Neural Networks in the Processing of Natural Language Texts in Information Learning Systems

Olha Tkachenko[1, 2], Kostiantyn Tkachenko[2, 3], Oleksandr Tkachenko[1], Roman Kyrychok[2], and Vladyslav Yaskevych[2]

[1] *State University of Infrastructure and Technologies, 9 Kirillivska str., Kyiv, 04071, Ukraine*
[2] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudryavska str., Kyiv, 04053, Ukraine*
[3] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37 Beresteyskyi ave., Kyiv, 03056, Ukraine*

Abstract
Processes of natural language text processing in informational learning systems or informational learning systems with elements of intellectualization are considered. Among these processes, attention was paid to simplification, and normalization of the text, highlighting the main essences of the subject area of the courses supported by the appropriate informational learning system. The use of neural networks for text processing consisted, in particular, of the unification of words, the formation of abbreviations, the removal of redundant clarifications, the replacement of terms (slang words), the removal of clarifying constructions and redundant symbols, and the correction of errors and paraphrasing. Natural language text processing in information learning systems or information learning systems with elements of intellectualization is based on the use of the Transformer model in neural networks, which, with the help of its unique architecture, facilitates the parallelization of processing processes, simplifies the use of these processes, and increases the efficiency and speed of training of the corresponding neural network. The considered model of the neural network effectively determines the patterns of test fragments (words, phrases) and finds connections in the training data of the network. All this contributes to the acceleration of natural language text processing processes, even when using a small amount of training data for training. The use of the Transformer model in neural networks contributes to normalization (words, phrases), simplification of the text and reduction of text volumes, and removal of complex wording. All this contributes to the efficient and quick processing of large texts.

Keywords
Natural language text processing, neural network, normalization, simplification, embedding.

1. Introduction

Processing of natural language in intellectual systems (including educational ones) began when A. Turing proposed a model for testing the system for so-called "consciousness".

Then algorithms for finding connections between words began to be developed, word formation was studied, lexical/syntactic analysis of sentences was performed, etc.

All this became the basis of the first automatic translator.

Progress in natural language processing began with the use of artificial deep neural networks for text processing, which, with the help of statistical, combinatorial, and mathematical methods, determine the relationships between elements themselves [1–7].

This contributed to the exclusion of such processes in natural language processing as:

- Finding connections between words.
- Word formation research.
- Lexical analyses.
- Syntactic analyses.

CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine
EMAIL oitkachen@gmail.com (O. Tkachenko); tkachenko.kostyantyn@gmail.com (K. Tkachenko); aatokg@gmail.com (O. Tkachenko); r.kyrychok@kubg.edu.ua (R. Kyrychok); v.yaskevych@kubg.edu.ua (V. Yaskevych)
ORCID: 0000-0003-1800-618X (O. Tkachenko); 0000-0003-0549-3396 (K. Tkachenko);0000-0001-6911-2770 (O. Tkachenko); 0000-0002-9919-9691 (R. Kyrychok); 0000-0002-5796-2521 (V. Yaskevych)

And such methods as:

- Normalization.
- Fragmentation.
- Tokenization of text performs pre-processing of text for neural networks.

The recurrent neural network is the first model that was able to achieve good results in natural language processing.

Recurrent neural networks process input data sequentially and store context in blocks of memory that are used "in time" (from iteration to iteration).

The Transformer model in a neural network is a model that, unlike recurrent neural networks, processes data not sequentially, but simultaneously, "exploring" the context of the entire block of data at once using its architecture [8].

The Transformer model in a neural network consists of an encoder and a decoder, which helps to simplify the structure of the text and accelerate the learning of the model.

Among the systems that solve the problems of natural language processing, we highlight:

- ChatGPT (Generative Pre-trained Transformer model in a neural network) is a chatbot that simulates answers to questions. With high accuracy, it can give advice, write simple program code, simulate a certain entity (for example, a database), etc. [9].
- DALL-E is based on the given context, and generates an image, even if the given entities in real life are difficult or impossible to combine [10, 11].
- Grammarly checks the English text for grammatical, lexical, and spelling errors (not correcting the text immediately, but offering options for corrections) [12].
- Rytr generates a variety of texts based on topic, genre, environment, mood, etc. (based on ChatGPT's API) [13].

These systems use a Transformer model in a neural network that helps them process text data efficiently.

