UDC 007:681.512.2

THEORY OF HIERARCHICAL NUMBERS IN CALCULATION PROBLEMS SEMANTIC SIMILARITY OF NATURAL LANGUAGE CONSTRUCTIONS

I. Yu. Kashirin, Dr. Sc. (Tech.), full professor, RSREU, Ryazan, Russia; orcid.org/0000-0003-1694-7410, e-mail: igor-kashirin@mail.ru

The algebra of hierarchical numbers, operations and relations of the algebraic system are considered. A graphical representation of hierarchical numbers and operations with them is provided, and the remarkable properties of the operations are shown. Methods for normalizing hierarchical numbers for their subsequent use in processing natural language constructs are listed and explained. To use the theory of hierarchical numbers, ontologies of knowledge models are being developed in terms of generic taxonomies, which also have a hierarchical structure. General and applied ontologies are distinguished, which have significant differences in their design and application for understanding natural language sentences.

As a cross-cutting example, we took the subject area of English-language political articles of international electronic media, in particular: RT, cnn, TASS, NYTimes. The technology for calculating the semantic similarity of natural language constructions is considered, for which the well-known bert-base-cased neural network models of the latest versions are used, as well as the author's IYu-bert-cased model. A new method for computing semantic similarity using hierarchical number theory is presented.

The experimental part of the material is based on the use of software tools of the Python v.3 language (Anaconda 3): the Spacy library v.3.2.1, the CorpusMining v.2.1 retriever, the mIYu-bert v.1.0 software package. The last two tools were implemented by the author of the material.

The completed series of experiments allows us to qualify the methodology for using hierarchical numbers in calculating semantic similarity as the basis of a technology that is not inferior in efficiency to currently available international analogues.

***The purpose** of the work is to present the effective use of hierarchical number algebra to obtain and use a new neural network technology used to solve problems of automatic calculation of semantic similarity of natural language constructions.*

***Keywords:** hierarchical number theory, neural Bert models, natural language analysis, ontological taxonomies, semantic similarity.*

DOI: 10.21667/1995-4565-2024-88-38-52

Introduction

Natural language constructions are words in various forms, phrases, sentences and meaningful texts. The calculation of the semantic similarity of such constructions is considered by scientists as the main component of the tasks of relevant information retrieval, as well as the design of neural networks and other deep learning models (ML models) for natural language analysis [1]. In particular, the idea of semantic similarity is used in ML models that classify the texts of political articles of electronic mass media (mass media) into groups such as:

− fake news [2];
− articles provoking the anger of individual social groups [3];
− toxic publications that cause a depressed state in the reader [4];
− articles with positive or negative emotional mood [5];
− publications introducing pro-Western ideology [6];
− articles calling for internationalism and peacefulness [7].

The characteristic of the semantic proximity of texts allows us to distinguish a hierarchy of classes, which can include media materials from different states. The above defines the purpose of this article, which is to create formal means of calculating the semantic proximity of natural language constructions as a theoretical basis for designing appropriate applied algorithms.

The current political situation, expressed in an unprecedented escalation of the information war, makes it possible to call the formulated goal very relevant. To achieve the stated goal, it is proposed to use the theory of hierarchical numbers [8,9].

The theoretical part
The algebra of hierarchical numbers

Hierarchical numbers are numbers of the form [s] $a_0 . a_1 . a_2 . \ \ldots . a_i . \ldots . a_n$, where a_i – Hierarchical numbers are numbers of the form positive integers from a set N = {0, 1, 2, 3…}. s – the symbol of the sign «+» or «-»,a positive sign may not be indicated. For example, hierarchical numbers may look like this: 0.0.12.48.0 or -2.33.0.0.4 . The symbol denoting the set of hierarchical numbers is H.

These numbers are already used in one way or another in the practice of classification or addressing, for example, a universal decimal code or the IP address of a computer on a global network. However, the introduction of an algebraic system of hierarchical numbers makes it possible to perform operations with them similar to formal arithmetic, and to isolate binary relations for their comparison and analysis of non-trivial properties of operations and relations.

Consider the algebra of binary hierarchical numbers.

Let B be a set of numbers with elements {0, 1}, n ϵ B (n = 0 or n = 1), let there also be a highlighted character ".".

The set A = B ∪ "." ∪ ∧ is defined as an alphabet with integers from B, where "∪" is the operation of combining sets, and ∧ is an empty character.

Then the grammar:

$\hat{h} \rightarrow \Lambda$, $\hat{h} \rightarrow h$, $\hat{h} \rightarrow -h$,
$h \rightarrow < n >$, $h \rightarrow < n > . <h>$

describes a set of binary hierarchical numbers H with elements of h.

Examples of binary hierarchical numbers can be given: 0.1.0.0.1 or 1.0.-1.0 .

Binary hierarchical numbers are numerical indices of the vertices of two binary trees: positive and negative with one common vertex 0.

The generation of a vertex to the left of 0 is performed by the binary operation 0+0=0.0,

the generation of a vertex to the right is performed by the binary operation 0 + 1 = 0.1.

The generation of negative vertices is performed by the operation "-", respectively: 0-0 = -0.0, 0-1 = -0.1.

Graphically, this can be represented by a tree spreading in a positive or negative direction (Figure 1):

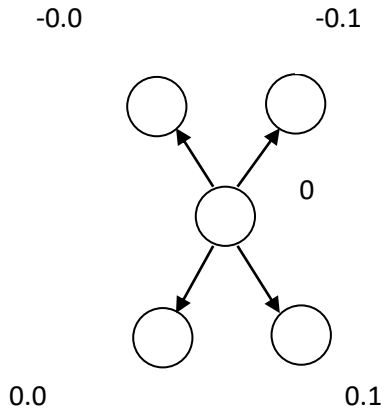

Figure 1 – An image of algebraic operations in the form of trees

However, using negative elements can make the meaning of operations more complicated. For example, the generation of the "+" trace may look like this:

0.1 + 1.1 = 0.1.1.1, 0 + 0.1 = 0.0.1 .

However, the example 0.0.-1 + 1.1 = 0.0.-1.1.1 indicates the presence of more complex tree travel routes using not only descent but also local ascents (Figure 2).

This application of hierarchical numbers will be discussed further with specific examples. The reverse operation of generation, the removal of the terminal vertex "--", is unary:

0.1.1.1-- = 0.1.1, 0.1.0-- = 0.1, 0.-1.-1 -- = 0.-1.-1 .

