CONSTRUCTIVE-SYNTHESIZING MODELING OF NATURAL LANGUAGE TEXTS

Means for solving the problem of establishing the natural language texts authorship were developed. Theoretical tools consist of a constructors set was developed on the basis of structural and production modeling. These constructors are presented in this work. Some results of experimental studies based on this approach have been published in previous works by the author, the main results should be published in the next ones.

Constructors developed: converter of natural language text into tagged, tagged text into a formal stochastic grammar and the authors style similarity degree establishment of two natural language works based on the coincidence of the corresponding stochastic grammars (their substitution rules).

In this paper, constructors are developed and presented that model a natural language text in the form of a stochastic grammar that displays the structures of sentences in it. This approach allows you to highlight the syntactic features of the construction of phrases by the author, which is a characteristic of his speech. Working with a sentence as a unit of text for analyzing its construction will allow you to more accurately capture the author's style in terms of the words use, their sequences and speech style characteristic. It allows you not to be tied to specific parts of speech, but reveals the general logic of constructing phrases, which can be more informative in terms of the author's style characteristics for any text.

The presented work is a theoretical basis for solving the problems of the text authorship establishing and identifying borrowings. Experimental studies have also been carried out. The statistical similarity of solutions to the problems of establishing authorship and identifying borrowings was experimentally revealed, which will be presented in the next article of the authors.

The proposed approach makes it possible to highlight the semantic features of the author's phrases construction, which is a characteristic of his speech. Working with a sentence as a unit of text to analyze its construction will allow you to more accurately determine the author's style in terms of the use of words, their sequences and characteristic language constructions. Allows not to be attached to specific parts of speech, but reveals the general logic of building phrases.

It is planned to use the created model in the future to determine the authorship of natural language texts of various directions: fiction and technical literature.

Keywords: natural language texts, constructive-synthesizing modeling, establishing authorship, formal grammars, stochastic grammars, text models

Introduction

The work develops an approach to the construction formalization proposed by the author by the constructive modeling means. This approach allows you to highlight the semantic features of the author's phrases construction,
which is a characteristic of his speech. The use of the developed model is assumed in the field of determining the texts authorship and identifying borrowed ones.

This approach to text analysis is promising, due to the presence of each person’s own style and approach to constructing phrases. People communication by means of natural language texts is carried out with the use of not individual words, but expressed, sentences. A sentence as a sequence of individual words meanings is a new unit with a set of semantic values inherent only to it, among which there are also those that are not considered direct derivatives of the existing sentence composition, which is due to the peculiarity of its construction.

Modeling speech communication, in its entirety transmitted with the help of information language, is impossible without studying the features of the sentence structure. This aspect study of syntactic structures is important, in addition to purely linguistic tasks, for understanding the features and regularities of a person's mental activity.

Related works

The main text studies and methods of its formalization mostly work with words or their short sequences, and only a small part of the studies are based on the sentence as a unit of the text.

Widespread methods that investigate the test based on symbols or their sequence of some length [7] or separately words [8] and their sequences [6], work with lemmas [9] or parts of speech [10]. However, such an approach does not always adequately reflect the peculiarities of the text under study and the author's style.

The approach to the sentence as a text unit opens up new opportunities for work, since the peculiarities of its construction and the analysis of the words used within one sentence can serve as an additional source of information about the author [13]. Similar approaches were used for context research [11], working with the author’s mood research [12], in terms of a deeper understanding of the text [14]. Many studies also testify to the importance of considering the sentence as a single unit, rather than a collection of individual words [15], the importance of their order [16] and the general context [17].

Currently, there are many approaches to building models proposed [18, 19]. The most popular methods are the use of trees [20] and the construction of various neural networks [23, 24]. However, the most universal tool has not yet been found [22].

Such approaches are most widespread in the field of work with artificial intelligence in terms of text recognition and understanding, which largely confirms the necessity and relevance of using a separate sentence as a unit of text structure.

The processes of texts authorship identifying using the constructive-production modeling

Generalized designer

Development of a constructive-synthesizing approach to solving the problem of technical text authorship establishing. The generalized constructor is a triple called \( C \) [2]

\[
C = (\mathcal{M}, \Sigma, \Lambda)
\]

where \( \mathcal{M} \) – the non-homogeneous carrier of the structure, which expands during the construction process; \( \Sigma \) – is the operations and relations signature, that consisting of binding, substitution and derivation operations, operations on attributes, and a substitution relation; \( \Lambda \) – construction information support (CIS).

According to \( \Lambda \) [2] the form with attribute \( w \) is called a set of terminals and non-terminals that are united by binding operations. The developed constructors use a simple binding operation (and relationship) - concatenation. A form that contains only terminals is called a construction. Constructions are formed by derivation from the initial non-terminal, substitution operations and operations on attributes, and generalized partial and full derivation operations.

The operation of partial derivation (\( \Rightarrow \in \Sigma_p \)) consists of choosing a suitable substitution rule from their set, performing this substitution and performing operations on the attributes corresponding to the selected rule in a certain sequence.

The operation of complete derivation (or simply derivation, \( \Rightarrow \in \Sigma_p \)) consists of the sequential execution of the operation of partial derivation, starting from the initial nonterminal and ending with the construction.

