Linguistic Analysis Method of Ukrainian Commercial Textual Content for Data Mining

Oleg Bisikalo[1][0000-0002-7607-1943], and Victoria Vysotska[2][0000-0001-6417-3689]

[1]Vinnytsia National Technical University, Vinnytsia, Ukraine
obisikalo@gmail.com
[2]Lviv Polytechnic National University, Lviv, Ukraine
Victoria.A.Vysotska@lpnu.ua

Abstract. This article deals with the scientific and practical task of automatically detecting significant keywords and rubricating Ukrainian content in Internet systems based on the method of linguistic analysis of text information. The article presents theoretical and experimental substantiation of the method of linguistic analysis of Ukrainian content using Porter's stemming. The method is aiming to automatically detect significant keywords of Ukrainian content on the basis of the proposed formalization of components of analysis - grammatical (grapheme), morphological, syntactic, semantic, referential, and structural.

Keywords: Text, Ukrainian, Algorithm, Content Monitoring, Keywords, Content analysis, Porter's Stemmer, Linguistic Analysis, Syntactic Analysis.

1 Introduction

In practical terms, the analysis of the symbolic level of the organization of naturalistic text is limited to the separation of syntactic punctuation from the word itself, the allocation of abbreviations, abbreviations, etc. Analysis of existing texts shows that at the level of sign organization of the text one is using the descriptive capabilities of the semiotic system to encode knowledge about fragments of real reality. For instance, the use of quotation marks (for example, the movie theater “Star”) indicates that the token in quotation marks cannot be considered in the meaning given in the dictionary. The proper names given in the text may coincide with the spelling of common words, but have different meanings (for example, the group Black September, Black Friday, student Sophia Vovk (wolf), assistant Andriy Krolik (rabbit), teacher Nadiya Kohut (cock), singer Katya Chile, singer Vinnytska Alona, Actor David Duchovny as Fox William Mulder, Liberty Avenue, May 1 Street, Actress Sarah Gabriel, Actress Nastya Zadorozhna, etc.). In addition, a number of tokens in the text are not subject to the grammatical rules of the language, but act as semantic units of sign level (for example, the number 30, the percentage value of 15%, the reduction of millions, thousand or kg, etc.). These features of naturalistic text make it necessary to develop a significant level of text organization as the initial stage of building a model for under-

standing text. A linguistic method of processing textual information to automatically detect meaningful keywords consists of six steps.

1. Grammatical (graphemic) analysis of textual content that is, parsing text with regard to the features of graphs of different languages.
2. Morphological analysis of textual content.
3. Syntax analysis of textual content.
4. Semantic analysis of textual content.
5. Reference analysis for the formation of interphase unities.
6. Structural analysis of textual content.

The input of the graphemic (two-graphemic) analysis or parsing step is the current text file and the a priori reference models (lines and characters). Separation of such units of text as a name, designation, title, etc. allows us to identify at this stage some functional elements of the structure of concepts. Therefore, it is advisable to begin with an analysis of character-level text to solve the urgent problem of forming effective domain-specific knowledge recognition procedures. The electronic component of this is electronic dictionaries of abbreviations, geographical names, names. This approach is caused by the diversity of sign (grapheme) representation of lexical units in the text, which defines their different semantic functions in one context or another. For the automated processing of naturalistic information, it is also essential to define the structure of the text - to separate service information, highlight paragraphs, headings, and more. The text is considered as a sort of organized sequence of lines and graphemes.

2 Relate the Highlighted Issue to Important Scientific and Practical Work

The article deals with the scientific and practical task of automatically detecting significant keywords and rubricating Ukrainian-language content in Internet systems based on the method of linguistic analysis of textual information. The work was performed within the framework of joint scientific researches of the Department of Information Systems and Networks of the Lviv Polytechnic National University on the topic «Research, development and implementation of intelligent distributed information technologies and systems based on database resources, data warehouses, data spaces and knowledge in order to accelerate the formation processes of modern information society», as well as the department of automation and information-measuring technique of Vinnytsia National Technical University within spine Research Center of Applied and Computational Linguistics. The results of the research were carried out within the framework of the state budget research works on the topics "Development of methods, algorithms and software for modeling, designing and optimization of intellectual information systems based on Web technologies «WEB» and "Intelligent information technology of image analysis of text and synthesis of integrated knowledge base language content". Scientific research was also carried out

within the framework of the initiative topics of the ISM Department of Lviv Polytechnic National University on the development of intelligent distributed systems based on an ontological approach to integrate information resources.

3 Analysis of Recent Research and Publications

Text content (article, commentary, book, etc.) contains a considerable amount of data in natural language, some of which is abstract [1-7]. The text is presented as a unified sequence of character units, the main properties of which are information, structural and communicative connectivity / integrity, which reflects the content / structure of the text [8-22]. The method of text processing is linguistic content analysis (e.g., comments, forums, articles, etc.) [23-30]. The process of text processing divides the content into tokens using finite state machines (Fig. 1).

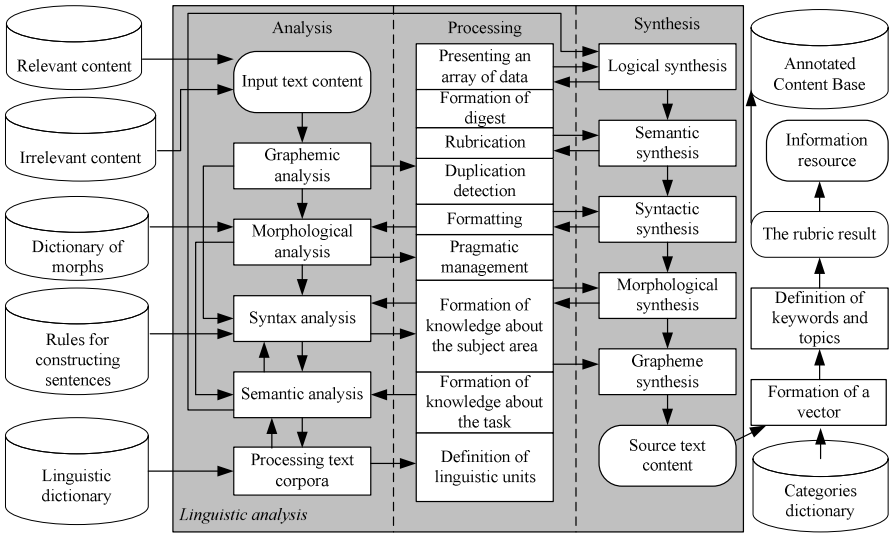


Fig. 1. Structural diagram of linguistic analysis of textual content

4 Analysis of Scientific Results

As a functional-semantic-structural unity, the text conforms to the rules of construction, reveals the regularities of content and formal connection of constituent units. Cohesiveness is manifested through external structural indicators and formal dependence of the text components, and integrity through thematic, conceptual and modal dependence. Integrity leads to a meaningful and communicative organization of text, and coherence to a form, a structural organization. Therefore, it is proposed to analyze the multilevel content structure in the analysis: linear sequence of characters; linear

sequence of morphological structures; linear sequence of sentences; a network of interconnected unities (Alg. 1).

Algorithm 1. Linguistic analysis of textual content.

Stage 1. Grammatical (graphemic) analysis of textual content C_1.

Step 1. Divide textual commercial content $C_1 \to C_2$ into sentences and paragraphs.

Step 2. Divide the content character chain C_2 into words.

Step 3. Allocate numbers, numbers, dates, unchanged turns, and content cuts C_2.

Step 4. Publish non-text content C_2 characters.

Step 5. Formation and analysis of linear content enhancement technology for content C_2.

Stage 2. Morphological analysis of textual content C_2.

Step 1. Obtained the basic (word form with cut offs).

Step 2. A grammatical category is formed for different words (collection of grammatical meanings: rarity, deviation, deviation).

Step 3. Formation of linear ability of morphological structure.

Stage 3. Syntax analysis $\alpha_4 : (C_2, U_K, T) \to C_3$ of textual content C_2.

Stage 4. Semantic analysis of textual content C_3.

Step 1. The word matches the semantic classes in the dictionary.

Step 2. Selection of morphosemantic alternatives required for this review.

Step 3. Cut the words into a single structure.

Step 4. Generate an orderly number of superposition entries with basic lexical functions and semantic classes. The accuracy of results is the most commonly used / corrective dictionary.

Stage 5. Reference analysis for interphase unities.

Step 1. Contextual analysis of text commercial content C_3. With it, the resolution of local references (the one that is, his) is realized and the expression of the expression is the kernel of unity.

Step 2. Thematic analysis. Separation of statements on a theme and a rheum allocates thematic structures which are used, for example, at formation of a digest.