In particular, the encoder is used for quick context aggregation, and the decoder is used for generating a new sequence of tokens:

- Words.
- Phrases.
- Individual sentences.
- Text fragments.

ChatGPT remembers large amounts of context used to generate texts. Learning is uncontrolled and controlled [14].

During unsupervised training, the neural network learns to generate words in the correct order using n-context words.

Controlled learning uses already trained neural network on unsupervised data and applies "reinforcement learning with human feedback", which consists of the following steps:

1. *Controlled fine-tuning.*

   A small amount of demonstration data selected by experts is used to train the SFT (supervised fine-tuning) model [15]. This stage is used only once.

2. *Imitation of human preferences*.

   Experts evaluate the output of the SFT model, creating a new data set consisting of "comparison data". A reward model (reward model [16]) is trained on this set.

3. *Proximal policy optimization.*

   A reward model is used to further fine-tune and improve the SFT model.

Grammarly is a Ukrainian system that helps to find and correct errors in English text [12].

Grammarly uses machine learning, deep neural networks, and natural language processing techniques to solve the problems of parsing and improving a piece of text in natural language [17].

GECToR is an approach to correcting grammatical errors in the text based on the principle of "tags, not rewriting" [18].

That is, problem areas in the text are marked with tags, and options for their correction are offered.

GECToR contains an encoder block and two parallel linear layers that are responsible for detecting and correcting errors in the text.

The system works by predicting changes that will occur in the text.

Tokens contain such special tags as:

- "keep"
- "delete"
- "append_t1"
- "replace_t2" (t1 and t2 are new tokens).

There are also special tags to indicate the change of case or tense of the verb.

DALL-E generates an image according to its textual description provided by the user [10, 11].

The connection between textual semantics and visual representation in DALL-E is created with the help of another OpenAI model—CLIP (contrastive language image pre-training) [19].

The CLIP model is trained on a dataset of captions and the pictures they describe.

CLIP model training includes, in particular:

- Passage of images and texts through encoders to display objects in n-dimensional space, where the relationship between the text and the image is determined.
- Finding the level of similarity between text and image embeddings.
- Learning maximizes the level of similarity between n-coded words.
- Using the Transformer model in a neural network to encode text and images.

After training the CLIP model, images are generated taking into account the available visual semantics.

For this, a diffusion model is used, the training of which in DALL-E is used to transform the encoded CLIP text context into an encoded image and to decode it into a human-understandable representation.

Rytr—generates various texts for different situations, for example, for posts on social networks or when generating questions for an interview [13].

Rytr uses ChatGPT to generate texts. Rytr serves as a kind of wrapper for quickly making queries to ChatGPT and filtering its responses [20].

2. Text Processing Methods

Text preprocessing is an integral part of any natural language processing system.

There are many methods to perform text optimization, fragmentation, tokenization, finding the relationship between tokens, etc.

The full cycle of natural language processing involves:

- Tokenization.
- Lexical analysis.
- Syntactic analysis.
- Semantic analysis.
- Pragmatic analysis.

Tokenization—segmentation of text consisting of symbols (letters, spaces, numbers, punctuation marks, etc.) into words and phrases.

The main task of tokenization is the unification of the text, its fragmentation, and division into tokens (morphemes, words, or phrases) according to defined rules.

Normalization of the text involves the unification of tokens, for example, "y.", "year" and "Year" should be reduced to the same form.

Words are divided into separate parts using computational morphology.

In the Ukrainian language are distinguished:

- Prefixes.
- Roots.
- Suffixes.
- Endings.

Categories to which the structure of the word can be attributed:

- Isolated, when it is impossible to divide the word into smaller parts.
- Agglutinative, when the word can be divided into small morphemes.
- Inflectional, when there are no clear boundaries between morphemes and morphemes can acquire several grammatical meanings.
- Polysynthetic, which is similar to agglutinative, but also suggests such a variant, when morphemes are combined and form a complex of words (groups of words), which can be a whole sentence.

Embedding (*vector representation of a word*) is a set of methods by which words are transformed into an n-dimensional vector of real numbers [21].

The simplest method of representing tokens as a vector is to use a one-hot vector [22], which has the dimension of a dictionary and consists of 0.

The token is encoded using the corresponding position in the dictionary, which will be equal to 1 in this vector [23].

