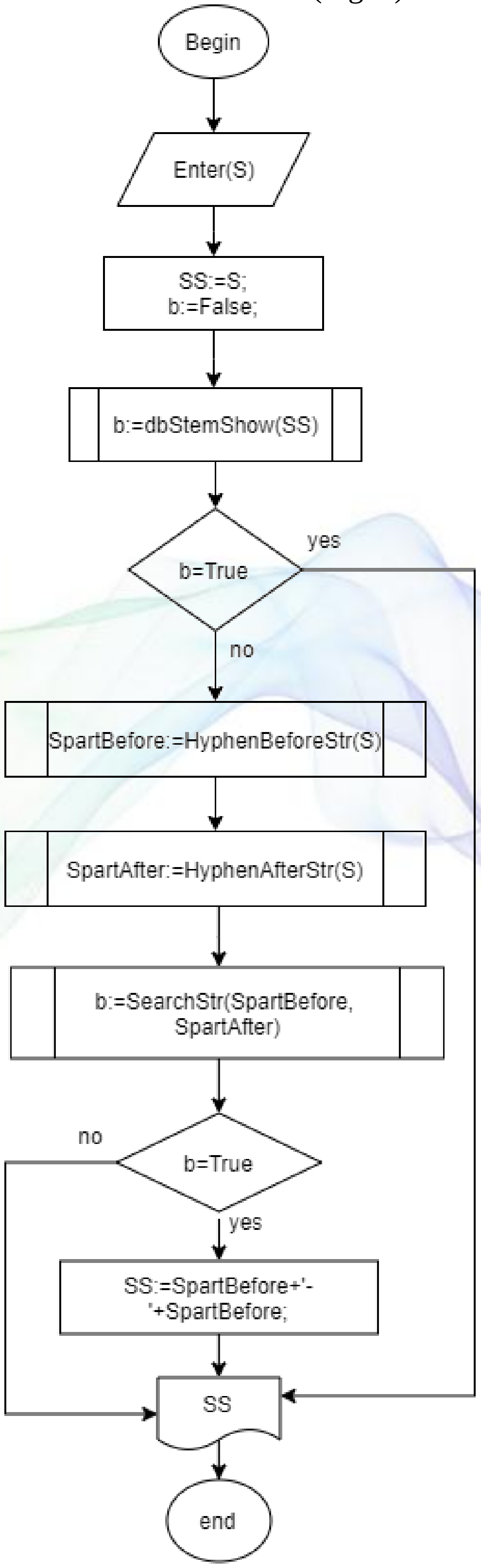
Begin
Enter(S)
SS:=S;
b:=False;
b:=dbStemShow(SS)
b=True
yes
no
SpartBefore:=HyphenBeforeStr(S)
SpartAfter:=HyphenAfterStr(S)
b:=SearchStr(SpartBefore, SpartAfter)
no
b=True
yes
SS:=SpartBefore+'-'+SpartBefore;
SS
end

Fig. 3. StemWP Algorithm Block Diagram

Description of variables and functions of the StemWP algorithm: - "S" - string; - "SS" - a string containing a duplicate of string S; - "b" - logical label; - "SpartBefore" - contains the result of the HyphenBeforeStr function (string S); - "SpartAfter" - contains the result of the "HyphenAfterStr (string S)" function; - "HyphenBeforeStr (string S)" - user-defined function that cuts out a part of the string before the hyphen sign; - "HyphenAfterStr (string S)" - user-defined function that cuts out the part of the string after the hyphen sign; - "SearchStr (string str1, string str2)" - a user-defined function that checks whether "str2" is part of "str1", if so, it returns true, otherwise - false. - "dbStemShow (string S)" - custom function that searches the database; Words like ws are composed of "a sequence of alphabet characters". Sometimes these words contain an apostrophe. For example, "a'lo", "kitob". To find the basis for this type of words, the prefix is removed, not the derived affixes (Figure 4).

Next, let's describe the variables and functions of the StemWS algorithm - "S" - string; - "SS" - a string containing a duplicate of string S; - "b" - logical label; - "i" - iteration variable; - "pref" - string for prefixes; - "Length (string S)" - a standard function that returns the length of a string; - "dbStemShow (string S)" - custom function that searches the database; - "preffunc (string S)" - user-defined function that extracts prefixes from a string variable; - "SubStr (firstindex: int, lastindex: int)" is a standard function that extracts characters from firstindex to lastindex;