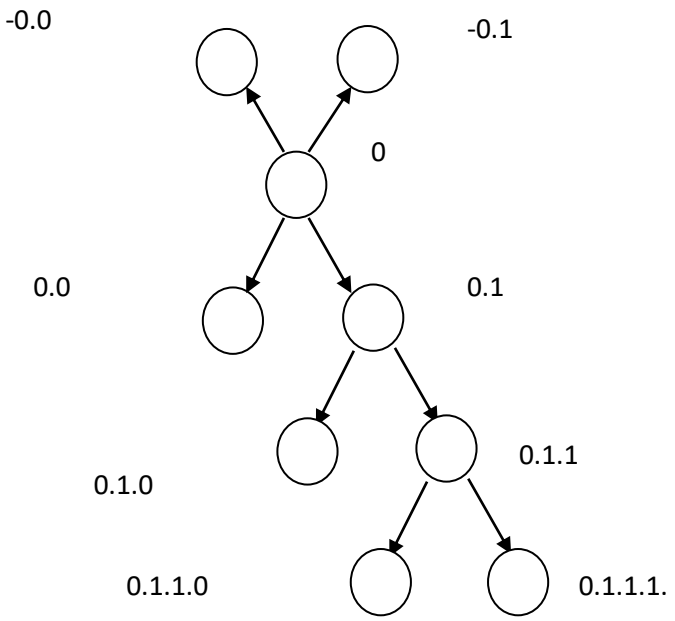

Figure 2 – An example of trees with a subtree of negative hierarchical numbers

In graphical interpretation, the number can be considered the absolute index of any vertex, i.e. starting from the top of the tree 0 or relative, displaying the path through the tree from one of any vertices to other vertices up and down. The absolute index always starts with the character 0.

When solving practical problems, only the positive part of the algebra of binary hierarchical numbers can be considered. In this case, operations claiming to receive a negative index will have a result of 0.

One more rather popular operation «º»can be cited, namely, the calculation of the most common vertex, which is interpreted as a search for a common ancestor of two argument vertices:

$$0.1.1.1 \text{ º } 0.1.0 = 0.1.0 \text{ º } 0.1.1.1 = 0.1 \quad .$$

This generalization/multiplication operation is commutative, i.e. a º b = b º a.

This generalization/multiplication operation is commutative, i.e. multiplying a positive number by a negative number is always 0. "

An important operation is "∧" as the calculation of the path from the vertex specified by the first argument to the vertex specified by the second argument.

Examples of such calculations could be given for the previous figure:

$0.1.0 \wedge 0.1.1.1 = 0.1.0.\ 0.1.\ 0.1.1.\ 0.1.1.1$
$0.1.1.1 \wedge 0.1.0 = 0.1.1.1\ 0.1.1.\ 0.1\ 0.1.0$
$0.1.0.\ 0.1.\ 0.1.1.\ 0.1.1.1 \cong 0.1.1.1\ 0.1.1.\ 0.1\ 0.1.0$

Here " ≅ " is the ratio of the equality of the lengths of two hierarchical numbers. However, such a calculation leads to an unnecessarily complicated result.

Note that the common ancestor for 0.1.1.1 and 0.1.0 is 0.1, from which both argument numbers begin. As a result, when calculating the operation "∧", these fragments are omitted for all vertices of the path from the first vertex to the second. This is necessary to get an idea of the complexity of the path from the first vertex to the second, despite the depth of the tree.

Then the correct operation "∧" turns out like this:

0.1.0 ∧ 0.1.1.1 = [0.1.]0. [0.1.] [0.1.]1. [0.1].1.1 = 0.1.1.1
0.1.1.1 ∧ 0.1.0 = [0.1.]1.1 [0.1.]1. [0.1] [0.1.]0 = 1.1.1.0
0.1.1.1 ≅ 1.1.1.0

After considering the semantics of the above operations, it is possible to define a universal arithmetic algebra of hierarchical numbers H:

$$H = < H, \Omega >, \Omega = \{+, -, --, ^{o}, \wedge, \oplus\},$$

where Ω is the signature of the algebra, i.e. the set of operations. All operations are binary, with the exception of "- -", which is single. The meaning of the operation "⊕" will be discussed later, using appropriate examples.

The considered algebra can be extended to the algebraic system H = < H, Ω, R > by introducing a set of relations R = {< , >, ≅ , = }, where the relations "a > b" and "b < a " are respectively "number a is more complex than number b" and "number b is shorter than number a".

Applying hierarchical numbers to design ontological taxonomies

Taxonomies are semantic relations that set the order on a set of concepts or properties and are used in modern knowledge models. The main taxonomy is the generic one, defined by the is-a relation. A more powerful relation can be considered the triple icf relation (triad), which forms the corresponding icf ontology of the general level.

General ontologies describe concepts of the most abstract level of knowledge, which rarely have a representation in the dictionary forms of natural language. However, their remarkable property is a compact description of the complex of basic relations: is-a, contr, form (i,c,f):

$$icf(a,b,c) = is\text{-}a(a, b) \cup is\text{-}a(a, c) \cup form(a, b) \cup form(a, c) \cup contr(b, c),$$

where is-a (a, b) means that the concept of b belongs to the more general class a, form (a, b) – the concept of a can manifest itself in the form of the concept of b, contr (b, c) – the concepts of b and c are in some way opposite (in volume, content, etc.). Since binary relations are sets of pairs, the operation of set-theoretic union "∪" is used here. A more detailed description of the icf triads can be found in [10].

To calculate the semantic similarity of natural language constructions, it is proposed to use, in addition to classical language models such as bert-cased, word2vec [11], also ontological taxonomies marked with hierarchical indexes. If ML language models are the main mechanism for calculating similarity, then marked-up taxonomies will be considered additional tools or "means of clarifying adjustment".

Further examples will be taken from the subject area "political news articles in English-language electronic media". The task of calculating semantic proximity in this case is an important element in solving the problem of automatic classification of political articles by ideological orientation into "pro-Western" and "pro-Russian" [6].

Let's consider a general ontology describing the topic of "political events". It is shown in Figure 3 and consists exclusively of the icf triads discussed earlier. Due to the *polymorphic properties of icf relations* [10], all vertices of the tree of this taxonomy can be viewed through refraction from the angle of concepts corresponding to any other vertices.