The disadvantage of a one-hot vector is that dictionaries can have millions of different values, and deep neural networks usually contain many linear layers of abstraction. Such encoding is inefficient and is replaced by other embedding methods, such as [24]:

- word2vec [25]
- GloVe
- fastText, etc.

These methods have a small (compared to a one-hot vector), fixed, and defined size.

These methods are based on the use of neural networks, and the goal of their training

is the vector approximation of words that are close to each other in the context.
Word2vec is a simple and efficient method of learning vectors for tokens, which can be created using [24, 25]:

- CBOW (continuous bag-of-words) [26].
- Skip-gram (SG) [27].
- Optimization methods (hierarchical softmax and negative selection) [28].

The model used is a single context, which is a fully connected neural network.

The input is a word represented by a one-hot vector [22] of dimension V (dictionary size), where only one value is 1, and all others are 0.

Multi-contextuality works according to the same principle, only the input is not one context word, but a set of context words.

In Skip-gram, context is found using a word.

Before propagating an error is the sum of all errors.

The models discussed above are not very complex, but they take a long time to train on large amounts of data.

Therefore, optimization methods have been developed that speed up learning, namely:

- Hierarchical softmax.
- Negative sampling.

Negative sampling is a method of optimizing the training of word2vec models.

The main problem with the models is that during training, all the weights of the original matrix are used.

Therefore, each iteration of learning a neural network (including such a neural network as a Transformer), which will then be used in the processing of natural language texts, is expensive.

When using negative sampling, not all vectors are taken from the original matrix, but only a small part of them.

To these vectors is added a so-called "positive" vector, which represents (displays) the predicted word.

When using negative sampling, a mechanism must be added that will reduce the impact of tokens that are in the context of "positive."

A unigram table of a certain size is created, which is almost a thousand times larger than the size of the dictionary.

The table is filled with token indexes.

The more often a token occurs in the training sample of the neural network, the more times its index will occur in the unigram table.

Hierarchical softmax uses a binary tree that represents a dictionary of size V (V words must be leaves of the tree).

The number of internal nodes is equal to (V–1). The input matrix representing (displaying) the words is unchanged.

There is no output matrix, and instead, the weights of the internal nodes of the binary tree are used, the number of which is smaller than the size of the dictionary.

The created path is used to estimate the probability of finding a word in the context.

To find a word, you need to move from the root with the selection of a branch using the formula:

$$p(i, left) = \sigma(h * r_i),$$

where r_i are weight coefficients of internal nodes.

The second method of creating embedding is global vectors for representing GloVe words.

It is a log-linear regression that combines the advantages of matrix factorization and local context methods (for example, skip-gram, CBOW) [26, 27, 29].

This approach:

- Effectively uses statistical information.
- Learning occurs only on non-zero elements in the word-sharing matrix X, which contains elements X_{ij} (the value of which is the number of times word j occurred in context i).

Also calculated:

$X_i = \sum_j^k x_{ij}$ is the number of words in the context.

$P_{ij} = P(i|j) = X_{ij}/X_i$ is the probability of word j appearing in context i.

To get started, you need:

- Consider a simple example that reveals connections between two words i and j.
- Their relationship can be tested by examining their co-occurrence with another word k.

The following patterns are distinguished:

- If k is more related to i, then P_{ik}/P_{jk} will have a value greater than 1.
- If k is more related to j, then P_{ik}/P_{jk} will have a value close to 0.
- If k is almost equally related to i and j, then P_{ik}/P_{jk} will have a value close to 1.

Generative neural networks are used in natural language processing to generate a sequence of tokens that depend on the input context and the already generated sequence of words.

The following main neural network models are used for sequence processing: recurrent neural network and Transformer neural network.

Recurrent neural networks are networks in which connections between elements form a directed loop.

A recurrent neural network sequentially processes input data using an internal memory that aggregates context and transfers it between iterations.

The recurrent neural network has the following modifications: LSTM (*long short-temp memory*) [30, 31] and GRU (gated recurrent units) [32], which improve the internal memory mechanism.

The transformer neural network, unlike the recurrent neural network, processes the input data not sequentially, but all at the same time.

A recurrent neural network uses an internal state (memory) to process sequential inputs, unlike conventional feedforward neural networks. With the help of this feature, it becomes possible to effectively process natural language, and form appropriate sentences.

There are many examples of recurrent neural network applications in natural language processing.