To form structures, it is necessary to perform several clarifying transformations of constructors:

- specialization – defines the subject area: the semantic nature of the medium, the finite set of operations and their semantics, the operations attributes, the order of their execution and restrictions on substitution rules;
- interpretation – consists in connecting the operations of the signature with the execution algorithms of some algorithmic designer;
- concretization – expansion of axiomatics by production rules set, specific sets assignment of non-terminal and terminal symbols with their attributes and, if necessary, attribute values;
- implementation – formation of a structure from the constructor carrier elements by performing algorithms related to signature operations.
Constructor-converter of natural language text into a tagged text

The purpose of construction is to convert technical text into tagged text. For each word in a sentence, its attribute is determined in the composition: part of speech (pos), number (num) and gender (gen). Consider the specialization of the designer:

\[ C = (M, Σ, A) \rightarrow C_p = (M_p, Σ_p, A_p). \]

where \( M_p \) – a carrier that includes terminal and non-terminal alphabets, initial and tagged texts, as well as a set of production rules \( Ψ \), separate rules \( ψ_i: \langle s_i, g_i \rangle \), where \( i \) – rule number, \( s_i \) – is a sequence of substitution relations, \( g_i \) – is a sequence of operations on attributes, \( Σ_p \) – operations and connections for elements \( M_p \); CIS \( A_p \) \( \models \) \( A \).

The following provisions are included in CIS \( A_p \):

1. Signature \( Σ_p \) contains the signature of specific binding operations and attribute operations.
2. The terminals \( T \) include symbols and words of the Ukrainian language, denoted as \( * \) - letters that can be used to form words, \( ⊙ \) – space, \( ∣ \) – end-of-sentence symbols, \( \perp \) – end-of-text symbol, \( W_{i,j} \) - the \( j \)-th word in the \( i \)-th sentence. The first word of each sentence will additionally store information about its length \( l \), and the first word of the text will store the number of words in the longest sentence, \( max \), and the total number of sentences in the text, \( S \).
3. Non-terminals \( N = \{ σ, η, ε \} \) – auxiliary elements, where \( ε \) is the symbol 'empty'.

The following attribute operations are presented.

The operation \( Ω \) (word, ends, pos \( \downarrow \) word) \( \in Σ_p \) - determines the part of speech pos for the word 'word', which can take one of the following values: verb (v), noun (n), numeral (num), pronoun (pron), adjective (adj), conjunction (conj), adverb (adv), preposition (prep), verb adverb (v_adv), interjection and particle (frac), verb adjective (v_adj).

The operation \( ⊙ \) (word, ends, num \( \downarrow \) word) \( \in Σ_p \) – determines the number num for the word 'word', which can be singular (sing) or plural (plur).

The operation \( ⊕ \) (word, ends, gen \( \downarrow \) word) \( \in Σ_p \) – determines the gender (gen) for the word 'word', which can take one of the following values: feminine (f), masculine (m), and neuter (n).

Each of these operations compares the word with all elements of 'ends' - terminals \( A \), \( Ω \), \( ⊙ \), \( ⊕ \).

If there is a match with a specific ending, a result is formed, and the parameters pos, num, and gen are assigned corresponding values.

The operation \( = (a, b) \in Σ_p \) – assigns the value \( b \) to the variable \( a \).

The operation \( + (c, a, b) \in Σ_p \) – adds \( c = a + b \).

The Comparison operation \( × ((a, b, c, d) \in Σ_p \) – compares a with \( b \). If \( a \) is greater, then the nested operation \( c \) is executed; if it is smaller, then \( d \) is executed.

Interpretation of the designer

Let's form a constructive system from the SR designer, as an elemental design base, and the SA algorithmic designer, as a model-executor of the design.

\[ (C_p = (M_p, Σ_p, A_p), C_A = (M_A, V_A, Σ_A, A_A)) \rightarrow (C_{PAI} = M_p, Σ_p, A_p). \]

where \( V_A = \{ A_1 \mid \ V_1, \ldots, A_l \mid V_l \} \) – set of forming algorithms in the basic algorithmic structure, \( X \); and \( V_1 \) – possible input and output data of the algorithm \( A \mid V \), \( M \cup A \mid V \) \( \cup X( \bigcup A_i \mid V_i \cup Y(A_i)) \) – carrier of the algorithmic structure, \( Σ_A \) – a set of algorithm binding operations, \( Λ_A \) – CIS, \( Ω(\bigcup C_A) \) – set of algorithms constructed in \( C_A \), \( M_p \) = \( M \cup \bigcup A_i \mid V_i \). The Comparison operation \( ⊕ \) (word, ends, gen \( \downarrow \) word) \( \in Σ_p \) – determines the gender (gen) for the word 'word', which can take one of the following values: feminine (f), masculine (m), and neuter (n).

Each of these operations compares the word with all elements of 'ends' - terminals \( A \), \( Ω \), \( ⊙ \), \( ⊕ \).

If there is a match with a specific ending, a result is formed, and the parameters pos, num, and gen are assigned corresponding values.

The operation \( = (a, b) \in Σ_p \) – assigns the value \( b \) to the variable \( a \).

The operation \( + (c, a, b) \in Σ_p \) – adds \( c = a + b \).

The Comparison operation \( × ((a, b, c, d) \in Σ_p \) – compares a with \( b \). If \( a \) is greater, then the nested operation \( c \) is executed; if it is smaller, then \( d \) is executed.

Structure \( C_{PAI} \) contains algorithms for performing operations:

- \( A_0 \) is an algorithms composition, \( A_1 \) \( \cdot A_2 \) – sequential execution of algorithm \( A_1 \) after \( A_2 \);
- \( A_3 \) is a substitution, where \( l \) – current form, \( s_i \) – the rule to be executed;
- \( A_4 \) is a partial output, where \( Ψ \) – the set of production rules to be executed;
- \( A_5 \) is a complete derivation, where \( σ \) – is an axiom, \( Ψ \) – is a set of production rules, \( Ω \) – is a set of formed constructions;
- \( A_6 \) is a definition for the word of its language part pos;
- \( A_7 \) is a definition for the word of its number num;
- \( A_8 \) is a definition for the word of its gender gen;
- \( A_9 \) is an assignment of the value \( b \) to variable \( a \);
- \( A_{10} \) is adding \( c = a + b \);
- \( A_{11} \) is execution of action \( c \) or \( d \) based on the result of comparing \( a \) and \( b \).