For example, the "Objects" of events can, under certain conditions, become "Subjects" and vice versa. Each of the participants in the events may be "Peaceful" under certain conditions, or may be "Combative". The same can be said about the "Theme of the event", which may consist in considering any "Objects" or "Subjects", and may be "Peaceful" or "Combat".

For the example given, operations with hierarchical numbers may be useful in calculating hyponyms and hyperonyms of concepts, as well as determining the common hyperonym of two or more concepts. For example, to define hyponyms of the same level for the concept of "Event participants", the following expressions can be calculated:

$$0.0.1+ 0 = 0.0.1.0 \text{ и } 0.0.1 + 1 = 0.0.1.1,$$

which corresponds to getting the hyponyms "Objects" and "Subjects". The general hyperonym for the concepts of "Event topic" and "Subject" is calculated using the operation «^{o}»:

$$0.0.0\ ^{o}\ 0.0.1.1 = 0.0,$$

which corresponds to the concept of "Elements of the situation". Obviously, obtaining hyperonyms for a concept is performed by an operation «--».

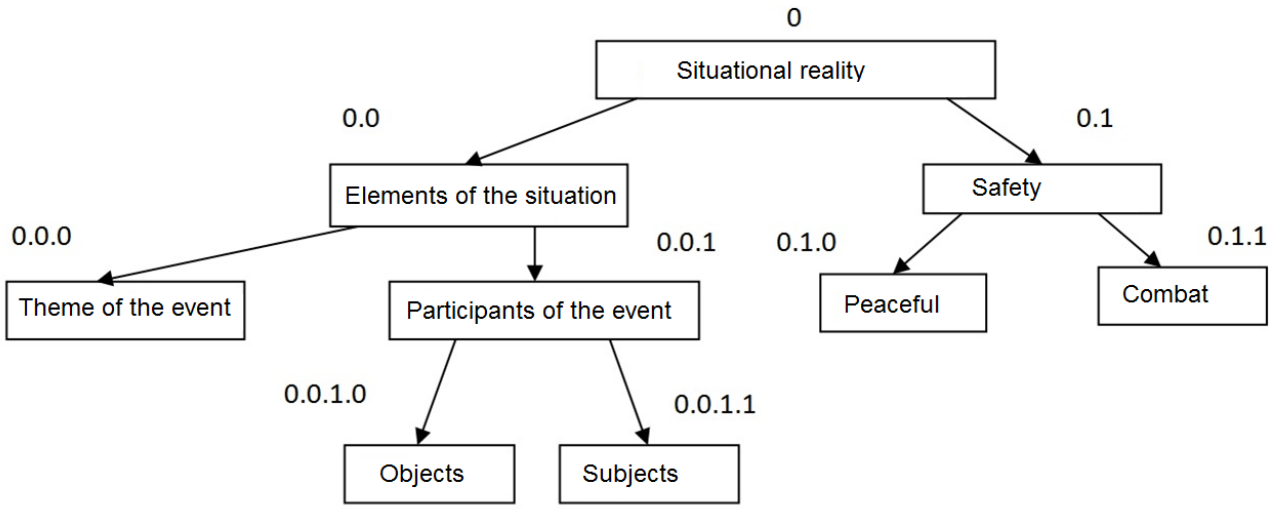

Figure 3 – Graphical representation of the general ontology of "Political events"

Normalization of hierarchical numbers

Hierarchical numbers have many of the features of binary or decimal numbers in classical number theory and formal arithmetic, but at the same time, in their original form, they cannot be used as numerical features in the design of deep learning models. Therefore, for their use, normalization may be necessary, transforming these numbers into decimals from the segment [0, 1]. In addition, it can be seen from the previous presentation that hierarchical numbers are designed in such a way that semantically similar concepts are indexed by numbers similar in structure. Therefore, during normalization, it makes sense to keep the proximity of normalized numbers corresponding to close vertices in the hierarchy. In the tasks of natural language analysis, this would simultaneously mean semantic proximity. Let's take an example of the well-known idea of semantic space, graphically represented by Figure 4.

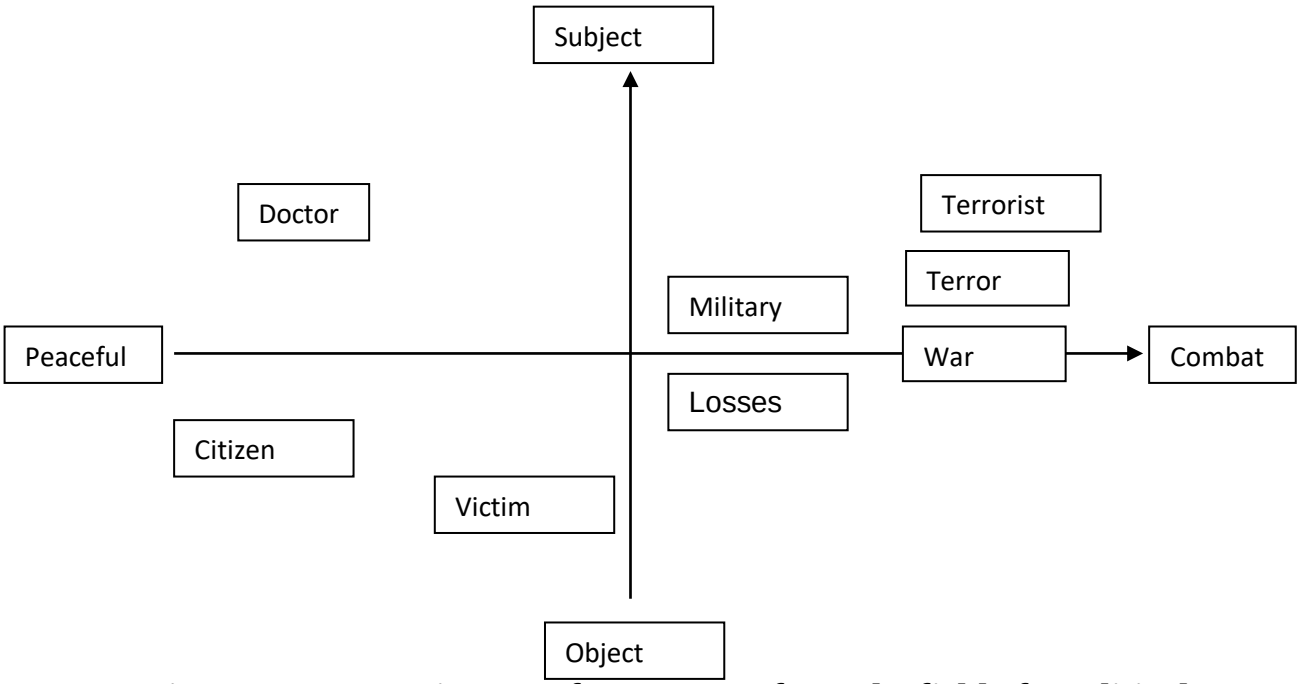

Figure 4 – Semantic space for concepts from the field of "Political events"