Step 3. Determine the regular repetition, synonymization and re-nomination of keywords; the identity of the reference, that is, the ratio of words to the subject of the image; presence of implication based on situational connections.

Stage 6. Structural analysis of textual content C_3. The prerequisites for use are a high degree of coincidence of terms of unity, a discursive unit, a sentence in a semantic language, utterance, and an elementary discursive unit.

Step 1. Identify the basic set of rhetorical connections between content unities.

Step 2. Building a nonlinear unity network. The openness of a link set involves its extension and adaptation to analyze the structure of the text C_3.

Let us consider in detail each of the stages of the proposed algorithm.

Step 1. Grammatical (graphemic) analysis of textual content. The grapheme is called the minimum content unit of written text. The objective of this level of recognition is to build a formalized representation of the grapheme structure of the text and to

develop a formal apparatus for separating and classifying text units on multiple lines and graphs. Generalized recognition algorithm works with certain restrictions on the input text: formatted width; does not contain hyphenation; Does not contain objects as a table, figure, formula or graphic symbol; submitted in known languages, such as English, Ukrainian, and German, rather than Ancient Egyptian, Mongolian, or Elven. The ultimate goal of recognizing the graphemic level of text representation is to build a grapheme structure of text, which includes separating on a plurality of lines and graphemes of input such semantically independent units of text as fragments (discourses), sentences, syntagms, tokens, and defining the types (classes) of enumerated units of text and units the relationship between them in a specific input text. The process of recognition at the grapheme level of the text representation involves two stages, as shown in Fig. 2.

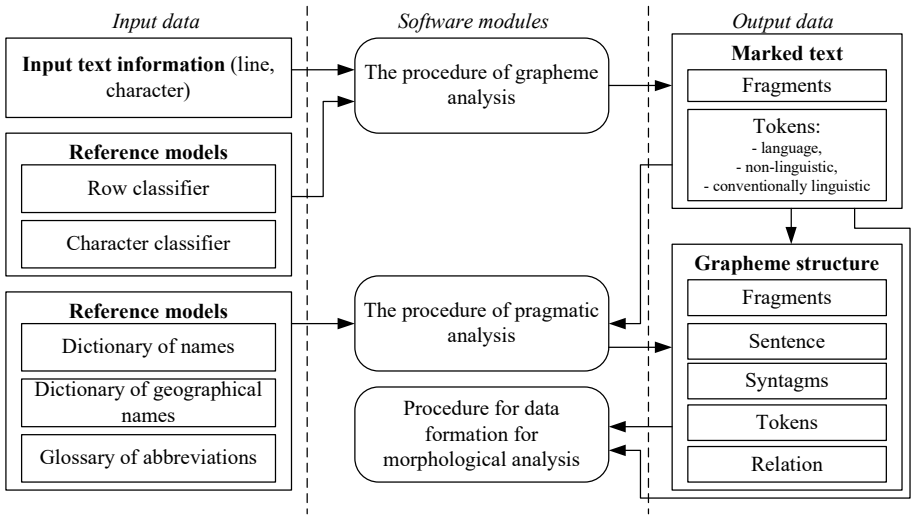


Fig. 2. Structural-logistical scheme of recognition of knowledge from the subject area at the graphemic stage of textual information analysis

The purpose of the first stage is to separate substantively separate fragments in the text, tokens in each fragment of the text and determine the language of the input text and / or fragments of the text. The input of the first stage is the current text file and a priori reference models of rows and graphs. The string classifier includes the following significant classes: empty string (EmpStr), full string (full string, FulStr), incomplete right (IncRgt), incomplete left (IncLgt), symmetric incomplete (SmtInc). The rules for recognizing lines in the text are given in Table. 1. Many reference models of graphems are conveniently presented in the normal Backus-Naur (BPF) form, the abbreviations of which are given in the table. 2. Consider grammar $G = < V, T, S, P >$, where the alphabet is $V = < Gr, T >$; terminal symbols are $T : = <$ A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, Q, P, R, S, T, U, W, V, X, Y, Z, Ä, Ö, Ü, Ą, Ć, Ę, Ł, Ń, Ó, Ś, Ź, Ż, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, w, v, x, y, z, ä, ö, ü, ß, ą, ć, ę, ł, ń, ó, ś, ź, ż, А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш,

Щ, Ь, Ю, Я, Є, І, Ї, Ґ, Ы, Э, Ъ, а, б, в, г, д, е, ж, з, и, й, к, л, м, о, н, п, р, с, т, у, ф, х, ц, ч, ш, щ, ь, ю, я, є, і, ї, г, ы, э, ъ, '> and a list of tuples for those defined in Table 2 elements of the set of reference models of graphems:

Table 1. Rules for recognizing lines in text in the grapheme stage

Line name	First position	Last position	All positions
Empty string (EmpStr)	–	–	Gap
Full string (*FulStr*)	Symbol	Symbol	–
Incomplete right (*IncRgt*)	Symbol	Gap	–
Incomplete left (*IncLgt*)	Gap	Symbol	–
Symmetric incomplete (*SmtInc*)	Gap	Gap	–

Table 2. Denomination of the elements of the set of reference models of graphems

N	Name	Abbr.	N	Name	Abbr.
1	Grammar	G	35	Ukrainian small letter	*Usm*
2	Alphabet	*V*	36	Russian capital letter	*Rcp*
3	Term	*T*	37	Russian small letter	*Rsm*
4	Initial character	*S*	38	Consonant letter	*Cnl*
5	Production rules	*P*	39	Vowel letter	*Vwl*
6	Symbol	*Sb*	40	Latin capital consonant letter	*Lcc*
7	Space	*Sp*	41	Latin small consonant letter	*Lsc*
8	Digit	*Dgt*	42	Latin capital vowel letter	*Lcv*
9	Special symbol	*Ssb*	43	Latin small vowel letter	*Lsv*
10	Syntactic sign	*Ssg*	44	Cyrillic capital consonant letter	*Ccc*
11	Letter	*Ltr*	45	Cyrillic small consonant letter	*Csc*
12	Latin letter	*Lat*	46	Cyrillic capital vowel letter	*Ccv*
13	Cyrillic letter	*Cyr*	47	Cyrillic small vowel letter	*Csv*
14	English alphabet	*Eng*	48	English capital consonant letter	*Ecc*
15	German alphabet	*Ger*	49	English small consonant letter	*Esc*
16	Polish alphabet	*Pol*	50	English capital vowel letter	*Ecv*
17	Ukrainian alphabet	*Ukr*	51	English small vowel letter	*Esv*
18	Russian alphabet	*Rus*	52	German capital consonant letter	*Gcc*
19	Official symbol	*Osb*	53	German small consonant letter	*Gsc*
20	Brackets	*Bsb*	54	German capital vowel letter	*Gcv*
21	Mathematical symbol	*Msb*	55	German small vowel letter	*Gsv*
22	Capital letter	*Cpl*	56	Polish capital consonant letter	*Pcc*
23	Small letter	*Sml*	57	Polish small consonant letter	*Psc*
24	Latin capital letter	*Lcp*	58	Polish capital vowel letter	*Pcv*

N	Name	Abbr.	N	Name	Abbr.
25	Latin small letter	*Lsm*	59	Polish small vowel letter	*Psv*
26	Cyrillic capital letter	*Ccp*	60	Ukrainian capital consonant letter	*Ucc*
27	Cyrillic small letter	*Csm*	61	Ukrainian small consonant letter	*Usc*
28	English capital letter	*Ecp*	62	Ukrainian capital vowel letter	*Ucv*
29	English small letter	*Esm*	63	Ukrainian small vowel letter	*Usv*
30	German capital letter	*Gcp*	64	Russian capital consonant letter	*Rcc*
31	German small letter	*Gsm*	65	Russian small consonant letter	*Rsc*
32	Polish letter	*Pcp*	66	Russian capital vowel letter	*Rcv*
33	Polish small letter	*Psm*	67	Russian small vowel letter	*Rsv*
34	Ukrainian capital letter	*Ucp*			