For example, machine translation, which simultaneously contains two models, which are recurrent neural networks.

One of them serves as an encoder, and the other as a decoder.

This model is called seq2seq [33].

LSTM is a recurrent neural network that processes a sequence of dynamic size

$$X = (x_1, x_2 \dots x_n),$$

adding new content to a block of memory with activation gates that control what information should be forgotten and what should be retained.

At each iteration t, the memory c_t and the hidden state h_t are updated.

The GTU (like the LSTM) contains enable gates that control the flow of information in the middle of the block but does not have a separate memory module.

The activation of output h_t at the time t is the linear interpolation between the result of the last iteration h_{t-1} and the activated candidate for output $\tilde{h}_t$:

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t,$$

where z_t is the so-called “update valve”, which updates the received information and is calculated by the formula:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

$\tilde{h}_t$ is calculated similarly to the component of a recurrent neural network:

$$\tilde{h}_t = tanh(Wx_t + U(r_t \odot h_{t-1}))$$

where r_t is the so-called “state clearing valve” and is calculated similarly to the “update valve”:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

The best common feature between GRU and LSTM is the update component, which is missing from a conventional recurrent neural network that simply replaces context with value.

This component is computed with the new input token and the previous hidden state.

Although both GRU and LSTM preserve the current context and add a new one on top of it.

A transformer is a neural network that avoids repetition, and its main component is an attention mechanism for creating a dependency between input and output [34].

With the help of the Transformer neural network, effective parallelization is carried out, which increases the quality and ease of learning.

The Transformer model in a neural network adheres to the encoder-decoder architecture. The encoding and decoding blocks contain the attention mechanism, normalization, and forward propagation network.

The encoder consists of n identical layers.

Each layer contains two sublayers. The first is a multilateral mechanism of attention, and the second is a fully connected network of forward propagation.

After each sublayer, the residual connection and its normalization are used.

The decoder consists of n identical layers, and a third layer is added to the two sublayers from the encoder.

This layer adds the context of the encoder to the decoder using a multi-attention mechanism.

Like the encoder, each sublayer has a residual connection and layer normalization.

Also, in the first sub-layer of multilateral attention, a mask is added to prevent the transition between positions (generated words).

The attention mechanism is the main component of the Transformer neural network.

Attention accepts as input "question," "key," and "value" in vector form.

The output is a weighted sum of input vectors, each of which is combined with its weight.

Multilateral attention consists of h attention. At the output, they are concatenated and passed through an additional level of weights.

The input vectors V, K, and Q are the same for all multi-attention sublayers, except for the layer that adds the encoder context to the decoder, where V and K will be the output of the encoder, and Q will be the current state of the decoder.

The transformer model in the neural network uses a conventional forward propagation neural network, which consists of two linear transformations and an activation function between them:

$\mathrm{FFN}(x) = \max(0, xW_1+b_1)W_2+b_2$

For more efficient work, you need to use trained embedding to represent (represent) words in vector n-space.

At the output of the decoder, a linear transformation is performed from the embedding size to the dictionary size, and softmax is applied to find the next word token until a custom token is generated.

To optimize the Transformer neural network, a dropout layer is used, which is applied to the outputs of each of the sublayers before performing the residual connection.

Also, the inputs to the encoder and decoder are additionally activated with the help of the discard layer.

3. Methods of Optimization of Neural Networks

There are many techniques to help improve neural networks, from updating weights to preventing overtraining.

The following methods can be distinguished:

- throwing away
- stochastic optimizer Adam [35]
- residual connection [36].

Neural networks with limited training data retrain quickly but show poor results even on the training set.

Therefore, many methods have been developed that try to solve this problem.

With the help of the optimization mechanism—the dropout method [37], neurons are extracted from the input or hidden layers.

When removed, neurons are thought to become temporarily inactive along with all their input and output connections.

In the simplest version, each neuron is stored with a fixed probability that is close to 0.5, and each neuron is independent of the others.

The Adam stochastic optimizer is used for gradient optimization of first-order stochastic objective functions using lower-order moment estimation.

The Adam stochastic optimizer is simple to implement, yet computationally efficient, requires little memory, and is invariant to sudden changes in gradients.

Residual connection is used for deep neural networks, which are difficult to train due to the constantly growing number of layers.

The main problem that arises in the training of neural networks is the rapidly occurring gradient explosion on large structures of neural networks.