When specifying \( C_{PAI} \) the following is parameterized:

\[ C_{PAI} \rightarrow C_{PAI}(TT) = (M_{TT}, Σ_{TT}, A_{TT}). \]
where \(A_K \rightarrow \Lambda_1 \cup \{M_K = T_T \cup N\} \cup \Lambda_1\). TT is a technical text submitted for analysis.
In the substitution rules \(\psi_i: \langle s_i, g_i, i\rangle\) the sequence of substitution relations \(s_i\) consists of the relation \(s_{i,1}\) – analysis of the TT, \(s_{i,2}\) – formation of a words’ set \(W_{i,j}\) with their attributes. Operations \(g_{i,j}\) are performed after execution of \(s_{i,1}\) and before \(s_{i,2}\).

**Initial construction conditions:** \(\sigma\) – a non-terminal from which the derivation begins and the initial values \(\text{max} = 1, i = 1\) and \(j = 1\).

**Construction completion condition:**

- all incoming text is tagged.

The designer has the following specialization:

- Construction completion condition:

  - the parsing of the text and the formation of the first element in the tagged text \(W_{i,j}\) begins
  \(s_1 = (\sigma \rightarrow \eta, W_{i,j} \rightarrow \varepsilon)\).

  - Parsing occurs from one character to the next with its rewriting in \(W_{i,j}\) for further tagging
  \(s_2 = (\eta \rightarrow \ast \eta, W_{i,j} \rightarrow \ast W_{i,j})\).

  - When a space or end-of-sentence mark is reached, a tagging determination is made for the word and the next word is passed
  \(s_3 = (\eta \rightarrow \ast \eta, \varepsilon)\).

  - The operations \(\ominus, \ominus\), and \(\otimes\) in attribute operations determine the part of speech, number, and gender of a word, respectively. The transition to the next word in the sentence occurs. The flag "done" for each word is set to position 0, and it will be used later for rule formation
  \(g_3 = \langle \ominus \langle W_{i,j}, \text{pos} \downarrow W_{i,j} \rangle, \ominus \langle W_{i,j}, \text{num} \downarrow W_{i,j} \rangle, \ominus \langle W_{i,j}, \text{gen} \downarrow W_{i,j} \rangle = (\text{done} \downarrow W_{i,j}, 0), + (j, 1, j) \rangle\).

  - The rule \(s_3\) is applied when the end of the sentence is reached, and like the previous rule, the tagging for the next word is determined and the next sentence is passed. A transition to the next word is performed. The length of each sentence is calculated and set and stored as an attribute of its first word. The maximum sentence length is determined as an attribute of the very first word in the text. Along with this, to mark the end of the sentence, \(\downarrow (W_{i,j} = \downarrow)\) will be written to its final position. This is necessary for the correct operation of the following constructor
  \(s_4 = (\eta \rightarrow _\downarrow \sigma, \varepsilon)\).

  - \(g_4 = \langle \ominus \langle W_{i,j}, \text{pos} \downarrow W_{i,j} \rangle, \ominus \langle W_{i,j}, \text{num} \downarrow W_{i,j} \rangle, \ominus \langle W_{i,j}, \text{gen} \downarrow W_{i,j} \rangle = (\text{down} \downarrow W_{i,j}, 0), + (j, 1, j) = (W_{i,j}, \downarrow = (j, 1), + (i, 1, j) \rangle\).

  - The last rule is used when the end of the text is reached and is final. The \(\text{am}\) attribute of the first word stores the total number of sentences in the text
  \(s_5 = (\eta \rightarrow \downarrow, \varepsilon)\).

  - \(g_5 = (\text{am} \downarrow W_{i,1}, i)\).

**Realization**

The constructor implementation is the language constructions formation from its carrier elements through the algorithm’s execution related to signature operations according to the rules of substitution:

\[\mathcal{C}_{\text{PAIK}} \rightarrow \mathcal{O}(\mathcal{C}_{\text{PAIK}}(\mathcal{O}_{\text{PAIK}}(\mathcal{T}_{\text{PAIK}}))\).

where \(\mathcal{O}_{\text{PAIK}}(\mathcal{T}_{\text{PAIK}}) = \mathcal{O}(\mathcal{C}_{\text{PAIK}}(\mathcal{T}_{\text{PAIK}}))\). \(\mathcal{O}\) – all possible outcomes of the constructor, however, since the generated constructor is based on a specific text, the resulting processed text \(\mathcal{O}\) will be the only possible outcome.

As a result of the constructor implementation, the processed text with tagged words as \(\mathcal{O}(\mathcal{C}_{\text{PAIK}}(\mathcal{T}_{\text{PAIK}}))\) was received.

For example, let’s take the sentence «Чорні ґрати розпанахли небо. Червонорожеве воно тянуло, манілио».

The result of the designer's work will look like this:

\(W_{1,1} = \text{adj, plur, - Чорні}; W_{1,2} = \text{n.plur, - ґрати}; W_{1,3} = \text{v.plur, - розпанахал}; W_{1,4} = \text{n.sing, небо}; W_{2,1} = \text{adj, sing,n Чорвонорожеве; W_{2,2} = pr,sing,n воно; W_{2,3} = v,sing,n тянуло; W_{2,4} = v,sing,n манілио}\)

**Tagged text constructor-converter into formal substitution rules set with a probability measure**

The purpose of construction is to build a stochastic constructor rule that formalizes the syntactic component of the technical text.

The initial construction condition is the implementation of the \(\mathcal{C}_{\text{TR}}\) constructor – the tagged text \(T_g\) obtained as a result of the constructor implementation \(\mathcal{C}_{\text{PAIK}} \rightarrow \mathcal{O}(\mathcal{C}_{\text{TR}}(\mathcal{T}_{\text{TR}}))\).