Two axes of semantic coordinates are set here: "Object - Subject" and "Peaceful - Combat". This means that all other concepts should be located in this space, depending on their proximity in the chosen coordinate system. It would seem that if this principle was followed, each word could be indexed by two numbers from the proposed coordinate system. However, the proposed measurements are clearly not enough if we expand the system of basic concepts, for example, with the concepts of "Topic - Participants", which also belong to the taxonomy of the example discussed earlier (Figure 5).

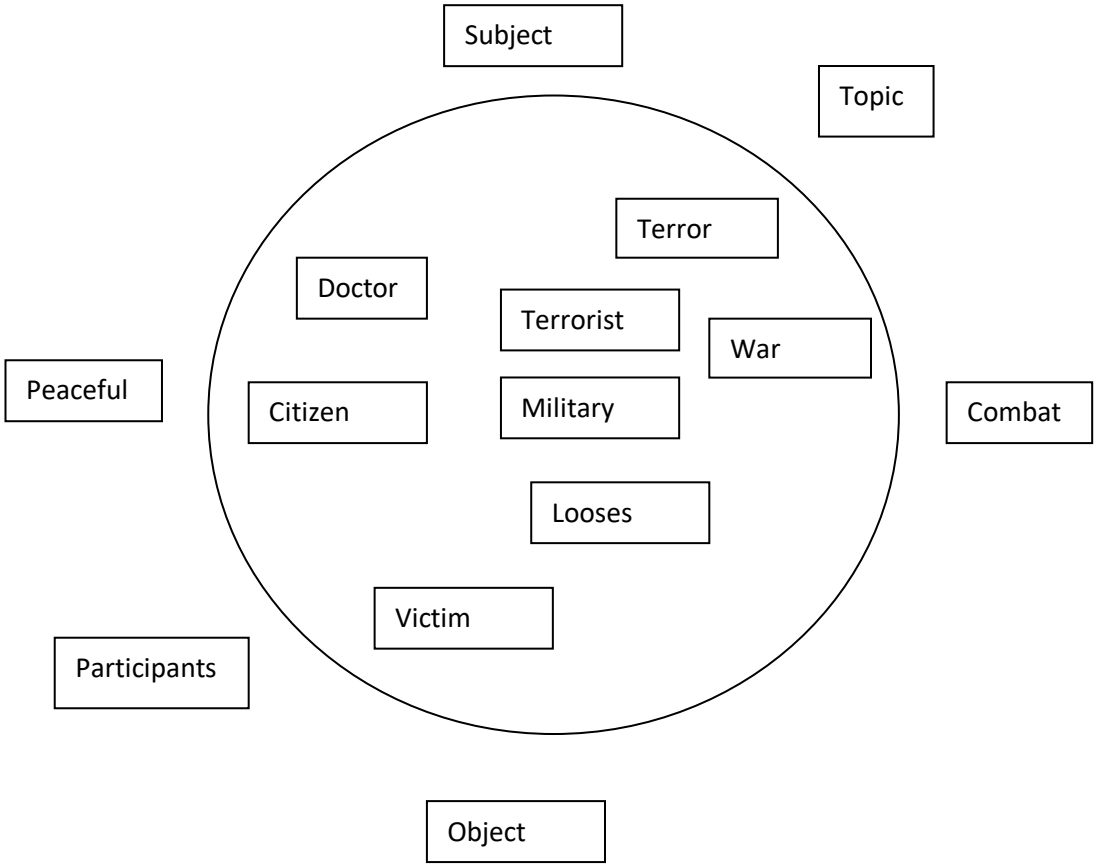

Figure 5 – Extended semantic space

It becomes clear that by moving objects towards the "Participants", we simultaneously increase the proximity to the "Peaceful", although this is not always correct. Therefore, it is impossible to place a concept diagram on a plane, and it is multidimensional by nature.

On the other hand, two types of semantic similarity can be distinguished:

− generic similarity (synonyms, hyponyms);

− topological similarity (situational, joint presence of words in sentences).

As will be shown below, both of these types can be implemented in an application-level taxonomy. Such generalized proximity will be called taxonomic proximity. In this case, it is possible to implement an algorithm that is a simple binary distribution (Figure 6).

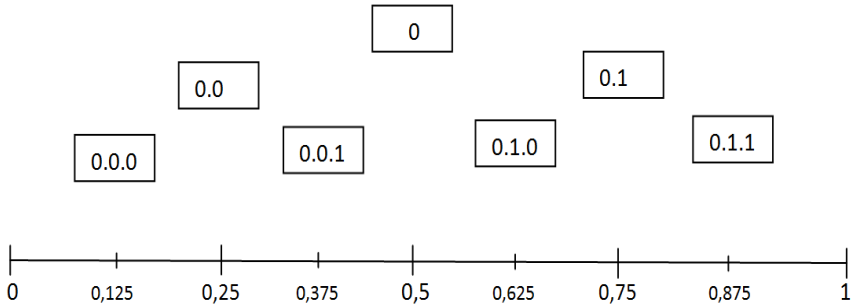

Figure 6 – Simple binary distribution of hierarchical numbers

It is clear from the figure that with each new level of hierarchical numbers, decimal numbers become more and more detailed, since the appearance of new levels is associated with the division of previous intervals. At the deepest levels of the hierarchy, there may be a loss of precision in encoding numbers. In this case, you can use the logarithmic distribution shown in Figure 7.