To solve this problem, normalization layers (layer normalization, batch normalization, etc.) have been introduced, which help networks with dozens of levels to become similar at the time of stochastic gradient descent.

During the assimilation step, the problem of degradation occurs, where the desired accuracy is gradually reached and then drops sharply.

Therefore, it is possible to solve the problem of degradation by using the residual connection in the neural network.

Denoting the base representation as $H(x)$ for the nonlinear layers of the neural network the representation looks like

$F(x) = H(x) - x.$

Thus, the representation of $H(x)$ will look like this:

$F(x) + x,$

which is a residual compound.

To create a text simplification and normalization system, you need to perform the following actions:

- Pre-processing of the text, which standardizes the sentence, breaks it into tokens and adds special tokens.
- representation of the word in the form of a vector, which transforms the token into the corresponding vector space.
- use of a generative model that creates a text sequence depending on the received context.

Before performing the above actions, a text database (a database of text educational content of an information learning system or an information learning system with elements of intellectualization) is created with pairs:

<input text, desired result>.

The input text is separate sentences from various information sources that correspond to the topic of the selected subject area (theory of algorithms, cyber security, ontologies, artificial intelligence).

The desired result is a modified version of the input text, which is created with the help of experts, following the following rules:

- Combined words (groups of words) are converted into an abbreviation, for example:
  - “neural network”—“NN”
  - “Turing machine”—“MT”
  - “subject area”—“SA”.
- Removal of phrases (words, groups of words) that do not significantly affect the content of the text, for example:

“The first step in the study of neural networks was made in 1943, when an article by neurophysiologist Warren McCulloch and mathematician Walter Pitts was published, devoted to artificial neurons, as well as the implementation of a neural network model using electrical circuits”

turns into

“The study of NN began with an article devoted to the implementation of NN using electrical circuits”.

- Exclusion of redundant clarifications in the educational text, for example:

“This science (meaning artificial intelligence) is related to neuroscience, including cognitive neuroscience, systems neuroscience, computational neuroscience” becomes “This science is related to neuroscience.”

The larger the text database (the database of the text educational content of the information learning system or the information learning system with elements of intellectualization), the more the neural network will recognize patterns and find various connections.

The creation of such a database is one of the most important stages because its quality directly affects the output of natural language processing of educational content.

Natural language text preprocessing is an important part of preparing the data for training the Transformer neural network.

This network works with the content of an informational learning system or an informational learning system with elements of intellectualization.

The stages of processing depending on the goals of text processing (educational content) of the informational learning system.

To simplify and normalize the text (in particular, the answers of students given in natural language), the following steps should be taken to standardize this text:

- Transfer of all alphabetic characters in the words of the natural language text (for example, the student’s answers) to the lower case of writing (that is, all capital letters are removed).
- Reduction (but not abbreviations) are translated into full words, for example, words such as “y.” and “etc.”, are reduced to the words “year”, “and the like”, “and so on” respectively.
- Replacement of various variants of quotation marks with one type of “double quotation marks” (depending on the language, these quotation marks have different coding).
- Insertion before and after individual characters of the English language, which is a separate letter, an additional special character <t> (there may also be a space < >) to check whether it belongs to an indefinite article (if this character is not an article, then it is excluded from consideration and simplify the text).
- Inserting before and after individual characters of the Ukrainian language, which is a separate letter, an additional

special character <t> (there may also be a space < >) to check whether it belongs to the group of prepositions (if this character is not a preposition, it is excluded from consideration and simplify the text).
- Insertion before and after individual characters, which is a number (single digit) of an additional special character <t> (there may also be a space < >).
- Replacing several consecutive spaces in the text with one space.

After standardizing the text, sentences of the answer (or educational content) are divided into tokens with the help of a special symbol (maybe a space), which is placed during the standardization and normalization of the text.

At the same time, the so-called “empty elements” among the received tokens should be removed.

All sequences of additionally received tokens begin with the special character <s> and end with the special character </s> to mark the beginning and end of a sentence (a period-ended piece of educational content text), respectively.

All sentences after standardization and normalization (an ordered collection of tokens from which these sentences are created) are used to train the corresponding neural network.

This model is used in natural language processing, and all sentences form a dictionary in which the total number of each token in the training sets is counted.