**Construction completion condition:** each sentence of the tagged text is converted into a corresponding set of rules \(\mathcal{O}(\mathcal{C}_{\text{TR}}(\mathcal{T}_{\text{TR}})))\), which happens under the condition \(r_5 = \text{true}\), which is set when the last word of the longest sentence in the text is reached. This will serve as an indication that all other words in the text have already been processed and the rule building process is complete.

The designer has the following specialization:

\[\mathcal{C} = (M, \Sigma, \Lambda) \rightarrow \mathcal{C}_{\text{TR}}(T_g) = (M_T, \Sigma_T, \Lambda_T),\]
where \( M_T \) – is a carrier that includes tagged text \( Tg \), \( \Sigma_T \) – operations and relationships on elements \( M_T \) and axiomatics \( A_T \).

The operation \( * (r, a, b) \) – checking that the attributes \( \text{pos}_a, \text{num}_a, \text{gen}_a \) of element match the attributes \( \text{pos}_b, \text{num}_b, \text{gen}_b \) of element \( b \), where \( a \) and \( b \) are tagged words. If there is a complete match, the result is 1, otherwise - 0.

The operation \( \& (y, x_1, x_2) \) – is a logical and with an unlimited number of operands \( y = x_1 \) and \( x_2 \) and …;

A loop operation \( + (a, c) \) – where \( a \) is a condition, \( c \) – is an operation performed while the condition is valid;

The operation \( - (c, a, b) \) – is equal to \( c = a - b \) in infixed form;

The operation \( : (c, a, b) \) – is equal to \( c = a : b \) in infixed form, division of real numbers;

The operation \( \leq (r, a, b) \) – is a comparing \( a \leq b \) with saving the result in \( r \).

Interpreting the \( C_T \) constructor using the same algorithmic constructor \( C_A \):

\[
(C_T, C_A) \mapsto (C_T = M_{T'}, \Sigma_{T'}, A_{T'}). \]

The resulting constructor will generate a language characteristic of the processed text and structurally similar texts of a certain author.

Thus, each sentence of the presented text will be presented in the form of a chain of rules that will reflect the sequence of used parts of speech and the probability of their appearance in the presented sequence.

Initial conditions: the initial form \( W_{1,1} \) – is the first word in the text, where \( i = 1, m = 2 \) word numbers in them, \( t = 0 \) – the number of matches with the selected pair of parameters in a sequence of two words, \( k = 1 \) – the number of the rule being built, \( f = \text{true} \), \( t2 = \text{false} \) – are conditions for the execution of the rules: if true, it is available for use, if false - not. The probability of obtaining the entire sentence is defined as its speech parts sequences probabilities product.

Interpreting the \( C_T \) constructor using the same algorithmic constructor \( C_A \):

\[
(C_T, C_A) \mapsto (C_T = M_{T'}, \Sigma_{T'}, A_{T'}). \]

The resulting constructor will generate a language characteristic of the processed text and structurally similar texts of a certain author.

Thus, each sentence of the presented text will be presented in the form of a chain of rules that will reflect the sequence of used parts of speech and the probability of their appearance in the presented sequence.
Under this condition, the calculation of the probability of the selected sequence appearing in the text is used, the rules $s_1 - s_3$ become unreachable, and the rules $s_4 - s_7$ become available for processing

$$g_3 = \langle (n, n, 1), : (\text{prob}, t, n), = (r_1, f\ false), = (r_2, \text{true}), = (n, 1) \rangle.$$  

The following rules are responsible for forming rules for the first words of each of the sentences when all corresponding words are repeated

$$s_4 = (W_{i,j} \mapsto W_{n,j}, \omega_k),$$  

$$g_{a,4} = (\langle 0, \text{done} \downarrow W_{i,j}, x1 \rangle, \ast (W_{i,j}, W_{n,j}, x2)).$$

If the attributes of the words match, a new rule $\omega_k$ is built, $\sigma$ and $W_{i,j} \alpha_{i,j}$ are written in its left and right parts, respectively, where $\alpha_{i,j}$ is the non-terminal of the newly formed rules, and the probability of its activation for the text $\text{prob}$ is written

$$g_{a,2} = (\langle \& (y, x1, x2), \rangle <\langle (y, 0), (L \downarrow \omega_{i,j}, \sigma), = (R \downarrow \omega_{i,j}, W_{i,j}), = (\text{prob} \downarrow \omega_{i,j}, \text{prob}), \varepsilon) \rangle).$$

After the formation of the rule, the flag of the presence of at least one rule on this layer is set $u = \text{true}$, the rule receives the uniqueness index in the $\text{idone}$ layer, and the constructor moves to the next sentence

$$g_{a,4} = (\langle (n, 1, n), = (\text{done} \downarrow W_{i,j}, \text{idone}), = (u, \text{true}) \rangle).$$

The next rule is similar to rule $s_2$, does not make calculations and is responsible for increasing the sentence number $n$ to move further through the layer

$$s_5 = (W_{i,j} \mapsto W_{n,j}, \downarrow, e),$$  

$$g_5 = (\langle (n, 1, n) \rangle).$$

If there is only one word in the sentence, rule $s_6$ is used

$$s_6 = (W_{i,j} \downarrow \mapsto W_{n,j}, \omega_k).$$

To form a rule in this case, the following checks will be carried out: whether the word is included in another done rule, whether the corresponding attributes of the words match

$$g_{a,3} = (\langle 0, \text{done} \downarrow W_{i,j}, \text{x1} \rangle, \ast (W_{i,j}, W_{n,j}, x2)).$$

If the sentence is not the first, we form the corresponding rule $\omega_k$. In its left part, write $\sigma$, respectively, and only $W_{l,m}$ in the right part

$$g_{a,3} = (\langle \& (x, x1, x2, x3), \rangle <\langle (y, 0), (L \downarrow \omega_{i,j}, \sigma), = (R \downarrow \omega_{i,j}, W_{i,j}), = (\text{prob} \downarrow \omega_{i,j}, \text{prob}), \varepsilon) \rangle).$$