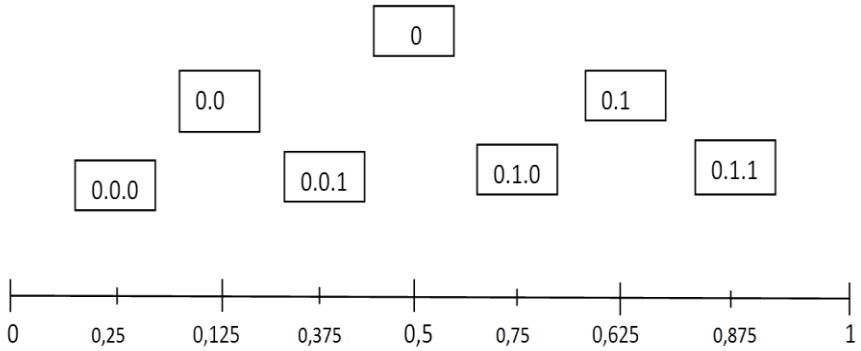

Figure 7 – Logarithmic distribution of hierarchical numbers

The principle of proximity of hyponyms is implemented here: deeper levels of numbers are more similar in meaning. Adding each new row to the depth results in recalculation of all vertex indexes. If we denote by n the accuracy of the taxonomy (the maximum number of hierarchy levels for the current implementation of the taxonomy), then the distribution as a Norm normalization function looks like this:

Norm(0) = 0,5	:	0-0,4(9)	0,4(9)-0,9(9)
Norm(0.0), Norm(0.1)	:	$0,(0^n)1$	$0,9(9^n)$

Norm(0.0.0), Norm(0.0.1), Norm(0.1.0), Norm (0.1.1): $0,(0^{n-1})1$ $0,(0^{n/2})9$

The considered normalization options are applicable only to positive binary hierarchical numbers. For more complex cases, it is necessary to use the splitting of decimal numbers into even more fragments. Such normalization remains outside the scope of this article.

Hierarchical numbers for applied ontological taxonomies

As noted earlier, polymorphism is typical only for general taxonomies of knowledge models. Real media articles contain natural language constructions corresponding to the concepts of applied ontologies, taxonomies of which can be built, for example, on the basis of causa or is-a relationships. Fragments of icf triads may occur in these taxonomies, but these are rather exceptional cases. However, for any taxonomy, all operations and relations of the algebraic system of hierarchical numbers remain valid.

At the same time, to solve the problem of understanding texts, applied ontologies should rely on the concepts of general ontologies. This makes it possible to use limited forms of polymorphism by introducing multiple inheritance. Such an example is given in Figure 8.

Here, the relationship arrows for applied-level concepts point upwards, indicating the essence of which hyperonyms they inherit. However, these concepts are hyponyms, i.e. more specific in relation to concepts corresponding to the peaks of higher levels and the general ontology. This means that hierarchical indexes of applied concepts should be formed on the basis of indexes of concepts of the general ontology, and therefore they will be more complex.

Let's now consider an example with specific offers from real electronic media:

1."March 23, 2024 Shooting at Moscow concert venue leaves over 130 dead." (cnn)

2."On March 23, 2024, terrorists attacked Moscow, killing more than 130 citizens." (ru.wikipedia.org)

3."Terrorists strike the Russian capital, leaving at least 60 dead." (RT)

4."On 11.09.2001, two Boston planes destroyed the World Trade Center in New York." (NYTimes)

5."On October 11, 2022, a new multifunctional medical center was opened in Lugansk." (TASS)

These examples demonstrate sentences that are both similar in meaning and semantically different.

Now, for the analysis of natural language constructions, more specific concepts will be needed, which are present in one form or another in the above sentences. As follows from Figure 8, new concepts are now becoming: "Attack", "Victim", "Civilian (Citizen)", "Military", "Peaceful victims", "Combat losses", "Terror", "War", "Terrorist".