Tokens are sorted by their calculated number (from the largest number to the smallest), acquire their index relative to the occupied position, and are deleted if their number is less than the specified minimum number.

Vector representation of the word (embedding).

The use of embedding helps to increase the speed and accuracy of neural network training, which plays the role of the main model of the natural language processing process.

The main tasks of creating a vector representation of words in the text of a student’s answer given in natural language:
- Get rid of the one-hot vector during the design of the main model [22].
- Approximation of vectors of words that occur in the context of each other.
- Finding an associative connection between individual pairs of words (word forms, lexemes, etc.).

Transformer model in the neural network model as an input, the encoder accepts a sequence of words of the input text, which has already undergone primary pre-processing.

The decoder accepts the coded text generated in the system (based on the use of a trained neural network) and the context provided by the encoder.

The initial value of the encoded text is a special token <s>) and the context is provided by the encoder.

Each newly generated token is appended to the generated text to be reused by the decoder until the token equals the special character </s>.

The encoder consists of three consecutive sublayers.

Each sublayer contains an attention block and a fully connected neural network.

After these sub-layers, the main neural network, according to which processing of natural language in the information learning system with elements of intellectualization of the system takes place, contains a layer of discarding and a layer in which the residual connection and/or normalization takes place.

The decoder, like the encoder, contains n linear sublayers, each of which has a similar structure to the encoder sublayers, but with certain differences.

The last vector of the generated matrix is taken at the output of the decoder.

This vector undergoes a linear transformation and a softmax activation function to form a one-hot vector that defines the next token in the sequence.

Normalization and simplification of natural language text (in particular:
- Student answers.
- The educational content of the description of completed individual tasks.).

The information educational system with elements of intellectualization can be performed under the following conditions:
- The source text has a smaller or equal size than the input text (the size is the

number of tokens in the sentences of the corresponding natural language text).
- Replacement of terms (in particular, their synonymization).
- Replacing slang words with their literary counterparts.
- Replacing slang words (including Internet slang words) with their literary counterparts.
- Replacement of a group of words with corresponding constant abbreviations.
- Addition/deletion/modification of text (for example, an answer presented in natural language) contained in the database of text educational content of the information educational system with elements of intellectualization.
- Simplification of the selected fragment of the text (including standardization, and normalization).
- Taking into account when simplifying the text presented in natural language, contexts either already available in the database, or generated using a neural network and taking into account the dictionary of contexts.

Simplification of the text in the informational educational system with elements of intellectualization involves:

- Preparation of educational data.
- Using a neural network model for teaching embedding.
- A neural network model for text generation.
- Services for managing simplification processes.

An extension that uses and interacts with the simplification system provides:

- Adding text to the database of text educational content of the information educational system with elements of intellectualization.
- Simplification of the selected text from the database (according to the above rules).
- Simplification of the entered text, taking into account possible contexts and the corresponding dictionary of contexts.

The collection of training data of the neural network of the information training system with elements of intellectualization involves:

- Primary text processing (both manual and automatic).
- Text fragmentation (breaking into separate sentences).
- Sentence tokenization (breaking the text into separate tokens).
- Adding special tokens (sentence start (<ts>) and sentence end (</ts>)) that facilitate interaction with the neural network.
- Creation of a dictionary (sampling and counting of tokens for all tokenized sentences is performed, the dictionary is organized in descending order by number).

Tokenized sentences and created (generated from the texts of the selected subject area) dictionary will be used by embedding and sentence generation systems.

Embedding involves:

- Loading of tokenized sentences, dictionaries, and trained vectors (if the training was already performed).
- Generation of a unigram table from the dictionary.
- Neural network training using embedding.
- Preservation of the so-called "learned vectors" in the neural network, which will then serve as "standards" in the processing of natural language texts; and learning.
- Launching the API server for embedding testing (checking the distance to a specific token and finding the nearest tokens).

The operation of the Transformer model in the neural network includes, in particular:

- Loading of tokenized sentences, trained vectors, and trained model, if training was already performed.
- Launch of the API server for text normalization and simplification using a generative model.
- Management of training with a specified number of epochs and to a specified accuracy, testing (on a training set), and simplification of the entered text.

When processing educational data, where raw text is accepted as input, the following are created:

- A file with a list of input-tokenized sentences (these sentences will be accepted by neural networks as input during training).