1. *Gr* : = < *Sb* ∪ *Sp* > is recognized text content as a set of characters and spaces;
2. *Sp* : = <_> is space as terminal symbol;
3. *Sb* : = < *Ltr* ∪ *Dgt* ∪ *Ssb* ∪ *Ssg* > is plural letters, numbers, special characters, and syntax characters;
4. *Ltr* : = <*Lat*∪*Cyr*∪*Eng*∪*Ger*∪*Pol*∪*Ukr*∪*Rus*∪*Cpl*∪*Sml*∪*Cnl*∪*Vwl*∪‘> is a set of Latin, Cyrillic, English, German, Polish, Ukrainian, Russian letters, including uppercase and lowercase letters of the respective languages, consonants and vowels, as well as apostrophe as a terminal symbol;
5. *Dgt* : = < 0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9 > is a set of numbers;
6. *Ssb* : = < *Osb* ∪ *Bsb* ∪ *Msb* > is service characters set, brackets and math symbols;
7. *Ssg* : = <«∪»∪“∪”∪,∪.∪:∪;∪-∪?∪!> is terminal symbols of syntactic characters;
8. *Cpl* : = < *Lcp* ∪ *Ccp* ∪ *Ecp* ∪ *Gcp* ∪ *Pcp* ∪ *Ucp* ∪ *Rcp* > is a set of capital letters of the respective languages;
9. *Sml* : = < *Lsm* ∪ *Csm* ∪ *Esm* ∪ *Gsm*∪ *Psm* ∪ *Usm*∪ *Rsm* > is a set of lowercase letters of the respective languages;
10. *Lat* : = < *Lcp* ∪ *Lsm* > is a set of Latin letters both capital and small;
11. *Cyr* : = < *Ccp* ∪ *Csm* > is a set of Cyrillic letters both capital and small;
12. *Eng* : = < *Ecp* ∪ *Esm* > is a set of English letters both capital and small;
13. *Ger* : = < *Gcp* ∪ *Gsm* > is a set of German letters both capital and small;
14. *Pol* : = < *Pcp* ∪ *Psm* > is a set of Polish letters both capital and small;
15. *Ukr* : = < *Ucp* ∪ *Usm* > is a set of Ukrainian letters both capital and small;
16. *Rus* : = < *Rcp* ∪ *Rsm* > is a set of Russian letters both capital and small;
17. *Osb* : = <№∪%∪/∪@∪#∪$∪&∪*∪\> is a set of terminal service characters;
18. *Bsb* : =< [∪] ∪ { ∪ } ∪ (∪) > is a set of terminal characters of brackets;
19. *Msb* : = < + ∪ < ∪ > ∪ = > is a set of terminal mathematical symbols;
20. *Cnl* : = < *Ecc* ∪ *Esc* ∪ *Gcc* ∪ *Gsc* ∪ *Pcc* ∪ *Psc* ∪ *Ucc* ∪ *Usc* ∪ *Rcc* ∪ *Rsc* > is a set of uppercase and lowercase letters of the respective languages;

21. *Vwl* : = < *Ecv* ∪ *Esv* ∪ *Gcv* ∪ *Gsv* ∪ *Pcv* ∪ *Psv* ∪ *Ucv* ∪ *Usv* ∪ *Rcv* ∪ *Rsv* > is a set of uppercase and lowercase letters of the respective languages;
22. *Lcp* : = < *Lcc* ∪ *Lcv* ∪ Q ∪ V ∪ X > is Latin capital letters;
23. *Lsm* : = < *Lsc* ∪ *Lsv* ∪ q ∪ v ∪ x > is Latin lowercase letters;
24. *Ccp* :. = < *Ccc* ∪ *Csv* ∪ Ь ∪ Й > is Cyrillic capital letters;
25. *Csm* : = < *Csc* ∪ *Csv* ∪ ь ∪ й > is Cyrillic lowercase letters;
26. *Ecp* : = < *Lcc* ∪ *Lcv* ∪ Q ∪ V ∪ X > is English capital letters;
27. *Esm* : = < *Lsc* ∪ *Lsv* ∪ q ∪ v ∪ x > is English lowercase letters;
28. *Gcp* : = < *Lcc* ∪ *Lcv* ∪ Ä ∪ Ö ∪ Ü ∪ Q ∪ V ∪ X > is German capital letters;
29. *Gsm* : = < *Lsc* ∪ *Lsv* ∪ ä ∪ ö ∪ ü ∪ ß ∪ q ∪ v ∪ x > - German lowercase letters;
30. *Pcp* : = < *Lcc*∪*Lcv*∪Ą∪Ć∪Ę∪Ł∪Ń∪Ó∪Ś∪Ź∪Ż > is Polish capital letters;
31. *Psm* : = < *Lsc*∪*Lsv*∪ą∪ć∪ę∪ł∪ń∪ó∪ś∪ź∪ż > is Polish lowercase letters;
32. *Ucp* : = < *Ccc* ∪ *Ccv* ∪ Є ∪ І ∪ Ї ∪ Ґ > is Ukrainian capital letters;
33. *Usm* : = < *Csc* ∪ *Csv* ∪ є ∪ і ∪ ї ∪ ґ> is Ukrainian lowercase letters;
34. *Rcp* : = < *Ccc* ∪ *Ccv* ∪ Ы ∪ Э ∪ Ъ > is Russian capital letters;
35. *Rsm* : = < *Csc* ∪ *Csv* ∪ ы ∪ э ∪ ъ> is Russian lowercase letters;
36. *Lcc* : = < B ∪ C ∪ D ∪ F ∪ G ∪ H ∪ J ∪ K ∪ L ∪ M ∪ N ∪ P ∪ R ∪ S ∪ T ∪ W ∪ Z> is terminal Latin capital consonant letters;
37. *Lcv* : = < A ∪ E ∪ I ∪ O ∪ U ∪ Y> is terminal Latin capital loud letters;
38. *Lsc* : = < b ∪ c ∪ d ∪ f ∪ g ∪ h ∪ j ∪ k ∪ l ∪ m ∪ n ∪ p ∪ r ∪ s ∪ t ∪ w ∪ x ∪ z > is terminal Latin lowercase consonant letters;
39. *Lsv*: = < a ∪ e ∪ i ∪ o ∪ u ∪ v ∪ y > is terminal Latin lowercase loud letters;
40. *Ccc* : = < Б ∪ В ∪ Г ∪ Д ∪ Ж ∪ З ∪ К ∪ Л ∪ М ∪ Н ∪ П ∪ Р ∪ С ∪ Т ∪ Ф ∪ Х ∪ Ц ∪ Ч ∪ Ш ∪ Щ > is terminal Cyrillic capital consonant letters;
41. *Csv* : = < А ∪ Е ∪ И ∪ О ∪ У ∪ Ю ∪ Я > is terminal Cyrillic capital loud letters;
42. *Csc* := < б ∪ в ∪ г ∪ д ∪ ж ∪ з ∪ к ∪ л ∪ м ∪ н ∪ п ∪ р ∪ с ∪ т ∪ ф ∪ х ∪ ц ∪ ч ∪ ш ∪ щ > is terminal Cyrillic lowercase consonant letters;
43. *Csv* : = < а ∪ е ∪ и ∪ о ∪ у ∪ ю ∪ я > is terminal Cyrillic lowercase loud letters.

The following production rules are offered to recognize the language of the text:

P := < S→S *Gr*, S→Λ, *Gr*→*Gr Sb*, *Gr*→*Gr Sp*, *Gr*→Λ, *Sp*→_, *Sb*→*Ltr*, *Sb* →*Dgt*, *Sb*→*Ssb*, *Sb*→*Ssg*, *Ltr*→*Lat*, *Ltr*→*Cyr*, *Ltr*→*Eng*, *Ltr*→*Ger*, *Ltr*→*Pol*, *Ltr*→*Ukr*, *Ltr*→*Rus*, *Ltr*→*Cpl*, *Ltr*→*Sml*, *Ltr*→*Cnl*, *Ltr*→*Vwl*, *Ltr*→‘, *Ssb*→*Osb*, *Ssb*→*Bsb*, *Ssb*→*Msb*, *Cpl*→*Lcp*, *Cpl*→*Ccp*, *Cpl*→*Ecp*, *Cpl*→*Gcp*, *Cpl*→*Pcp*, *Cpl*→*Ucp*, *Cpl*→*Rcp*, *Sml*→*Lsm*, *Sml*→*Csm*, *Sml*→*Esm*, *Sml* →*Gsm*, *Sml*→*Psm*, *Sml*→*Usm*, *Sml*→*Rsm*, *Lat*→*Lcp*, *Lat*→*Lsm*, *Cyr*→*Ccp*, *Cyr*→*Csm*, *Eng*→ *Ecp*, *Eng*→*Esm*, *Ger*→*Gcp*, *Ger*→*Gsm*, *Pol*→*Pcp*, *Pol*→*Psm*, *Ukr*→*Ucp*, *Ukr*→*Usm*, *Rus*→ *Rcp*, *Rus*→*Rsm*, *Lcp*→*Lcc*, *Lcp*→*Lcv*, *Lcp*→Q, *Lcp*→V, *Lcp*→X, *Lsm*→*Lsc*, *Lsm*→*Lsv*, *Lsm* →q, *Lsm*→v, *Lsm*→x, *Ccp*→*Ccc*, *Ccp*→*Csv*, *Ccp*→Ь, *Ccp*→Й, *Csm*→*Csc*, *Csm*→*Csv*, *Csm* →ь, *Csm*→й, *Ecp*→*Lcc*, *Ecp*→*Lcv*, *Ecp*→Q, *Ecp*→V, *Ecp*→X, *Esm*→*Lsc*, *Esm*→*Lsv*, *Esm*→ q, *Esm*→v, *Esm*→x, *Gcp*→*Lcc*, *Gcp*→*Lcv*, *Gcp*→Ä, *Gcp*→Ö, *Gcp*→Ü, *Gcp*→Q, *Gcp*→V, *Gcp*→X, *Gsm*→*Lsc*, *Gsm*→*Lsv*, *Gsm*→ä, *Gsm*→ö, *Gsm*→ü, *Gsm*→ß, *Gsm*→q, *Gsm*→v, *Gsm* →x, *Pcp*→*Lcc*, *Pcp*→*Lcv*, *Pcp*→Ą, *Pcp*→Ć, *Pcp*→Ę, *Pcp*→Ł, *Pcp*→Ń, *Pcp*→Ó, *Pcp*→Ś,