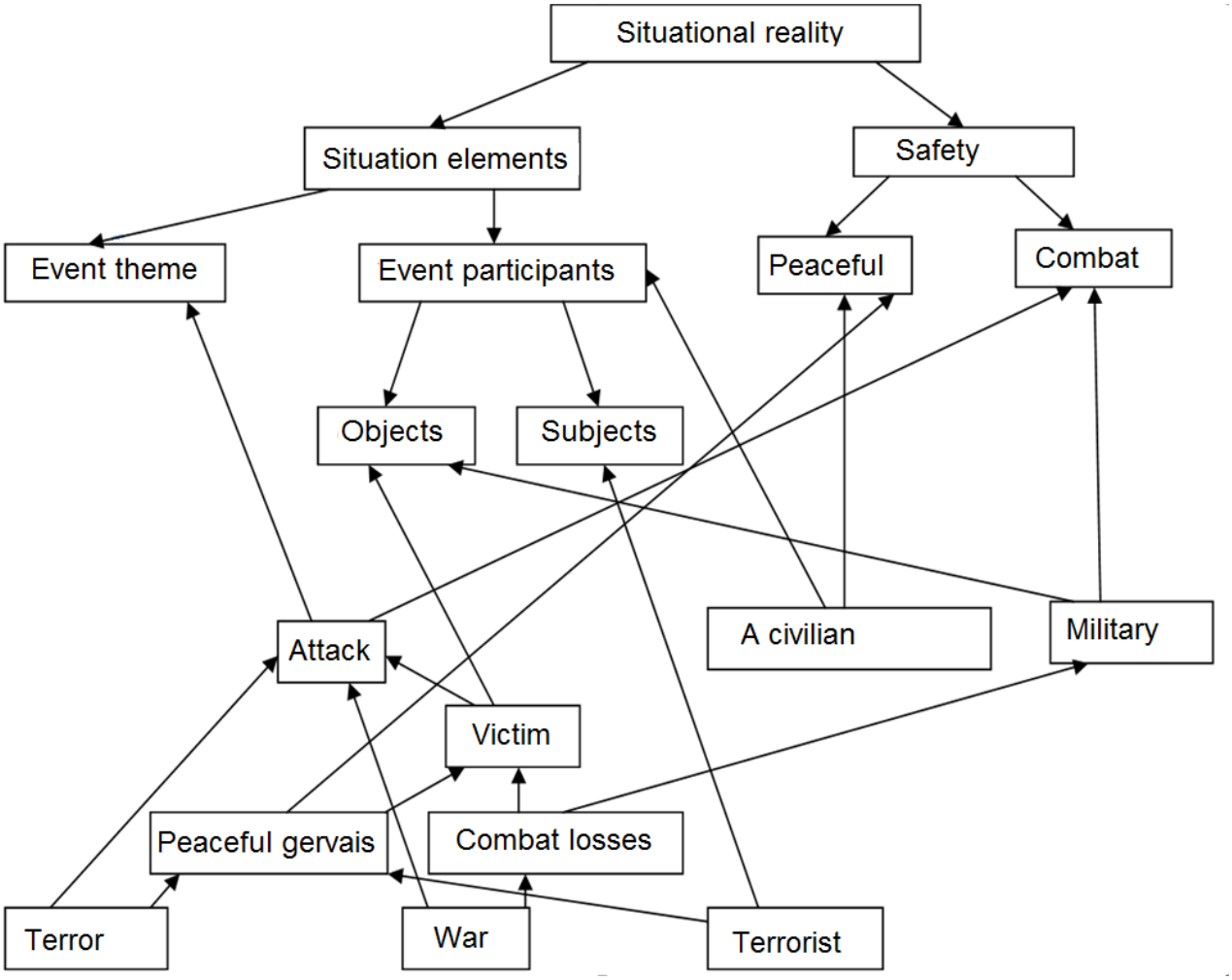

Figure 8 – Ontology with applied taxonomy and multiple inheritance

Let's consider two methods of indexing the vertices of applied taxonomy: the tracing method and the multiple inheritance method. To implement both methods, you will need to expand the set of binary hierarchical numbers to complex hierarchical numbers, which may include the use of negative elements "-1". Note that, as in the case of classical, for example, decimal numbers, the algebra of hierarchical numbers itself remains unchanged. It is only necessary to clearly define what they will mean.

The tracing method

The numbers indicate the recording of a route from one vertex of the taxonomic tree to another.

Let's consider the meaning of the operations of the algebra H from this point of view.

It is required to define a new concept of "Attack" as a combination of the concepts of "Event theme" and "Combat":

Event theme (0.0.0) + Combat (0.1.1) = Attack (0.0.0.-1.-1.1.1)

0.0.0 → 0.0 → 0 → 0.1 → 0.1.1 = 0.0.0.-1.-1.1.1

in the notation of algebraic operations: 0.0.0 - 1 - 1 + 0 = 0.0.0.-1.-1.1.1 ,
where -1 means climbing up one vertex of the tree.

Subject + Peaceful = A Civilian

0.0.1.1 – 0.0.1 – 0.0 – 0 + 1 + 1 = 0.0.1.1.-1.-1.-1.0.1.0.1.0

In operations: 0.0.1.1-1-1-1+1+0 = 0.0.1.1.-1.-1.-1.0.1.0.1.0

Subject + Combat = Military

0.0.1.1 – 0.0.1 – 0.0 – 0 + 0.1 + 0.1.1 = 0.0.1.1.-1.-1.-1.0.1.0.1.1

Object + Peaceful= A Civilian

0.0.1.0 – 0.0.1 – 0.0 – 0 + 0.1 + 0.1.0 = 0.0.1.0.-1.-1.-1.0.1.0.1.0

Object of Attack = Victim Event theme + Combat + Object = Victim

(Theme) 0.0.0.-1.-1.1.1 – 0.0.0. – 0.0 + 0.01 + 0.0.1.0 = 0.0.0.-1.-1.1.1.-1.-1.1.0

In operations: 0.0.0-1-1+1+1 -1-1+0+1+0 = 0.0.0.-1.-1.1.1.-1.-1.0.1.0

Combat + Victim = Combat Victim

Comat + (Object + Combat + Event theme) ,

where the "Combat " index is not duplicated as a result, since the "+" operation is not associative.

Peaceful + Victim = Peaceful Victim

Attack + Peaceful Victim = Terror

Citizen's Attack = Terror

Attack + Combat Sacrifice = War

The method of multiple inheritance

The numbers represent the indices of two ancestral vertices separated by the digit "-1". To obtain such numbers, it is necessary to use the operation of multiple inheritance synthesis "⊕". Here you can refer to Figure 9, which presents new, deeper concepts of the applied level: "shooting, hitting, attacking, killing, leaving, destruction, death." These concepts are often found in dictionary form in political articles. Let's look at the examples.

Subject + Peaceful = A civilian (The abbreviated form of a complex number)

the route in the tree: 0.0.1.1 → 0.0.1 → 0.0 → 0 → 0.1 → 0.1.0

multiple inheritance operation 0.0.1.1 ⊕ 0.1.0 = 0.0.1.1.-1.0.1.0

The Object of The Attack = Victim Event theme + Combat + Object = Victim
(The abbreviated form of a complex number)

the route in the tree: 0.0.0. →1. → 1.1.1 → 0.0.0. → 0.0 → 0.01 → 0.0.1.0
multiple inheritance operation:
0.0.0 ⊕ 0.1.1 ⊕ 0.0.1.0 = 0.0.0.-1.0.1.1.-1.0.0.1.0

Figure 9 shows hierarchical indexes only for the vertices involved in the examples considered, since displaying all the indexes would make the figure difficult to read.

The software implementation uses two deep learning language models and additional tools as a means of clarifying adjustment, mentioned earlier, to compare with existing implementations.

The models are:

− a well-known language model of the latest current version of bert-base-cased, developed and constantly updated by the research department of Google AI Language;

− the IYu-bert-base-cased model, obtained by the author of the article on the basis of additional training of bert-base-cased (one of the previous versions) on the CorpusMining v.2.1 retriever cases.

An additional toolkit is the mIYu-bert v.1.0 software package, also programmatically implemented by the author of this article.

The theory of hierarchical numbers can be used in practice in several ways.

The first of them is a complete reorganization of the bert model by adjusting the number and content of layers of encoders and decoders of the neural network in accordance with the number and content of levels of ontological taxonomy.

The experimental part
Software implementation of semantic similarity calculation in Python

In this case, hierarchical indexes can be the basis for word vectorization in the tokenizer of the bert model. In this case, the original bert model can be significantly simplified, provided that the accuracy characteristics are preserved. However, this method was not considered due to its great complexity, since thousands (!) of highly qualified programmers worked on the basic model.

The second method is to apply normalized hierarchical numbers only in the modernization of the bert model tokenizer. The methods of normalization of hierarchical numbers were discussed earlier. This method is also very laborious, but its implementation requires fewer qualified specialists compared to the first method.