- A file with a list of output tokenized sentences (these sentences will be used by neural networks to approximate internal parameters by comparison with input sentences).
- Dictionary that stores all tokens and their number in JSON format.

The sources for teaching embedding are, in particular, files from:

- Input weights (stored in text format, where each token is a new line in the file, the first element of the line is the value of the token (word, number, sign, etc.), the other elements of the line are a vector representation of the token, where the values are arranged in order).
- Output weights (stored in text format, where each token is a new line in the file, the first element of the line is the value of the token, and the other elements of the line are a vector representation of the specified token (the values are in order).

The weights are used in training (as a communication between the hidden and output layers) and to form a vector representation of the word).

In pairs:

<token, vector representation>,

that were trained using the word2vec embedding model.

In the informational educational system with elements of intellectualization, a trained neural network (Transformer deep neural network) is used to process text presented in natural language (it can be a student's answer to a question or a description of a completed individual task).

Pre-processing of the text, which is performed by the neural network training data processing service, normalizes the text for its further use in the informational training system with elements of intellectualization.

The preparation of training data is an important step because it starts all the processes of training the Transformer model in a neural network in the information learning system with elements of intellectualization.

The more quality data, the better the result will be at the output after processing the natural language text.

The sentence is divided into tokens using a special delimiter symbol, and a tokenized sentence is formed, that is an array of tokens.

The token library of the information learning system is formed from these tokenized sentences.

The number of tokens in the training set of the neural network is counted and sorted in the descending direction.

With the help of a dictionary, a unigram table of the distribution of word noise is created (so that more words are more often included in the negative sample).

The neural network Transformer learns to find various patterns and connections in the training sample, which are used during training (both in training the neural network and in further processing of natural language texts).

To train the Transformer neural network, the following are used:

- pre-prepared sentences (from the educational content of the online course, which is supported by an informational educational system with elements of intellectualization).
- generated vector representation for each known token.

The Transformer model in neural networks has, in particular, the following characteristics:

- "question" and "key" in the attention layers have a dimension of 150 n, and "value"—15 n.
- the neural network has hidden layers.
- the dimension of the hidden layers is 2k n ($6 \le k \le 9$).
- n—the number of tokens in the input sentence.
- n is not a constant value, but it can change dynamically.
- discarding layers are used, which with a probability of 0.1 turn off one or another (unimportant, redundant) neuron, the absence of which does not affect the quality of natural language text processing, but reduces the time of this processing.

In the informational educational system with elements of intellectualization of learning control of the Transformer neural network.

This network is used in the processing of natural language texts, it is performed using the following actions:

- Training with a given number of iterations.

- Testing the model (transformer neural network) on the training set (training sample).
- Generation of the original text from the entered (input) text, taking into account the existing context.
- Preservation of the learned model in the knowledge base of the informational educational system with elements of intellectualization.
- Using the trained model for:
  - simplification of the text
  - text normalization
  - text standardization.
- Expansion (if necessary) of the system dictionary.
- Expansion (if necessary) of the library of contexts.

After training a neural network, an informational learning system with elements of intellectualization that uses it can:

- Recognize the contexts it (neural network) has studied.
- Generate new sentences.

Using different levels of abstraction to recognize certain patterns and relationships that help generate new sentences.

4. Identifying Entities in Text

An important element of natural language text analysis is the identification of named entities in the text (*Named Entity Recognition, NER*) [38]:

- Names of people.
- Geographical names.
- Names of academic disciplines of the university.
- Monetary amounts.
- Professional terms (concepts) of a certain subject area, etc.

To meet the unique needs of an information training system with intellectualization elements, it is necessary to adapt NER, for example, to:

- Classes of learning tasks.
- Learning objectives.
- Subject areas of training courses.
- Language of presentation of educational content.

Modern solutions for identifying entities in natural language texts use two technologies:

- Dictionary selection of named entities NER.
- Machine detection of NER entities.

For example, the task is to determine the essence of “programming language” and highlight its name.

If an information learning system with intellectualization elements is being developed for specialty 121 “Software Engineering” or 122 “Computer Science”, then:

- The list of programming languages is finite.
- To determine the language in a request, it is easier to use a special dictionary in which the authors of training courses describe all existing or studied programming languages at the university.

Dictionaries also allow you to set the synonymy of names (programming language C++—“pluses”) and normalize the name found in the request to form a query to the database of the information learning system with elements of intellectualization.