Pcp→Ź, *Pcp*→Ż, *Psm*→*Lsc*, *Psm*→*Lsv*, *Psm*→ą, *Psm*→ć, *Psm*→ę, *Psm*→ł, *Psm*→ń, *Psm*→ó, *Psm*→ś, *Psm*→ź, *Psm*→ż, *Ucp*→*Ccc*, *Ucp*→*Ccv*, *Ucp*→Є, *Ucp*→І, *Ucp*→Ї, *Ucp*→Ґ, *Usm*→ *Csc*, *Usm* →*Csv*, *Usm*→є, *Usm*→і, *Usm*→ї, *Usm*→ґ, *Rcp*→*Ccc*, *Rcp*→*Ccv*, *Rcp*→ЫІ, *Rcp*→Э, *Rcp*→Ъ, *Rsm*→*Csc*, *Rsm*→*Csv*, *Rsm*→ы, *Rsm*→э, *Rsm*→ъ, *Lcc*→B, *Lcc*→C, *Lcc*→D, *Lcc*→ F, *Lcc*→G, *Lcc*→H, *Lcc*→J, *Lcc*→K, *Lcc*→L, *Lcc*→M, *Lcc*→N, *Lcc*→P, *Lcc*→R, *Lcc*→S, *Lcc* →T, *Lcc*→W, *Lcc*→Z, *Lcv*→A, *Lcv*→E, *Lcv*→I, *Lcv*→O, *Lcv*→U, *Lcv*→Y, *Lsc*→b, *Lsc*→c, *Lsc*→d, *Lsc*→f, *Lsc*→g, *Lsc*→h, *Lsc*→j, *Lsc*→k, *Lsc*→l, *Lsc*→m, *Lsc*→n, *Lsc*→p, *Lsc*→q, *Lsc* →r, *Lsc*→s, *Lsc*→t, *Lsc*→w, *Lsc*→x, *Lsc*→z, *Lsv*→a, *Lsv*→e, *Lsv*→i, *Lsv*→o, *Lsv*→u, *Lsv*→v, *Lsv*→y, *Ccc*→Б, *Ccc*→В, *Ccc*→Г, *Ccc*→Д, *Ccc*→Ж, *Ccc*→З, *Ccc*→К, *Ccc*→Л, *Ccc*→М, *Ccc*→Н, *Ccc*→П, *Ccc*→Р, *Ccc*→С, *Ccc*→Т, *Ccc*→Ф, *Ccc*→Х, *Ccc*→Ц, *Ccc*→Ч, *Ccc*→Ш, *Ccc*→Щ, *Csv*→А, *Csv*→Е, *Csv*→И, *Csv*→О, *Csv*→У, *Csv*→Ю, *Csv*→Я, *Csc*→б, *Csc*→в, *Csc*→г, *Csc*→д, *Csc*→ж, *Csc*→з, *Csc*→к, *Csc*→л, *Csc*→м, *Csc*→н, *Csc*→п, *Csc*→р, *Csc*→с, *Csc*→т, *Csc*→ф, *Csc*→х, *Csc*→ц, *Csc*→ч, *Csc*→ш, *Csc*→щ, *Csv*→а, *Csv*→е, *Csv*→и, *Csv*→ о, *Csv*→у, *Csv*→ю, *Csv*→я >.

These production rules are used to identify meaningful units of analysis $< U'_C, U'_G >$ text commercial content X' (phrase, sentence, theme, idea, author, character, social situation, part of the text, clustered in the content of the category of analysis) (STEP 1 parsing based on the language of the text fragments) by a modified Potter algorithm (STAGE 2 stemming). We have the following requirements for choosing a linguistic unit of analysis: great for interpreting value; small in order not to interpret many meanings; easily identified; the number of units is large to isolate the sample.

Stage 2. Morphological analysis of textual content is to find the basics of words, for example [8] cuts out suffixes, prefixes, etc., leaving only the basis of the word (stemming). There are known algorithms for finding the basics, for example [8] cuts out suffixes, prefixes, etc., leaving only the base of the word. They also cut out the key words with a simple word-selection function, then each of the words is recognized by the base and written into a table, for example: keywords. However, we have the disadvantage - we need to take into account all the rules of formation of words in the Ukrainian language (flexions depending on gender and pronunciation, parts of the language, suffixes, prefixes, alternation of words in the basis of pronunciation, singular and plural, etc.). For example, for such words from the set M = {пошук**ов**ими, користувач**ам, високо**рейтинг**ово**го, рейтинг**у**}, such algorithms do not work (blue indicates the reason why it does not work - was not included in the rules). Increasing the rules geometrically increases the workload on the processing processes, for example, the task of checking and defining keys for 100 articles a day requires you to check every word through the finisher, suffixes, etc. - the complexity of the algorithm increases to the critical limit. For English-language texts, the complexity is less - there are only two cases and one ending for nouns. Already for German the complexity is increasing - 4 letters, compound words are spelled together with 2, 3 and more words and more. In [8] the algorithm works for L = {*Автомат – Автомат, Автомата – Автомат, Автоматом – Автомат, Ресурсів – ресурс*}. But it is better not to find the root by cutting it off unnecessarily, but by having thematic dictionaries of the basics of key words to find in the text these basics of words, their distribution (more at the beginning, or at the end, or in the middle of the text), and frequency of use relative to the total volume. And through the basis of doing statistics, it is to calculate the

number of identical bases. There is a well-known algorithm for English-language texts - Porter's stemmer [9], but for Ukrainian texts it does not work perfectly.

Porter's Stemmer is a Stemming algorithm published by Martin Porter in 1980. The original version of Stemmer was designed for English and was written in BCPL. Subsequently, Martin created the Snowball project and, using the basic idea of the algorithm, wrote Stemmer for common Indo-European languages, including Russian [10-17]. The algorithm does not use the bases of words, but only, following a series of rules, cuts off endings and suffixes, based on the features of the language, and therefore works quickly, but not always error-free. The algorithm was very popular and duplicated, often changed by different developers, and not always successfully. Around 2000, Porter decided to "freeze" the project and continue to distribute a single implementation of the algorithm (in several popular programming languages) from his site [10-17]. For example, this algorithm takes into account in Ukrainian-language texts only the presence of an ending, and the suffixes - not then, the words search, search identifies, and search - no. The form of the flexion determines the type of word, for example,

```
var $ADJECTIVE =
'/(ими|ій|ий|а|е|ова|ове|ів|є|їй|єе|еє|я|ім|ем|им|ім|их|іх|ою|йми|іми|у|
ю|ого|ому|оїй)$/'; //http://uk.wikipedia.org/wiki/Прикметник +
http://wapedia.mobi/uk/Прикметник
 var $PARTICIPLE = '/(ий|ого|ому|им|ім|а|ій|у|ою|ій|і|их|йми|их)$/';
//http://uk.wikipedia.org/wiki/Дієприкметник
 var $VERB =
'/(сь|ся|ив|ать|ять|у|ю|ав|али|учи|ячи|вши|ши|е|ме|ати|яти|є)$/';
//http://uk.wikipedia.org/wiki/Дієслово
 var $NOUN =
'/(а|ев|ов|е|ями|ами|еи|и|ей|ой|ий|й|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|ы
|ь|ию|ью|ю|ия|ья|я|і|ові|ї|ею|єю|ою|є|еві|ем|єм|ів|їв|\'ю)$/';
//http://uk.wikipedia.org/wiki/Іменник
```