The third method of software implementation using hierarchical numbers is the introduction of special tokens in the vectorization of natural language texts. It is quite effective, implemented on a number of examples and was considered in [6].

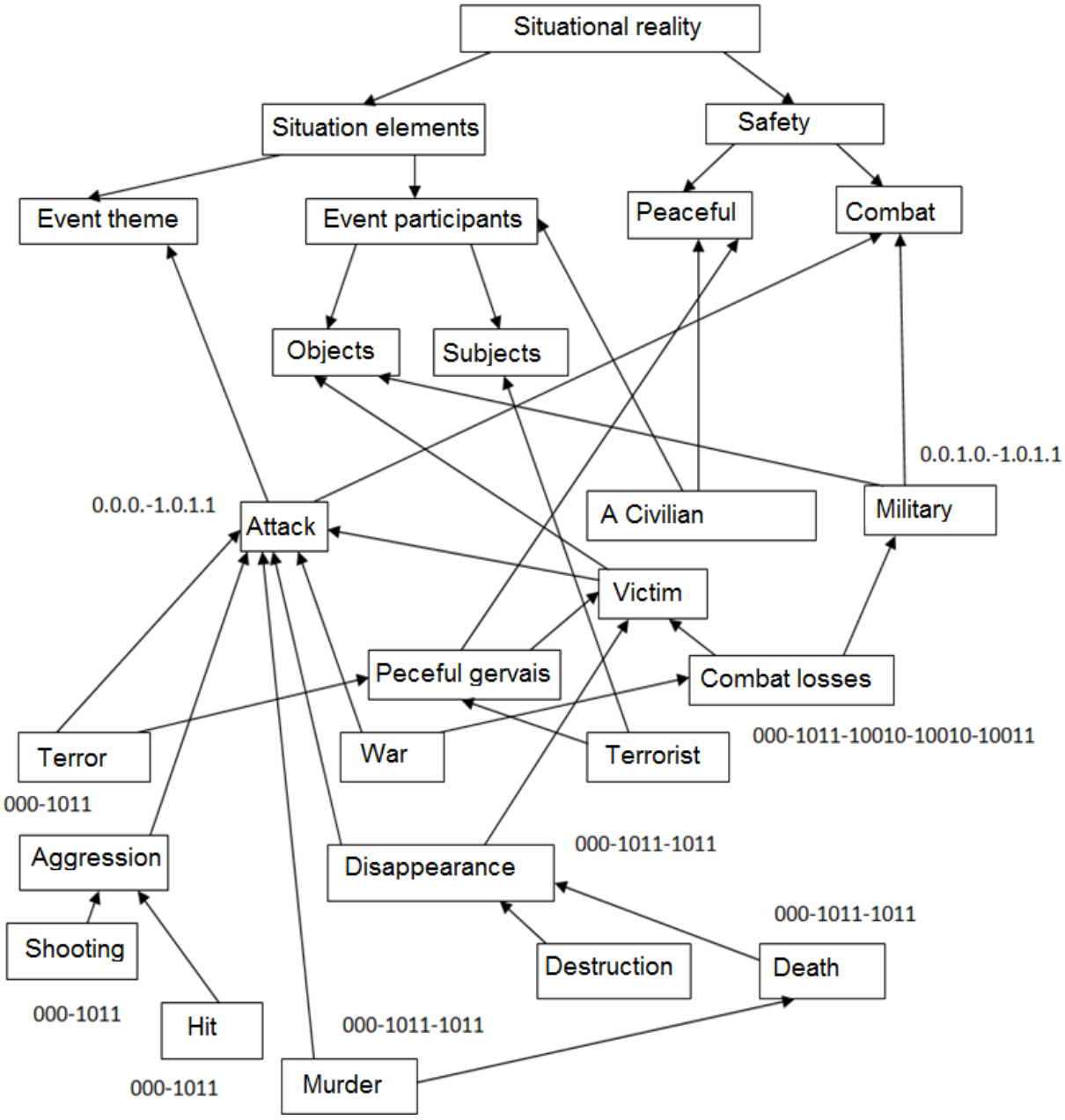

Figure 9 – Ontology with deep multiple inheritance

The fourth method, considered in this article, involves the preliminary calculation of semantic similarity by language models with subsequent correction of characteristics by additional tools. The toolkit, regardless of ML models, analyzes the input structures of texts for the presence of similar or opposite concepts using ready-made ontologies containing hierarchical indexes. Index comparison is implemented for words with the same type of markup during preliminary analysis by the spaCy toolkit v.3.2.1.

Semantic proximity is calculated for two words by the difference of hierarchical indices based on the operation "º" and the hierarchical coefficient μ, which is the threshold for increasing or decreasing the characteristic of semantic similarity calculated by language models.

Let's define the function σ:

$\omega_i > \omega_j$, $\sigma(\omega_i, \omega_j) = 1$,

$\omega_i \leq \omega_j$, $\sigma(\omega_i, \omega_j) = 0$,

где– два произвольных иерархических числа.

where ω_i and ω_j are two arbitrary hierarchical numbers.

Let ω_i^1 , ω_j^2 − hierarchical numbers that are indexes of words of the same syntactic markup in sentence 1 and sentence 2. Let sentence 1 contain n words and sentence 2 contain m words.

Then you can write an expression that calculates the correction factor W:

$$W = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sigma\left(\omega_i^1, \omega_j^2\right)}{n * m},$$

$$\text{if} \sum_{i=1}^{n}\sum_{j=1}^{m}\sigma\left(\omega_i^1, \omega_j^2\right) > \mu, \text{then W, it is taken with "+", otherwise " − ".}$$