And then the same as in $g_{a,4}$ - its creation flag $u = \text{true}$, is set, the rule receives the uniqueness index in the $\text{idone}$ layer and the executor moves to the next sentence

$$g_{a,4} = (\langle (n, 1, n), = (\text{done} \downarrow W_{i,j}, \text{idone}), = (u, \text{true}) \rangle).$$

And when the end is reached, rule $s_7$ is triggered

$$s_7 = (W_{i,j} \mapsto \downarrow \text{true}, e).$$

Reaching the end of a layer means the end of rule formation and transition to the formation of another. For this, the flags $r2 = \text{false}$ and $r1 = \text{true}$ are changed, which will close the rules $s_4 - s_7$ and open the rules $s_1 - s_3$ to search for other matches and count them. To reflect the operation of another rule in the layer, the rule’s uniqueness number for the $\text{idone}$ layer is increased

$$g_{a,1} = (\langle = (u, \text{true}, y), \rangle <\langle (y, 0), = (r_2, f\ false), = (r_1, \text{true}), + (\text{idone}, 1, \text{done}), = (t, 0), \rangle), e).$$

If the work with the layer is completed and rules have been formed for all the words in it, the constructor moves to the next layer, starting again from the first sentence $i=1$ to search for a match. The calculation of the uniqueness of the rules in the layer also starts from the beginning of $\text{idone}=1$. If the final layer is reached (the last word in the longest sentence $W_{l,\text{max}}$ is processed), the work of the performer with the first layer will be completed $r3 = \text{true}$, $r1 = \text{false}$, $r2 = \text{false}$

$$g_{a,2} = (\langle u, f\ false, y \rangle, \langle <\langle (y, 0), = (i, 1), = (\text{idone}, 1), = (t, 0), = (r_3, \text{true}), = (r_2, f\ false), = (r_1, f\ false), \rangle, e).$$

To continue forming rules from tagged text, operating the consecutive pairs of words in each sentence. The transition from word to word does not occur along the sentence, but according to the number of words in them. In this way, the constructor considers a pair of consecutive words in a sentence

$$s_8 = (W_{i,j} W_{l,m} \mapsto W_{n,j} W_{n,m}, e).$$

To consider an existing pair of words as similar, you need to check the following parameters: the words have not yet been processed; the attributes of the selected sequence of two words (part of speech, gender and number) match the numbered words in the next sentence, the previous string of words must also match, which is checked by $\text{jdone}$

$$g_{a,1} = (\langle 0, \text{done} \downarrow W_{i,j}, x1 \rangle, \ast (W_{i,j}, W_{n,p}, x2) \ast (W_{l,m}, W_{n,m}, x3), \langle - (k, j, 1), \langle <\langle (j, 1), = (\text{done} \downarrow \text{jdone}, x4), x4 = \text{true}\rangle).$$

If the attributes match, the pair is counted in the total number of similar sequences and the value of $t$ and the value of $n$ are increased to move to the next sentence in the layer

$$g_{a,2} = (\langle y, x1, x2, x3, x4 \rangle, <\langle (y, 0), + (t, 1, t), e \rangle, + (n, 1, n)).$$
The following rule is triggered when the end of the sentence is reached, the sentence number $n$ is incremented for further viewing of the words in the layer

$$s_9 = (W_{i,j} W_{i,m} r_3 \rightarrow W_{n,j} \perp, \varepsilon),$$

$$g_9 = (+ (n, 1, n)).$$

The following rule is executed when it is impossible to move further along the sentences due to reaching the end of the layer. Under this condition, the probability of the appearance of the selected sequence in the text is calculated, rules $s_8$ and $s_{10}$ become unreachable, and rules $s_{11}$ and $s_{14}$ become available for processing

$$s_{10} = (W_{i,j} r_3 \rightarrow \perp, \varepsilon),$$

$$g_{10} = (- (n, 1, 1), (\text{prob}, t, n), = (r_3, \text{false}), = (r_4, \text{true}), = (i, 1), = (n, 1)).$$

The next step is to revisit the current layer and create rules $\omega_k$

$$s_{11} = (W_{i,j} W_{i,m} r_4 \rightarrow W_{n,j} W_{n,m}, \omega_k).$$

To form the appropriate rule, the check from the first rule is repeated and we additionally check whether the word is the first in the sentence (x4) for the correct formation of the initial rules

$$g_{11,1} = (== (0, \text{done} \ \downarrow W_{i,j} x1), \star (W_{i,j} W_{i,j} x2), \star (W_{i,m} W_{n,m} x3), -(k, 1), <\rightarrow (j, 1, == \ (\text{done} \ \downarrow W_{i,j} \ \downarrow j \ \downarrow \text{done}, x4), x4 = \text{true})).$$

If everything matches and the word is not the first in the sentence, a new rule $\omega_k$ is built. $\alpha_{i,j}$ is written in the left part of the rule, $W_{i,m} \alpha_{i,m}$ is written in its right part, where $\alpha_{i,j}$ is the non-terminal of the newly formed rules, and the probability of its activation for the text $\text{prob}$ is written.

$$g_{11,2} = \ (\& (y, x1, x2, x3, x4),$$

$$\rightarrow (y, 0, \ (L \ \downarrow \ \omega_{i,j} \ \alpha_{i,j}), = (R \ \downarrow \ \omega_{i,j} W_{i,m} \alpha_{i,m}), = (\text{prob} \ \downarrow \ \omega_{i,j} \ \text{prob}), \varepsilon)).$$