Features of the algorithm. The algorithm works with individual words, so the context in which the word is used is unknown. Linguistics categories such as word structure (root, suffix, etc.) and parts of language (noun, adjective, etc.) are also not available. We currently have the following techniques for analyzing words:

- The term is cut off from the word, for example, ending the *увати* turns the word *критикувати* into a *критик*.
- The word has an constant ending. Words with this ending remain unchanged. Example – *ск* and constant words *блиск*, *тиск*, *обеліск* and more.
- The word changes the ending. This rule applies to words in which certain letters fall out during cancellation (*ядро* and *ядер* – ending *ер* changes by *р*) or change (*чоловік* and *чоловіче* - *к* changes by *ч*).
- The word corresponds to a regular expression. This is an attempt to combine several rules into one difficult one. Perhaps this technique will not live up to the final version of the algorithm. But now, the code contains expressions similar to: (*ов*)*, *ува*(в|вши|вшись|ла|ло|ли|ння|нні|нням|нню|ти|вся|всь|лись|лися|тись|тися)

- The word does not change when it is being staged, but it is an exception to the rules. This is an undesirable case for the algorithm. It forces the vocabulary of exception words to hold. Examples of the *віче*, *наче*.
- The word changes during stemming, but is also an exception. This is the worst case for the algorithm because it forces two words to be stored in the dictionary at once: the original and the stemmed. For example, the word *відер* should be changed to *відр*, although other words ending as ep are not so categorized (*авіадиспетчер*, *вітер*, *гравер* etc.).
- Short words remain unchanged. Service parts of a language (prepositions, conjunctions, parts) are usually very short words that are ignored by the algorithm (words up to 2 letters inclusive).

All of these techniques are applied by groups that form the rules of stemming. But this significantly complicates the algorithm for finding keychains. Therefore, it is first suggested to consider common endings - not traditional endings, as part of a word, but the sequence of letters that end a word (Table 3-4). In the Table 3-4 endings of words 1 to 4 letters long are given. Five or more letters are not taken into account, since there are not enough such words (for the maximum of 5 *йтесь* (6837), for 6 - *ванням* (4656), etc.). This has created a kind of map for the project of stemming. The purpose of the project is to build a static termination tree and to capture the algorithm of all branches of the tree. Generally, a more detailed tree can be built [18-22], but for commercial content we choose a weighted level of detail - from 500 words with common ending. Consider in more detail the idea of a Porter's stemmer, namely finding the basis of a word for a given source word [23-30]. The algorithm does not use the bases of words, but works consistently using a number of rules for truncating endings and suffixes (Fig. 3).

Table 3. Statistic table of common word-endings

Word-endings (number)					
я (164062)	ння (9001)	мось (20536)	али (10666)	ному (19112)	ові (17191)
ся (148160)	в (32681)	лось (10231)	ними (19089)	о (90454)	сті (8731)
лися (10338)	ня (9765)	тись (10366)	м (119779)	мо (33568)	ості (7636)
ося (30769)	а (68134)	лись (10337)	т (2980)	го (31445)	ю (80877)
ься (25211)	ь (151355)	тесь (19105)	ім (31343)	ло (17238)	ою (39616)
ися (21940)	сь (111459)	лась (10229)	им (31166)	ймо (11229)	ню (10075)
еся (19105)	ть (33055)	ість (7606)	ам (20154)	ємо (11136)	ною (20280)
шся (11775)	ось (30788)	и (123402)	ом (17018)	ого (31389)	кою (7497)
ася (10235)	ись (22656)	ми (62080)	ям (15717)	ало (10465)	нню (9054)
вся (10076)	есь (19114)	ти (20025)	нім (19333)	ного (19090)	стю (7648)
юся (8044)	ась (10239)	ли (17711)	ним (19093)	і (90275)	у (94504)
лася (10230)	вся (10016)	ими (31121)	ням (9434)	ні (31679)	му (35023)
мося (20532)	сть (7688)	ами (20106)	нням (8975)	ві (22543)	ну (23125)

Word-endings (number)					
лося (10233)	юсь (8047)	ями (9844)	ку (11624)	ті (12596)	ній (19549)
ться (25036)	ють (11222)	ати (10819)	ому (31585)	нні (9909)	ний (19042)
их (31127)	ах (20023)	ях (9855)	них (19092)	ї (34702)	ої (31421)
тися (10379)	ів (15898)	ав (10547)	ш (19163)	єш (11138)	є (11466)

Table 4. Statistic word-endings tree whose total weight is less than 1%

Word-endings (number)				
р (2709)	ч (959)	г (636)	п (341)	щ (110)
н (2531)	с (914)	з (581)	б (281)	ц (34)
д (1038)	л (754)	ж (353)	ф (214)	ґ (4)

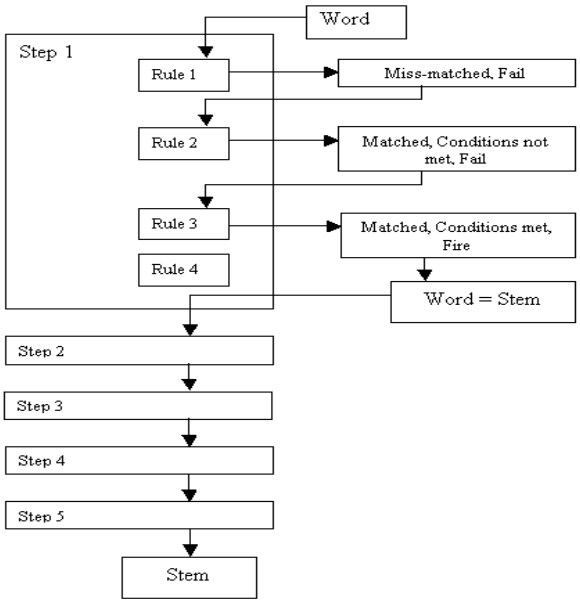


Fig. 3. Structural diagram of Porter's stemmer algorithm

First, let's introduce some definitions:

- **Vowels letters** are *а*, *е*, *і*, *ї*, *о*, *у*, *и*, *е*, *ю*, *я*.
- **RV** is part of the word after the first vowel. It is empty if there are no vowels in the word.
- **R1** is part of the word after the first combination is vowel-consonant.
- **R2** is part of **R1** after the first combination is vowel-consonant.

For example, in the word *інформаційний*: RV = *нформаційний*, R1 = *формаційний*, R2 = *маційний*. Now let's define several classes of word endings, leaving their original names in the original description of the algorithm.
Class 1. PERFECTIVE GERUND

- **Group 1**: *в*, *вши*, *вшися*. The ending should be preceded by the letter a or я.
- **Group 2**: *ив*, *ивши*, *ившися*.

Class 2. ADJECTIVE
а, е, *і*, *и*, *ими*, *іми*, *ій*, *ий*, їм, *ім*, *им*, *ього*, *ого*, *ьому*, *ому*, *їх*, *их*, *ую*, *ою*, *ая*, *яя*, *ою*, *єю*.
Class 3. PARTICIPLE

- **Group 1**: *вш*, *ува, ува, уч, юч, л*. The ending should be preceded by letter a or я.
- **Group 2**: *нн*, *н*, *ячи*, *ачи*, *ова, ову, єм*.

Class 4. REFLEXIVE are *ся*, *сь*.
Class 5. VERB

- **Group 1**: *ла*, *є*, *єте*, *йте*, *ли*, *люю*, *й*, *в*, *єм*, *ємо*, *ний*, *ло*, *ть*, *но*, *ють*, *ні*, *ть*, *єш*. The ending should be preceded by the letter a or я.
- **Group 2**: *ила*, *ела*, *ена*, *йте*, *ите*, *єте*, *юй*, *уй*, їй, *ай*, *ало*, *ив*, *или*, *имо*, *ений*, *ило*, *їло*, *ено*, *ють*, *ать*, *ені*, *ять*, *іть*, *ить*, *иш*, *ую*, *ю*.

Class 6. NOUN are *а*, *ев*, *ов*, *і*, *тя*, *е*, *ами*, *іями*, *ями*, *єї*, *єю*, *ями*, *ям*, *її*, *и*, *ою*, *ій*, *ой*, *ий*, *й*, *им*, *им*, *ім*, *ам*, *ом*, *о*, *у*, *ах*, *ях*, *ую*, *ю*, *ія*, *я*.
Class 7. SUPERLATIVE (*найдовший*, *миліший*, *більший*) are *ш*, *іш*.
Class 8. DERIVATIONAL (*милість*, *щедрість*, *малість*, *крайність*) is *ість*.
Class 9. ADJECTIVAL defined as ADJECTIVE or PARTICIPLE + ADJECTIVE. For example: *падюча = пада + юч + а*.