The following is a fragment of a Python program that calculates the semantic similarity of five sentences.

```
# Calculating  semantic similarity of words
word1 = "attack"
word2 = "shooting"
word3 = "destroyed"
word4 =  "killing"
embedding1 = get_bert_embedding(word1)
embedding2 = get_bert_embedding(word2)
similarity = 1 - cosine(embedding1, embedding2)
print(f" Semantic similarity between words '{word1}' and '{word2}': {similarity}")
embedding1_IYu = get_bert_embedding_IYu(word1)
embedding2_IYu = get_bert_embedding_IYu(word2)
similarity_IYu = 1 - cosine(embedding1_IYu, embedding2_IYu)
print(f"IYu Semantic similarity between words '{word1}' and '{word2}': {similarity_IYu}")
# Calculating  semantic similarity of words of sentences
sentence1 = "March 23, 2024 Shooting at Moscow concert venue leaves over 130 dead."
sentence2 = "On March 23, 2024, terrorists attacked Moscow, killing more than 130 citizens."
```

```
sentence3 = "Terrorists strike the Russian capital, leaving at least 60 dead."
sentence4 = "On 11.09.2001, two Boston planes destroyed the World Trade Center in New York."
sentence5 = "On October 11, 2022, a new multifunctional medical center was opened in Lugansk."
embedding1 = get_bert_embedding(sentence1)
embedding2 = get_bert_embedding(sentence2)
embedding4 = get_bert_embedding(sentence4)
embedding5 = get_bert_embedding(sentence5)
similarity_sentences = 1 - cosine(embedding1, embedding2)
print(f" Semantic similarity of words of sentences '{sentence1}' and '{sentence2}': {similarity_sentences}")
similarity_sentences = 1 - cosine(embedding1, embedding4)
print(f" Semantic similarity of words of sentences '{sentence1}' and '{sentence4}': {similarity_sentences}")
similarity_sentences = 1 - cosine(embedding1, embedding5)
print(f" Semantic similarity of words of sentences '{sentence1}' and '{sentence5}': {similarity_sentences}")
embedding1_IYu = get_bert_embedding_IYu(sentence1)
embedding2_IYu = get_bert_embedding_IYu(sentence2)
similarity_sentences_IYu = 1 - cosine(embedding1_IYu, embedding2_IYu)
print(f"IYu  Semantic similarity of words of sentences '{sentence1}' and '{sentence2}': {similarity_sentences_IYu}")
embedding5_IYu = get_bert_embedding_IYu(sentence5)
similarity_sentences_IYu = 1 - cosine(embedding1_IYu, embedding5_IYu)
print(f"IYu Semantic similarity of words of sentences '{sentence1}' and '{sentence5}': {similarity_sentences_IYu}")
```

The results for the selected language models and the model using hierarchical numbers (**mIYu-bert**) are shown in the table.

In sentences 1-3, the wording of various media outlets refers to the terrorist attack at Crocus Hall in Moscow. Sentence 4 is an example of news about the mall attack in New York City. Sentence 5 is a message about the commissioning of a new municipal medical center in Luhansk.

Experiments were conducted to calculate the semantic similarity of different pairs of five sentences selected from the media. The table shows the results of comparing all offers with offer 1. These data most vividly reflect the results of other experiments.

Table – Results of semantic similarity calculation

№	Title of the publication	Similarity bert-cased	Similarity IYu-bert	Similarity mIYu-bert
1	cnn	1.0	1.0	1.0
2	ru.wikipedia.org(norm)	0.9041026234626770	0.8125963211059570	0.8631073324010911
3	RT	0.9349522590637207	0.8757287859916687	0.9289013442340876
4	NYTimes	0.8471251726150513	0.7009884119033813	0.6909823476905342
5	TASS	0.8411691188812256	0.6638354063034058	0.6100887839438400

Conclusion

The experiments performed allow us to conclude about a small initial loss of the IYu-bert and mIYu-bert models when calculating the semantic similarity of identical sentences in comparison with the pre-trained bert-case model. However, IYu-bert and mIYu-bert significantly benefit from comparing opposite or very far from each other in terms of sentences, although they find them more similar (0.66, 0.61) than different.

In any case, the mIYu-bert model, modified based on the use of hierarchical numbers, improves the quality of similarity calculation in comparison with the latest version of the bpdtcnyjq bert-cased language model by about 5-8%, and the results are adjusted both upward and downward in semantic similarity, where necessary.

References

1. **Demidova L.A., Moroshkin N.A**. Aspekty razrabotki arkhitektury voprosno-otvetnoy sistemy dlya obrabotki bol'shikh dannykh na osnove neyrosetevogo modelirovaniya. // Vestnik RGRTU. 2023. № 86. P.145-155.
2. **Tida, V.S.; Hsu, S.H.; Hei, X. A.** Unified Training Process for Fake News Detection based on Fine-Tuned BERT Model. *arXiv* 2022, arXiv:2202.01907.
3. **Abro, S., et al.** (2020). Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications, 11*(8), 484–491. https://doi.org/10.14569/IJACSA.2020.0110861.
4. **Su R, Wu H, Xu B, Liu X, Wei L.** Developing a multi-dose computational model for drug-induced hepatotoxicity prediction based on toxicogenomics data. *IEEE/ACM Trans Comput Biol Bioinform* 2019;16:1231–9.
5. **Liu, H.; Zhang, Y.; Li, Y.; Kong, X.** Review on Emotion Recognition Based on Electroencephalography. *Front. Comput. Neurosci.* 2021, 84.

6. **Kashirin I.Yu.** Neyroseti novogo mnogopolyarnogo mira: klassifikatsiya elektronnykh novostey. // Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta. 2024. № 87. P.29-40.

7. **Anastasyev A.A., Astashkin M.S., Agafonov P.A., Kashirin I.Yu.** Determining the reliability of news using Ml-models, knowledge-based. / *IIASU'23 – Artificial intelligence in management, control, and data processing systems. Proceedings of the II All-Russian scientific conference* (Moscow, April 27–28, 2023) : In 5 volumes. – Moscow, Publishing House «KDU», 2023. – Volume 2. – 406 p. – Electronic edition. – URL: https://bookonlime.ru/node/72807 – doi: 10.31453/kdu.ru.978-5-7913-1352-2-2023-406. P-21-27.

8. Definition of Hierarchial Numbers [Electronic resource]. Update date: 02.04.2024 URL: https://kashirin.net/definition-of-hierarchical-numbers. (date of application: .09.04.2024).

9. **Kashirin I.Yu.** Iyerarkhicheskiye chisla dlya proyektirovaniya ICF-taksonomiy iskusstvennogo intellekta. Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta // 2020. № 71. S.71-82

10. The Idea of ICF Relationships and ICF Ontologies [Electronic resource]. Update date: 18.02.2024 URL: https://kashirin.net/the-idea-of-icf-ontologies. (date of application: 05.04.2024).

11. **Xiangyun Lei1, Edward Kim, Viktoriia Baibakova1 and Shijing Sun.** Lessons in Reproducibility: Insights from NLP Studies in Materials Science / arXiv:2307.15759v1 [physics.chem-ph] 28 Jul 2023.