After creating a rule, its creation flag $u = \text{true}$ is set, the rule receives a unique index in the idone layer, and the constructor moves to the next sentence

$$g_{11,3} = (+ (n, 1, n), = (\text{done} \ \downarrow W_{i,j} \ \downarrow \text{done}), = (u, \text{true})).$$

The next rule is similar to rule $s_9$, does not make calculations and is responsible for increasing the sentence number $n$ to move further through the layer

$$s_{12} = (W_{i,j} W_{i,m} r_4 \rightarrow W_{n,j} \perp, \varepsilon),$$

$$g_{12} = (+ (n, 1, n)).$$

If the last word in the sentence is in the layer, rule $s_{13}$ is triggered

$$s_{13} = (W_{i,j} \perp \rightarrow W_{n,j}, \omega_k).$$

To form a rule in this case, the following checks will be carried out: whether the word is included in another done rule, whether the corresponding attributes of the words match, whether the word is the first in the sentence and, if not, whether the previous chain matches

$$g_{13,1} = (== (0, \text{done} \ \downarrow W_{i,j} x1), \star (W_{i,j} W_{i,j} x2), \rightarrow (j, 1, == \ (\text{done} \ \downarrow W_{i,j} \ \downarrow j \ \downarrow \text{done}, x3), x3 = \text{true})).$$

If the sentence is not the first, we form the corresponding rule $\omega_k$. $\alpha_{i,j}$ is written in the left part of the rule, $W_{i,m} \alpha_{i,m}$ is written in its right part, where $\alpha_{i,j}$ is the non-terminal of the newly formed rules, and the probability of its activation for the text $\text{prob}$ is written.

$$g_{13,2} = \ (\& (y, x1, x2, x3),$$

$$\rightarrow (y, 0, \ (L \ \downarrow \ \omega_{i,j} \ \alpha_{i,j}), = (R \ \downarrow \ \omega_{i,j} W_{i,j}), = (\text{prob} \ \downarrow \ \omega_{i,j} \ \text{prob}), \varepsilon)).$$

Next, just like in $g_{11,1}$ – its creation flag $u = \text{true}$ is set, the rule receives the uniqueness index in the idone layer and the constructor moves to the next sentence

$$g_{13,3} = (+ (n, 1, n), = (\text{done} \ \downarrow W_{i,j} \ \downarrow \text{done}), = (u, \text{true})).$$

When the end is reached, rule $s_{14}$ is triggered

$$s_{14} = (W_{i,j} r_4 \rightarrow \perp, \varepsilon).$$

Reaching the end of a layer means the end of rule formation and transition to the formation of another. For this, the flags $r_4 = \text{false}$ and $r_3 = \text{true}$ are changed, which will close the rules $s_{11} - s_{14}$ and open the rules $s_9 - s_{10}$ to search for other matches and count them. To reflect the operation of another rule in the layer, the rule's uniqueness number for the idone layer is increased

$$g_{14,1} = (= (r_4, \text{false}), = (r_3, \text{true}), +(\text{done}, 1, \text{done}), = (t, 0), \leq (r, j, l \ \downarrow W_{i,j})).$$

The procedure of counting coincidences and calculating the probability of their occurrence for building rules on its basis continues until all words in the layer have been processed. To work with all chains, at each layer pass, the $j\text{done}$ uniqueness index is increased to check the calculation condition

$$g_{14,2} = (\rightarrow (a \ \downarrow W_{i,j}, i, \rightarrow (r, (= (i, 1, 1), == (a, \text{true}, 1), \rightarrow (1, 0, +(\text{done}, 1, \text{done})))).$$

If the work with the layer is completed and rules have been formed for all the words in it, the constructor moves to the next layer by increasing $j$ and starting again from the first sentence $i = 1$ to look for a match. The calculation of the uniqueness of rules in a layer also starts from the beginning of $\text{done} = 1$ and $j\text{done} = 1$. On the condition that the final layer is reached (the last word in the longest sentence $W_{i,\text{max}}$ is processed), the work of the constructor will be completed $r5 = \text{true}$.
\[ \theta_{1,4,3} = \langle (\max_j \ W_{1,1,j} \cdot (+j, 1, j)), = (i, 1), = (\text{done}, 1), = (\text{done}, 1) \rangle; (= (r5, \text{true}), (= (r3, \text{true}))) \].

As a result of the work of the constructor-converter with \( \Omega(C_{\text{PAIK}}(TT)) \), we get a set of rules that reflects the style of the author's language in the corresponding text \( \Omega(C_T(R)) \).

The implementation of the structure is the formation of language constructions from the elements of its carrier through the execution of algorithms related to signature operations according to the rules of axionomics:

\[ C_{\text{PK}} \rightarrow \tilde{\Omega}(C_{\text{PK}}). \]

where \( \tilde{\Omega}(C_{\text{PK}}) \subset \Omega(C_{\text{PK}}) \).

For example, let's take sentences that have the form:

«Ми були дуже схожі.
Я любила читати книжки.
А ти захоплювався виставами.
Але..»

Між нами було й багато різниці».

The tagged text for this example:

\[ W_{1,1} = \text{pron.plur} \ \text{Ми} \ W_{1,2} = \text{v.plur} \ \text{були} \ W_{1,3} = \text{adv,sing} \ \text{дуже} \ W_{1,4} = \text{adj.plur} \ \text{схожі} \]

\[ W_{2,1} = \text{pron.sing} \ \text{Я} \ W_{2,2} = \text{v.sing} \ \text{любила} \ W_{2,3} = \text{v.sing} \ \text{читати} \ W_{2,4} = \text{n.plur} \ \text{книжки} \]

\[ W_{3,1} = \text{conj} \ A \ W_{3,2} = \text{pron.sing} \ \text{ти} \ W_{3,3} = \text{v.sing} \ \text{захоплювався} \ W_{3,4} = \text{n.plur} \ \text{виставами} \]

\[ W_{4,1} = \text{pron.plur} \]

\[ W_{5,1} = \text{conj} \]

\[ W_{5,2} = \text{pron.plur} \ \text{нами} \ W_{5,3} = \text{v.sing} \ \text{було} \ W_{5,4} = \text{conj} \ \text{й} \ W_{5,5} = \text{adv,sing} \ \text{багато} \ W_{5,6} = \text{adj.plur} \ \text{різного}. \]