Rules. When looking for an ending, the longest one is chosen. For example, in the word *інформація ія* needs to be chosen, not *я*. All inspections are conducted on a part **RV**. So, when checking for PERFECTIVE GERUND the previous letters а and я should also be inside the **RV**. Letters before **RV** don't take part in inspections at all.

Step 1. To find and ending of PERFECTIVE GERUND. If it exists, then delete it and complete step. In other words, delete the ending of REFLEXIVE (if exists). Then in the following order check and if there is an ending delete: ADJECTIVAL, VERB, NOUN. Once one of them is found, then the step is completed.

Step 2. If the word ends with *і* – delete *і*.

Step 3. If in the **Step 2** there will be an end DERIVATIONAL, then delete it.

Step 4. One of three options is possible:

1. If the word ends with *н*, delete the last letter.
2. If the word ends with SUPERLATIVE, delete it and delete the last letter again if the word ends with *н*.
3. If the word ends with ь, delete it.

Stage 3. Syntax analysis of textual content. Syntax is known to be a set of rules that make it possible to construct formulas and recognize the correct formulas in the sequence of characters. It is important for the symbolic computation system that all but one of the expression logic operations are binary. This will be based on a parser. We will consider the process of revising the input sequence of characters in order to parse the grammatical structure according to the given formal grammar. A parser is a program or part of a program that performs parsing [10]. Generally (not just in the computer industry), the term syntactic parsing means the breakdown of text into parts of a language with the identification of their forms, purpose and syntactic relationship with other parts. This is largely determined by the stage of learning the differences and positioning parts of a particular language that can be quite difficult to formalize in inflected languages [22]. It is not at all easy to parse the sentences of such languages. For example, there are significant ambiguities in the structure of human language, that is, words and expressions that can themselves convey meaning in a vast number of variants, but only one of the meanings is relevant in a particular case. The success of choosing the right value in the vast majority of cases depends on many factors of contextual content, and it is almost impossible to predict all combinations of meaning. It is difficult to prepare formal rules for describing informal behavior, although, of course, there are strict rules, many of which form the basis of the grammar that forms the basis of the parser. During parsing, the text is framed into a data structure, usually a tree that matches the syntax structure of the input sequence and is well suited for further processing. As a rule, parser work in two stages: the first identifies meaningful tokens (lexical analysis is performed), the second creates a parse tree. For example (Fig. 4), for arithmetic expression 1+ 2*3:

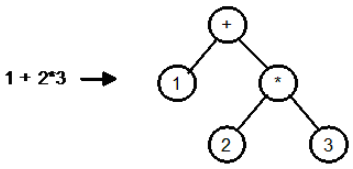


Fig. 4. An example of parsing an expression into a tree

A token is a sequence of one or more characters that stand out as an atomic object. The process of forming tokens is called tokenization or lexical analysis. Tokens are distinguished on the basis of the basic rules of the lexical analyzer (or lexer), which often differ depending on the scope [22]. Tokens are often classified by the position (location) of characters in the character sequence or context in the data stream. This is not just about highlighting a group of characters that are delimited by punctuation on either side (spaces or punctuation). Tokens are defined by the token rules and include grammatical elements of the language used in the data stream. In natural languages, these are usually categories of nouns, verbs, adjectives, or punctuation. The categories are used in the further processing of tokens with a parser or other functions in the program. The tasks of lexical analysis are as follows [20]:

- Convert a character set to a token sequence.

- Highlight each token as a logical part of the text (keyword, variable name, punctuation mark, etc.).
- Matching token and token - specific token text (“for”, “variable”, “;”, etc.).
- Selection of additional token attributes (e.g. variable value).
- Formation of an output token sequence that will be used by the parser as input.

The lexical analyzer usually does nothing with the combination of tokens it has allocated. For example, a typical lexical analyzer recognizes parentheses as characters, but does not check for each open parenthesis "(" closed parenthesis ")". This task remains for the parser or parser.

Step 4. Semantic analysis of textual content. Latent-semantic analysis is a method of processing information in natural language that allows you to analyze the relationship between a collection of documents (messages, articles, i.e. textual content) and terms (keywords) that occur in them. Compares some factors (topics) to all documents and terms. Initially, words correlate with semantic vocabulary classes. Then, the morphosemantic alternatives needed for this sentence are selected. What follows is the linking of words into a single structure and the formation of an ordered set of superposition entries of basic lexical functions and semantic classes. The accuracy of the result is determined by the completeness / correctness of the dictionary.

Step 5. Reference analysis for the formation of interphase unities. A contextual analysis of textual content is carried out. With it, the resolution of local references (the one that is, his) is realized and the expression of the expression is the kernel of unity. The following is a thematic analysis. Separation of statements on a theme and a rheum allocates thematic structures which are used, for example, at formation of a digest. Define regular repeatability, synonymization and re-nomination of keywords; the identity of the reference, that is, the relation of the words to the object of reflection; presence of implication based on situational connections.

Step 6. Structural analysis of textual content. The prerequisites for use are a high degree of coincidence of terms of unity, a discursive unit, a sentence in a semantic language, utterance, and an elementary discursive unit. A basic set of rhetorical relationships between content unities is identified and a nonlinear unity network is constructed [1-7]. The openness of a link set involves its extension and adaptation to analyze the structure of the text. There are several ways to use semantic analysis to define keywords as a phrase, that is, to define terms $Noun \in U_{K1}$ is nouns, noun phrases, or noun adjective among a plurality of text content words. For example, by the rules:

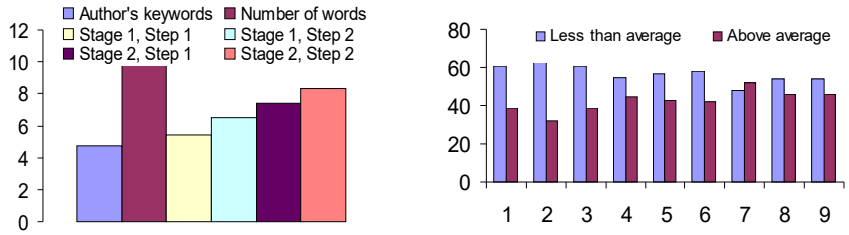
1. If the keyword is an adjective (flexion of the word *uй* - masculine noun). Then all the words used to the right of this adjective in any case are found in the text (the search follows the basis of this adjective) and a frequency dictionary is built for them. Those phrases that use more than a certain limit (but can be used less than the adjective itself) and are new keywords. The limit is determined by moderator.
2. If the keyword is a noun (flexion of the word is not *uй*), then all words to the right and left of it are analyzed.

3. First, all words to the left of him are checked for flexion *uй*. A frequency dictionary is also being built. It is determined by the set of words that occur most often with a certain moderator defined limit - these are the new keywords.
4. Then all the words on the right are analyzed - they should all be without flexion *uй*. Similarly, a frequency vocabulary is used to define many keywords.

Experimental studies. 100 scientific publications of the Bulletin of the Lviv Polytechnic National University of the series "Information Systems and Networks" (http://science.lp.edu.ua/sisn) from 783 and 805 (http://science.lp.edu.ua/SISN/SISN-2014, http://science.lp.edu.ua/sisn/vol-cur-805-2014-2) were selected as the linguistic base for the experimental study of the proposed method. The analysis of statistics of functioning of the system of detection of a set of keywords from 100 scientific articles was carried out in two stages, in particular:

1. Analyze all articles by checking common block words and thematic vocabulary.
2. Analyze all articles by checking refined blocked words and refined thematic vocabulary (with more startup, a set of unknown words (missing both in the thematic dictionary and in many blocked ones) is formed.