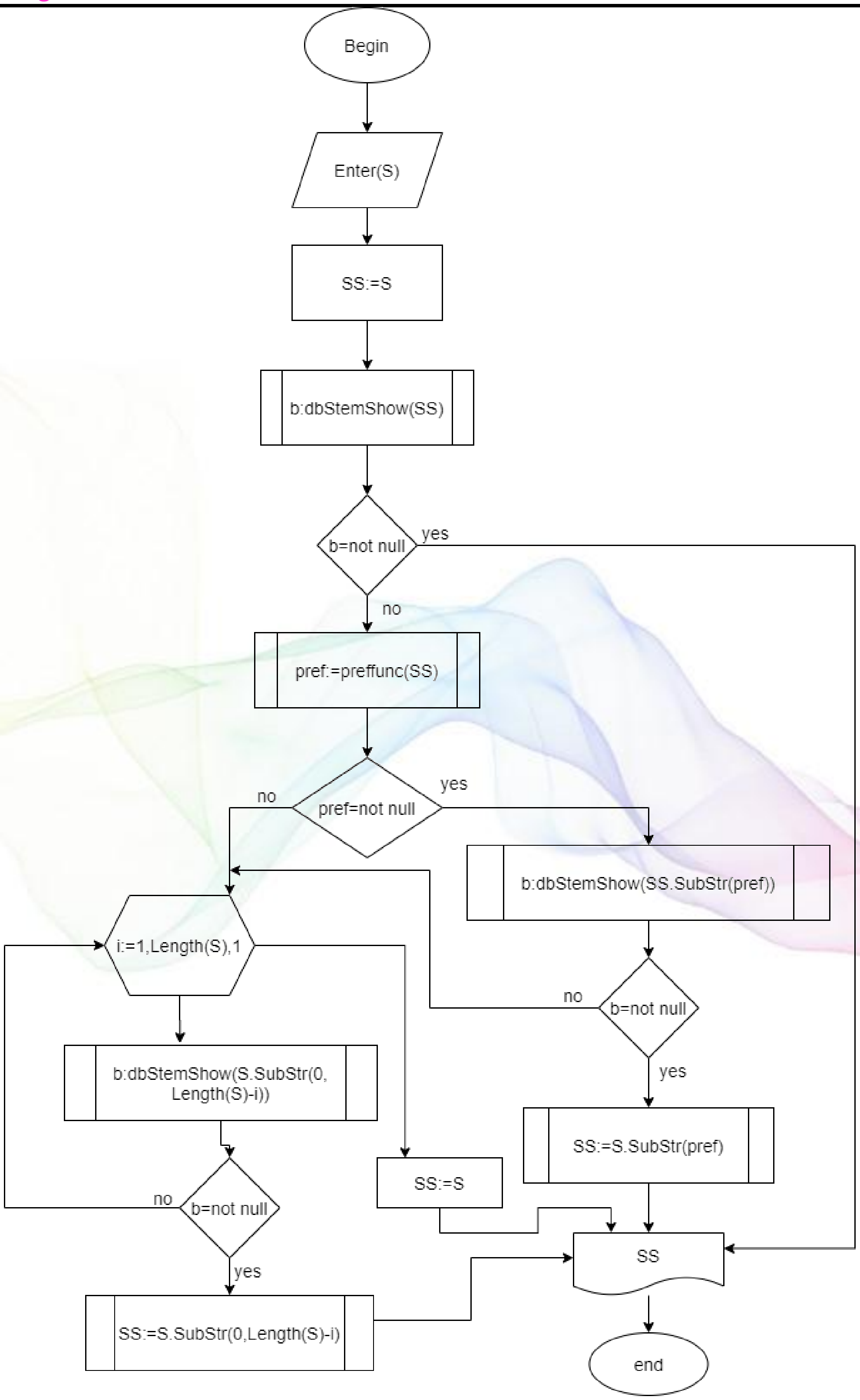

Fig. 4. StemWS Algorithm Block Diagram

Results and Discussion

To test the algorithm, we used text corpora by genre: fairy tales, legislative documents, Uzbek proverbs and religious works of Table 1.

Table 1.

№	Selected text corpora by genre	Number of words	Правильные основы
1	Fairy tales	1246	96%
2	Legislative documents	1290	92,5%
3	Religious works	1285	91,3%
4	Uzbek proverbs	1200	94%

Conclusions.

Thus, a method has been developed for normalizing texts in the Uzbek language based on the stemming algorithm. When developing the algorithm, a hybrid approach was used based on the joint application of an algorithmic method, a lexicon of linguistic rules and a database of normal forms of words of the Uzbek language. The accuracy of the proposed algorithm depends on the accuracy of the tokenization algorithm. At the same time, the issue of finding the roots of paired words separated by spaces was not considered here, since this problem is solved directly at the tokenization stage. The algorithm can be integrated into various automated systems for machine translation, information retrieval, information retrieval, etc.

References

1. Over 7000 languages are spoken in the world today, but not many are represented online // Consumers International. – URL: https://bit.ly/2ToxLQR.
2. Yampolskiy R.V. AI-Complete, AI-Hard, or AI-Easy: Classification of Problems in Artificial Intelligence // Midwest Artificial Intelligence and Cognitive Science Conference. – Cincinnati, 2012. – P. 1-8. – URL: https://bit.ly/3vDqhI1.
3. Razmi N.A. et al. Visualizing stemming techniques on online news articles text analytics // Bulletin of Electrical Engineering and Informatics. – 2021. – Vol. 10. – №1. – P. 365-373.
4. Rakhimova D.R., Turganbaeva A.O. Normalization of kazakh language words // Sci. Tech. J. Inf. Technol. Mech. Opt. – 2020. – Vol. 20, №4. – P. 545-551.
5. Sharifloo A., Shamsfard M. A Bottom Up approach to Persian Stemming // Int'l Joint Conf. on Natural Language Processing : Vol. 2. – Hyderabad, 2008. – P. 583-588. 6. Hassan Diyanati M. et al. Words Stemming Based on Structural and Semantic Similarity // Comput. Eng. Appl. – 2014. – Vol. 3, №2. – P. 89–99.
6. Bakaev I.I. Linguistic features tokenization of text corpora of the Uzbek language // Bulletin of TUIT: Management and Communication Technologies. – 2021. Vol. . – P. 1-8.
7. Bakaev I.I. Development of a stemming algorithm for words of the Uzbek language/ Cybernetics and Programming. - 2021. - No. 1. - P. 1 - 12. DOI: 10.25136 / 2644-5522.2021.1.35847 URL: https://nbpublish.com/library_read_article.php?id=35847