The result of the designer's work will be presented in the form of relevant rules:

\[ \sigma \rightarrow W_{1,1}a_{1,1}; a_{1,1} \rightarrow W_{1,2}a_{1,2}; a_{1,2} \rightarrow W_{1,3}a_{1,3}; a_{1,3} \rightarrow W_{1,4}; \]

\[ \sigma \rightarrow W_{2,1}a_{2,1}; a_{2,1} \rightarrow W_{2,2}a_{2,2}; a_{2,2} \rightarrow W_{2,3}a_{2,3}; a_{2,3} \rightarrow W_{2,4}; \]

\[ \sigma \rightarrow W_{3,1}a_{3,1}; a_{3,1} \rightarrow W_{3,2}a_{3,2}; a_{3,2} \rightarrow W_{3,3}a_{3,3}; a_{3,3} \rightarrow W_{3,4}; \]

\[ \sigma \rightarrow W_{4,1}; \]

\[ \sigma \rightarrow W_{5,1}a_{5,1}; a_{5,1} \rightarrow W_{5,2}a_{5,2}; a_{5,2} \rightarrow W_{5,3}a_{5,3}; a_{5,3} \rightarrow W_{5,4}a_{5,4}; a_{5,4} \rightarrow W_{5,5}a_{5,5}; a_{5,5} \rightarrow W_{5,6}. \]

The purpose of construction is to establish the degree of similarity of the two texts according to the syntactic style of the author's language, a comparison of the text models is carried out with the help of a constructor-meter.

The initial conditions for constructing a model of two texts in the form of a set of substitution rules with the probability of its activation \( \Omega(C_T(R)) \) and \( \Omega(C_T(R)) \), which represent the text of certain technical works \( \Omega(C_{\text{PAIK}}(TT)) \) and \( \Omega(C_{\text{PAIK}}(TT)) \), is the result of the execution of previous constructors.

Construction completion condition: \( r3 = \text{true} \), getting a number from 0 to 1 that reflects the similarity of two works after comparing all rules in two text models.

The designer has the following specialization:

\[ C = (M, \Sigma, A) \rightarrow C_T = (M_T, \Sigma_T, A_T). \]

where \( M_T \) is a medium that includes a set of rules describing the language of the author in a certain text \( R_t \), \( \Sigma_T \) are operations and relations on the elements \( M_T \) and CIS \( A_T \).

We interpret the structure \( C_T \) using the algorithmic structure \( C_A \):

\[ (C_T, C_A) \rightarrow (C_T = M_T, \Sigma_T, A_T). \]

where \( V_A = \{ A_i^{(1)} \}_{i=1} \) – is the set of forming algorithms of the basic algorithmic structure, \( X \) and \( Y_1 \) – are the set of definitions and values of the algorithm \( A_i^{(1)} \), \( M_A = \bigcup_{A_{(i)}^{(1)} \in V_A} (X(A_{(i)}^{(1)}) \cup Y(A_{(i)}^{(1)})) \) – he carrier of the algorithmic structure, \( \Sigma_1 \) – the set of operations linking algorithms, \( \Lambda_1 \) – the axiomatics of the algorithmic structure, \( \Omega(C_A) \) – a set of algorithms constructed in \( C_A \).

Next, the operation on attributes is presented.

The operation \( \min(m, a, b) \) compares the numbers \( a \) and \( b \), and stores the smallest in \( m \); The operation \( (c, a, b) \) is subtraction \( c = a - b \); The operation \( *(c, a, b) \) is multiplication \( c = a \cdot b \);
perform operations on the attributes

The second rule sequentially traverses all strings longer than one rule, advancing along their length for both

If all conditions are met, the product of the difference in their probabilities is calculated, and the result is

The first rule starts by comparing the rules of two constructors Ω(Cr(R1)) and Ω(Cr(R2)) describing two
texts that are examined for their similarity, i = 1, j = 1.

If the same rules or rules exist, the degree of their statistical structural similarity will be determined as the
product of the minimum difference in the probabilities of applying the corresponding rule

where $\hat{\varphi}_i$ – i-th sentence in T1 text and $\hat{\varphi}_j$ – j-th sentence in T2 text.

The degree of statistical structural similarity of T1 and T2 texts:

Initial conditions: rule $\rightarrow 1, i = m = j = n = 1$, where i and m are numbers of chains (sentences) in the text, j and n – are numbers of rules in chains. $max \downarrow \omega_{i,1}$ where $max = 0$ is the product of the difference in probabilities. $max_{ch} \downarrow \omega_{i,1}, max_{ch} = 0$ is the maximum length of the chain, res = 0 – is the total similarity of two texts. $k = n + 1, h = j + 1$, these are the next rules in the chain concerning j and n, respectively. And the flags for triggering s1 and s2 τ1 = true, τ2 = false, as well as the flag for completing the comparison τ3 = false.