In addition, at each stage, the review was performed in two steps for each article: analysis of the entire article (http://victana.lviv.ua/index.php/kliuchovi-slova) and analysis of the article without beginning (title, authors, editors, annotations in two languages, authors keywords in two languages, authors' place of work), and no literature list to determine errors in the accuracy of multiple keyword formation (Fig.5).



a) b)

Fig. 5. The results of the review of 100 articles

In Fig. 5a) the diagram of analysis of statistics of formation by set of sets of all potential keywords in comparison with the set defined by authors of articles is presented. The first column is the average number of keywords identified by the author (4.77), and the second is the average number of words that make up those author keywords (9.82). The third column is the arithmetic mean of the potential keywords defined systematically in Step 1, Step 1 (5.46); the fourth is in step 1, step 2 (6.51); the fifth - in step 1, step 1 (7.43); sixth - in step 2, step 2 (8.35). Label these columns accordingly $A_1 \div A_6$. The value A_3 is different from the value A_1 by 0.69 (in quantity but not

in content); accordingly A_4 is different from A_1 by 1.74; A_5 from A_1 by 2.66; A_6 from A_1 by 3.58. The value A_2 is different from the value A_3 by 4.36; accordingly A_2 from A_4 by 3.31; A_2 from A_5 by 2.39; A_2 from A_6 by 1.47. Therefore, on average, the author of an article defines fewer keywords than is actually present in this work. Adjusting the system parameters increases the number of defined keywords by almost 2 times (with a similar comparison the value A_1 with he value A_3 more at 1.144654; A_4 – at 1.36478; A_5 – at 1.557652; A_6 – at 1.750524). The total increment of the value obtained by the system depending on the moderation of the dictionaries is accordingly A_3 – 14.46541; A_4 – 36.47799; A_5 – 55.7652; A_6 – 75.05241. If consistently compare A_2 with $A_3 \div A_6$ (in how many times the value A_2 is greater), then we get accordingly the range 1.7985; 1.5084; 1.3217; 1.176.

In Figure 5, b) shows a chart that contains statistics on more detailed textures in the analyzed articles, where 1 is an analysis of the page of the article page (husband's manual and average reference), 2 is paragraphs in the article, 3 is lines of text, 4 is words. 5 is characters, 6 is characters and spaces, 7 is words on the page, 8 - characters on the page, 9 - characters and prints on the page.

In Figure 6 shows the distribution diagram of the set of sets of all potential keywords for each article compared to the set defined by the authors of the articles.

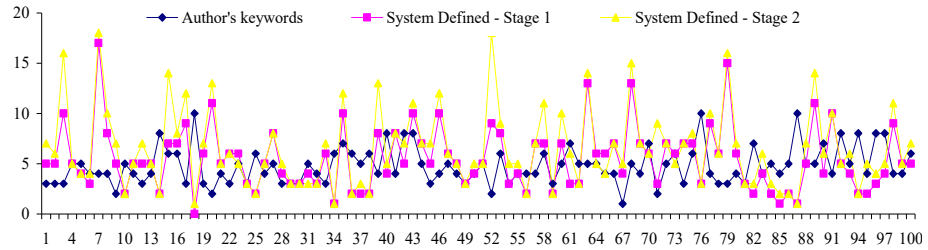


Fig. 6. The results of the inspection of 100 articles

Keyword accuracy is enhanced during the dictionary moderation process. The difference between the number of keywords identified by the author and the system defined in step 1, step 1 is 44.39919% (the percentage difference). The accuracy is improved in step 1, step 2 is 33.70672%, it is significantly improved in step 2, step 1 is 24.33809%, and in step 2, step 2 is already 14.96945%.

In the Table 5 the results of an analysis of the statistics of the set of sets of all potential keywords for each article compared to the set defined by the authors of the articles are presented, where A is for the author's keywords, B is for the system-defined keywords in stage 1 (step 1), C is for the key words system-defined keywords in stage 1 (step 2), D is for system-defined keywords in stage 2 (step 1), E is for system-defined keywords in stage 2 (step 2). In the Table 6-7 the statistics of the analysis of the text of the articles in the formation of sets of keywords for the construction of appropriate histograms for groups A-E are presented.

Table 5. Descriptive statistics for keyword formation for the texts studied

	A	*B*	*C*	*D*	*E*
Average	4.8080808	5.5151515	6.5656566	7.5050505	8.4343434
Standard error	0.1808594	0.3103929	0.3903499	0.3012972	0.3246112
Median	4	5	6	7	8
Moda	4	5	5	7	8
Standard deviation	1.7995281	3.0883707	3.8839324	2.9978691	3.2298405
Sample dispersion	3.2383014	9.5380334	15.084931	8.9872191	10.43187
Kurtosis	0.6528151	1.7052728	0.7486433	-0.456455	-0.504378
Asymmetry	0.9479385	1.1253053	1.0657159	0.5375984	0.5170473
Interval	8	16	17	12	13
Minimum	2	1	1	2	3
Maximum	10	17	18	14	16
Sum	476	546	650	743	835
Account	99	99	99	99	99
The biggest(1)	10	17	18	14	16
The smallest(1)	2	1	1	2	3
Reliability level (95.0%)	0.3589095	0.6159647	0.7746366	0.5979144	0.6441803

Table 6. Statistics for histograms for group *A* and group *B*

A	Frequency	Integral %	*A*	Frequency	Integral %	*B*	Frequency	Integral %	*B*	Frequency	Integral %
1	0	0.00%	4	27	27.27%	1	2	2.02%	5	20	20.20%
2	4	4.04%	5	21	48.48%	2	10	12.12%	7	16	36.36%
3	20	24.24%	3	20	68.69%	3	12	24.24%	3	12	48.48%
4	27	51.52%	6	11	79.80%	4	4	28.28%	2	10	58.59%
5	21	72.73%	8	8	87.88%	5	20	48.48%	6	9	67.68%
6	11	83.84%	7	5	92.93%	6	9	57.58%	4	4	71.72%
7	5	88.89%	2	4	96.97%	7	16	73.74%	8	4	75.76%
8	8	96.97%	10	3	100.00%	8	4	77.78%	10	4	79.80%
9	0	96.97%	1	0	100.00%	9	2	79.80%	11	3	82.83%
10	3	100.00%	9	0	100.00%	10	4	83.84%	12	3	85.86%
11	0	100.00%	11	0	100.00%	11	3	86.87%	14	3	88.89%
12	0	100.00%	12	0	100.00%	12	3	89.90%	1	2	90.91%
13	0	100.00%	13	0	100.00%	13	2	91.92%	9	2	92.93%
14	0	100.00%	14	0	100.00%	14	3	94.95%	13	2	94.95%
15	0	100.00%	15	0	100.00%	15	1	95.96%	16	2	96.97%
16	0	100.00%	16	0	100.00%	16	2	97.98%	18	2	98.99%
17	0	100.00%	17	0	100.00%	17	0	97.98%	15	1	100.00%
18	0	100.00%	18	0	100.00%	18	2	100.00%	17	0	100.00%
Mo	0	100.00%	Mor	0	100.00%	Mo	0	100.00%	Mo	0	100.00%

The author of a scientific article usually chooses at his discretion the number of keywords in the range of 2 to 8 words (most often 3-5 keywords). The system also defines a different number of words, depending on the writing style of the particular author (there are articles in which the system does not find any keywords by Zipf law). For group B, most often the system determined the number of keywords 5, 7 and 3 (more than 10), although the distribution of found keywords ranged from 1 to 18 words (except 17). For Group B, the system most commonly identified keywords as 5, 7, and 3, although the distribution of keywords found ranged from 1 to 18 words (except 17), but the number of keywords found increased and the highest reliability was achieved. For Group C, the system most often determined the number of keywords 7, 6, 5, 10 and 8, although the distribution of found keywords ranged from 2 to

14 words (significantly narrowed the range). For Group D, the number of keywords 8, 5, 7, and 10 was most often determined by the system, although the distribution of the keywords found ranged from 3 to 16 words (improved accuracy).

Table 7. Statistics for histograms for group *C*, group *D* and group *E*

C	Frequency	Integral %	*C*	Frequency	Integral %
1	2	2,02%	5	20	20,20%
2	10	12,12%	7	16	36,36%
3	12	24,24%	3	12	48,48%
4	4	28,28%	2	10	58,59%
5	20	48,48%	6	9	67,68%
6	9	57,58%	4	4	71,72%
7	16	73,74%	8	4	75,76%
8	4	77,78%	10	4	79,80%
9	2	79,80%	11	3	82,83%
10	4	83,84%	12	3	85,86%
11	3	86,87%	14	3	88,89%
12	3	89,90%	1	2	90,91%
13	2	91,92%	9	2	92,93%
14	3	94,95%	13	2	94,95%
15	1	95,96%	16	2	96,97%
16	2	97,98%	18	2	98,99%
17	0	97,98%	15	1	100,00%
18	2	100,00%	17	0	100,00%
Mor	0	100,00%	Mor	0	100,00%

D	Frequency	Integral %	*D*	Frequency	Integral %
1	0	0.00%	7	15	15.15%
2	1	1.01%	6	14	29.29%
3	5	6.06%	5	13	42.42%
4	9	15.15%	10	12	54.55%
5	13	28.28%	8	11	65.66%
6	14	42.42%	4	9	74.75%
7	15	57.58%	12	6	80.81%
8	11	68.69%	3	5	85.86%
9	4	72.73%	14	5	90.91%
10	12	84.85%	9	4	94.95%
11	1	85.86%	13	3	97.98%
12	6	91.92%	2	1	98.99%
13	3	94.95%	11	1	100.00%
14	5	100.00%	1	0	100.00%
15	0	100.00%	15	0	100.00%
16	0	100.00%	16	0	100.00%
17	0	100.00%	17	0	100.00%
18	0	100.00%	18	0	100.00%
Mor	0	100.00%	Mor	0	100.00%