The first rule is used to compare the first rules in all strings of text

For each rule, if their right parts match and the length of the chain is only one rule (that is, the sentence
consists of only one word)

If all conditions are met, the product of the difference in their probabilities is calculated, and the result is
stored in the first element of the chain. And until the end of the second text is reached, the products are added up in
res. If the chains from the first text end, the first rule is closed and the second is opened

The second rule sequentially traverses all strings longer than one rule, advancing along their length for both
texts under investigation. All rules of the second text are reviewed (m varies from 1 to the end of the text). For each sentence, a sequential review of all rules is performed

To start work and calculate the similarities, the right parts of the first rules in both texts are compared and we
perform operations on the attributes

If all conditions are met, the first rule in the chain is processed: the length of the chain that matches ch is
calculated, the product of the difference in the probabilities of the rules from both sim texts is found, and the maximum
length of the matching chain and the result of calculating their coincidence are stored in the first element of the chain
\[ g_{2,2} = \langle(y, x_1, x_2, x_3, x_4), \langle y, 0, \left(\langle ch \downarrow \vartheta_{1,1}, 1, ch \downarrow \vartheta_{1,1}\rangle, \left(\text{rule, min}(\langle r, \text{prob} \downarrow \vartheta_{i,j}, \text{prob} \downarrow \vartheta_{m,n}\rangle)\right)\right) \rangle, (\text{sim} \downarrow \vartheta_{1,1}, \text{rule}) \rangle, (\langle ch \downarrow \vartheta_{i,j}, \text{max} \text{ch} \downarrow \vartheta_{i,j}, \left(\left(\text{max} \downarrow \vartheta_{i,j}, \text{sim} \downarrow \vartheta_{i,j}\right)\right), +\langle j, 1, j\rangle, +\langle n_1, n_1\rangle), (\langle (j, 1), (n, 1), (m, 1, m)\rangle, \varepsilon) \rangle. \]

Then all subsequent chains and their rules are processed under the same conditions

\[ g_{2,3} = \langle (R \downarrow \vartheta_{i,h}, R \downarrow \vartheta_{m,k}, x_1), \langle l \downarrow W_{1,1} \downarrow \vartheta_{i,h}, h, = \langle x_2, \text{true}\rangle, = (x_2, \text{false}) \rangle, \langle l \downarrow W_{m_1} \downarrow \vartheta_{m,n}, k, = \langle x_3, \text{true}\rangle, = (x_3, \text{false}), \langle y, x_1, x_2, x_3 \rangle \rangle. \]

If the chain of coincidences is broken, the comparison of the rules of the 2nd text begins already for the next chain of rules of the first text. If the chain has ended, the transition to the next one is performed, and each of the rules in both texts is similarly checked for coincidence. If the rules in the text end, we close the possibility of executing the second rule \( t_2 = \text{false} \) and end the calculations using the flag \( t_3 = \text{true} \). Consequently: \( g_{2,4} = \langle \langle y, 0, \left(\langle ch \downarrow \vartheta_{i,v}, 1, ch \downarrow \vartheta_{i,j}\rangle, \left(\text{rule, min}(\langle r, \text{prob} \downarrow \vartheta_{i,h}, \text{prob} \downarrow \vartheta_{m,k}\rangle)\right)\right) \rangle, (\text{sim} \downarrow \vartheta_{1,1}, \text{rule}), +(h, 1, h), +\langle k_1, k\rangle \rangle, (\langle max \downarrow W_{1,1} \downarrow \vartheta_{m,0}, m, \varepsilon, \left(\langle res, max \downarrow \vartheta_{i,j}, r\rangle, +\langle i, 1, i\rangle \rangle, (n, 1), +\langle k_1, n_1\rangle, (j, 1), +\langle h, 1, j\rangle \rangle, (\langle max \downarrow W_{m_1} \downarrow \vartheta_{i,j}, l, \varepsilon, \langle (\tau_2, \text{false}) = (\tau_3, \text{true}) \rangle. \]

Note that \( \rho(T_1, T_2) = \rho(T_2, T_1) \rho(T_1, T_1) = 1 \) is a complete match, \( \rho(T_1, T_2) = 0 \) – if there are no sentences of the same structure in texts \( T_1 \) and \( T_2 \).

**Realization**

The implementation of the structure is the language constructions formation from the elements of its carrier through the execution of algorithms associated with signature operations according to the rules of axiomonics:

\[ \mathcal{C}_P \xrightarrow{\text{gr}} \mathcal{H}(\mathcal{C}_P), \]

where \( \mathcal{H}(\mathcal{C}_P) \in \mathcal{C}_P \). As the constructor’s work result is a number \( \mathcal{H}(\mathcal{C}_P) \in [0; 1] \), is obtained, which reflects the degree of similarity of the text.

**Conclusions**

In this paper, constructors are developed and presented that model a natural language text in the form of a stochastic grammar that displays the structures of sentences in it. This approach allows you to highlight the syntactic features of the construction of phrases by the author, which is a characteristic of his speech. Working with a sentence as a unit of text for analyzing its construction will allow you to more accurately capture the author's style in terms of the words used, their sequences and speech style characteristic. It allows you not to be tied to specific parts of speech, but reveals the general logic of constructing phrases, which can be more informative in terms of the author's style characteristics for any text.

The presented work is a theoretical basis for solving the problems of the text authorship establishing and identifying borrowings. Experimental studies have also been carried out, the results of which are partially presented in [3]. The statistical similarity of solutions to the problems of establishing authorship and identifying borrowings was experimentally revealed, which will be presented in the next article of the authors.

It is planned to use the created model in the future to determine the authorship of natural language texts of various directions: fiction and technical literature.

**References**

5. Площ М. Я. Граматика української мови. Морфеміка. Словогін. Морфологія. Підрозділ / М. Я. Площ.

Віктор Шинкаренко
Viktor Shynkarenko
DrS. Professor of Computer and information technologies department, Ukrainian State University of Science and Technologies, Dnipro, Ukraine, e-mail: shinkarenko_vi@ua.fm
https://orcid.org/0000-0001-8738-7225

Український державний університет науки та технологій

Інна Демидович
Inna Demidovich
PhD student of Computer and information technologies department, Ukrainian State University of Science and Technologies, Dnipro, Ukraine, e-mail: 2019demidovichinn@gmail.com
https://orcid.org/0000-0002-3644-184X

Український державний університет науки та технологій