E	Frequency	Integral %	*E*	Frequency	Integral %
1	0	0.00%	8	14	14.14%
2	0	0.00%	5	12	26.26%
3	1	1.01%	7	11	37.37%
4	9	10.10%	10	11	48.48%
5	12	22.22%	4	9	57.58%
6	9	31.31%	6	9	66.67%
7	11	42.42%	9	9	75.76%
8	14	56.57%	11	5	80.81%
9	9	65.66%	14	5	85.86%
10	11	76.77%	12	4	89.90%
11	5	81.82%	13	4	93.94%
12	4	85.86%	15	3	96.97%
13	4	89.90%	16	2	98.99%
14	5	94.95%	3	1	100.00%
15	3	97.98%	1	0	100.00%
16	2	100.00%	2	0	100.00%
17	0	100.00%	17	0	100.00%
18	0	100.00%	18	0	100.00%
Mor	0	100.00%	Mor	0	100.00%

5 Conclusions

The article presents theoretical and experimental substantiation of the method of linguistic analysis of Ukrainian-language commercial content using Porter's stemming. The method is aimed at automatic detection of significant keywords of Ukrainian-language content on the basis of the proposed formalization of components of analysis is grammatical (graphemes), morphological, syntactic, semantic, referential and structural. To implement grammatical analysis, the rules of line recognition in the text are proposed, the set of standard graph models for 5 languages according to the nor-

mal Backus-Naur form is determined, and the corresponding grammar $G = <V, T, S, P>$ for identifying meaningful units of textual commercial content analysis. The morphological analysis was implemented by adapting Stemming M. Porter's algorithm to the Ukrainian language, in particular, a static end-tree was constructed and a weighted level of detail was selected - from 500 words with common endings, rules for truncation of endings and suffixes were substantiated. The basic requirements and procedures of syntactic, semantic, referential and structural analysis of Ukrainian-language commercial content are defined. An experimental study of the method of linguistic analysis was conducted on the materials of 100 scientific publications from two issues (783 and 805) of the Bulletin of the Lviv Polytechnic National University of the series "Information Systems and Networks" (http://science.lp.edu.ua/sisn). Based on the proposed method, the keyword search system demonstrated the ability to improve itself by forming and refining a number of common blocked words and a thematic dictionary with the participation of moderators. It is found that, for the technical scientific texts of the experimental base, the authors of articles usually define fewer keywords on average than are actually present in this work. Numerous statistics show that debugging your system's keywords nearly doubles the number of defined keywords, without compromising accuracy or reliability. Further experimental research will require testing the proposed method to identify keywords from other categories of texts is scientific humanities, artistic, nonfiction, etc.

References

1. Bobicev, V., Kanishcheva, O., Cherednichenko, O.: Sentiment Analysis in the Ukrainian and Russian News. In: First Ukraine Conference on Electrical and Computer Engineering, 1050-1055 (2017)
2. Sharonova, N., Doroshenko, A., Cherednichenko, O.: Issues of fact-based information analysis. In: CEUR Workshop Proceedings, 2136, 11-19 (2018)
3. Cherednichenko, O., Babkova, N., Kanishcheva, O.: Complex Term Identification for Ukrainian Medical Texts. In: CEUR Workshop Proceedings, 146-154 (2018)
4. Khomytska, I., Teslyuk, V., Holovatyy, A., Morushko, O.: Development of methods, models, and means for the author attribution of a text. In: Eastern-European Journal of Enterprise Technologies, 3(2-93), 41-46 (2018)
5. Khomytska, I., Teslyuk, V.: Authorship and Style Attribution by Statistical Methods of Style Differentiation on the Phonological Level. In: Advances in Intelligent Systems and Computing III. AISC 871, Springer, 105-118 (2019)
6. Babichev, S.: An Evaluation of the Information Technology of Gene Expression Profiles Processing Stability for Different Levels of Noise Components. In: Data, 3(4), 48 (2018)
7. Babichev, S., Durnyak, B., Pikh, I., Senkivskyy, V.: An Evaluation of the Objective Clustering Inductive Technology Effectiveness Implemented Using Density-Based and Agglomerative Hierarchical Clustering Algorithms. In: Advances in Intelligent Systems and Computing, 1020, 532-553 (2020)
8. Lytvyn, V., Vysotska ,V., Peleshchak, I., Basyuk, T., Kovalchuk, V., Kubinska, S., Chyrun, L., Rusyn, B., Pohreliuk, L., Salo, T.: Identifying Textual Content Based on Thematic Analysis of Similar Texts in Big Data. In: International Scientific and Technical Conference on Computer Science and Information Nechnologies (CSIT), 84-91 (2019)

9. Moseichuk, V.: Porter stemming algorithm for Ukrainian languages. http://www.marazm.org.ua/document/stemer_ua/
10. Vysotska, V., Lytvyn, V., Kovalchuk, V., Kubinska, S., Dilai, M., Rusyn, B., Pohreliuk, L., Chyrun, L., Chyrun, S., Brodyak, O.: Method of Similar Textual Content Selection Based on Thematic Information Retrieval. In: International Scientific and Technical Conference on Computer Science and Information Nechnologies (CSIT), 1-6 (2019)
11. Russian stemming algorithm. http://snowball.tartarus.org
12. Porter stemmer. https://github.com/allaud/porter-stemmer
13. The Porter Stemming Algorithm. http://tartarus.org/~martin/PorterStemmer/
14. Porter Stemming Algorithm. http://snowball.tartarus.org/algorithms/porter/stemmer.html
15. English stemming algorithm. http://snowball.tartarus.org/algorithms/english/stemmer.html
16. Porter, M. F.: An algorithm for suffix stripping. http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html
17. Willett, P.: The Porter stemming algorithm: then and now. http://eprints.whiterose.ac.uk/1434/
18. Senyk, M. The Porter Stemming Algorithm for Ukrainian. http://www.senyk.poltava.ua
19. Vysotska, V., Fernandes, V.B., Lytvyn, V., Emmerich, M., Hrendus, M.: Method for Determining Linguometric Coefficient Dynamics of Ukrainian Text Content Authorship. In: Advances in Intelligent Systems and Computing, 871, 132-151 (2019)
20. Lytvyn, V., Vysotska, V., Pukach, P., Nytrebych, Z., Demkiv, I., Senyk, A., Malanchuk, O., Sachenko, S., Kovalchuk, R., Huzyk, N.: Analysis of the developed quantitative method for automatic attribution of scientific and technical text content written in Ukrainian. In: Eastern-European Journal of Enterprise Technologies, 6(2-96), 19-31 (2018)
21. Vysotska, V., Lytvyn, V., Hrendus, M., Kubinska, S., Brodyak, O.: Method of textual information authorship analysis based on stylometry. In: 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies, 9-16 (2018)
22. Kulchytskyi, I.: Statistical Analysis of the Short Stories by Roman Ivanychuk. In: CEUR Workshop Proceedings, Vol-2362, 312-321 (2019)
23. Shandruk, U.: Quantitative Characteristics of Key Words in Texts of Scientific Genre (on the Material of the Ukrainian Scientific Journal). In: CEUR Workshop Proceedings, Vol-2362, 163-172 (2019)
24. Lovins, J.B.: Development of a stemming algorithm. In: Mechanical Translation and Computational Linguistics 11:22–31 (1968)
25. Jongejan, B., Dalianis, H.: Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. http://www.aclweb.org/anthology/P/P09/P09-1017.pdf
26. Vysotska, V., Kanishcheva, O., Hlavcheva, Y.: Authorship Identification of the Scientific Text in Ukrainian with Using the Lingvometry Methods. In: Computer Sciences and Information Technologies, CSIT, 34-38 (2018)
27. Vysotska, V., Burov, Y., Lytvyn, V., Demchuk, A.: Defining Author's Style for Plagiarism Detection in Academic Environment. In: Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP, 128-133 (2018)
28. Lytvyn, V., Vysotska, V., Burov, Y., Bobyk, I., Ohirko, O.: The linguometric approach for co-authoring author's style definition. In: Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS, 29-34 (2018)
29. Hardcoded stemmer for Ukrainian. https://github.com/vgrichina/ukrainian-stemmer
30. Perestoronin, P.: The Porter Stemming Algorithm for Russian. http://blog.eigene.in/post/49598738049/snowball