For the NER task, in addition to the neural network, an approach based on language rules (rule-based) is also used, in which students communicate with an information learning system with elements of intellectualization.

In addition, several libraries are combined in one convenient API, which allows solving basic NLP problems for the Ukrainian (English) language, for example, as [39, 40]:

- Tokenization.
- Segmentation of offers.
- Word embedding.
- Morphology marking.
- Lemmatization.
- Normalization of phrases.
- Parsing.
- NER tags.
- Extraction of facts.

Based on modern technologies for identifying data from natural language texts of various natures, modern solutions are implemented for a wide range of problems solved using an information learning system with elements of intellectualization, in particular:

- Registration of a student’s request (for example, to a course database and/or teacher), a student’s response, and a

description of the solution to an individual assignment.
- Highlighting entities in a student's request, his answer to a question, and a description of the solution to an individual task.
- Prompt extraction of educational content (at the student's request) from the database of additional educational material of the course (including from the Internet via links).
- Comparison of documents (for example, different versions of the same answer, a piece of educational content).

Classification of texts (classification of requests, answers, and appeals, determining the importance of a request and searching for a ready-made answer, counseling students, etc.).

5. Understanding the Text

Technologies and systems for understanding natural language texts (*Natural Language Understanding, NLU*) link together entities (concepts, terms, keywords, etc.) that appear in the text with specific relationships [41].

Entities and connections between them form a unified system of what is described in the text.

This is the so-called "machine understanding" of the text, which allows, for example, to give correct answers to questions related to this description.

Important for NLU is the resolution of frequently occurring linguistic constructions, such as:

- Ellipsis—restoration of missing words (for example, in the sentence "The speed of program execution on computer P1 increased by 20%, and on P2—by 30%," the words "computer" and "increased" are missing).
- Anaphora—a pronoun replaces a noun.
- Homonymy—a polysemantic interpretation of a word (for example, a word such as "course" can refer to a subject area:
  - "finance" (exchange rate).
  - "navigation/shipping" (ship's course at sea).
  - "training" (course of educational material, course of study).

Processing natural language text entities involves turning data into knowledge.

Extracting entities in this case involves the use of an ontological approach.

This means that during the analysis of natural language text, a semantic hierarchy of concepts is built.

Thus, an important goal of processing texts in natural language is not just training the appropriate neural network, but also finding certain regularities present in these texts, connections (relationships) between concepts (terms, entities) and in them.

After training, the neural network (a natural language text processing model) even sometimes finds ways to further simplify the training data (output).

That is, after training, the neural network can find flaws in natural language text processing when filling a text database (a database of the textual educational content of an informational learning system or an informational learning system with elements of intellectualization).

It is also possible to achieve a higher level of abstraction of natural language text generation by feeding the trained neural network a text that it has already simplified.

It should be noted that the main context is almost unchanged with such processing.

And if you continue this chain of simplification, then the neural network will gradually lose the main context and reach the following logical conclusions:

- looped simplification (simplification stops happening, i.e. changes stop).
- generating texts with random words.

During the training of the Transformer neural network, a pattern was deduced regarding the dimensionality of the layer of multidirectional attention.

Scales have dimension $n \times l$,

where

n is the size of the incoming message (dynamic parameter).

l is a defined parameter of the linear transformation.

The parameter l affects the number of so-called "found" connections, and this number depends on the size of the embedding.

The following patterns are distinguished:

- If l is larger than the size of the embedding, then the neural network (natural language text processing model) will "invent" new connections.

- If *l* is smaller than the embedding size, then the neural network (natural language text processing model) may miss certain connections.
- If *l* is equal to the embedding size, then the trained neural network performs best.

This pattern can be compared to the resizing of an image when neural networks learn to recognize an image.

When the image is enlarged, its additional pixels are formed from those next to it, and if it is reduced, then certain pixels are removed.

6. Conclusions

The use of a neural network, according to which texts are processed in natural language, can also be used to solve other problems, in particular, such as:

- Simplification of documents.
- Unification of license agreements.
- Simplification of the texts of Internet pages (opened in the browser), which can be provided in educational courses, as a link to additional material.
- Simplification of the texts of PDF documents (opened in the browser), which can be provided in training courses as links to additional material.

The approach to natural language text processing proposed in the paper involves normalization and simplification of the text.

At the same time, the Transformer model in a neural network is used for text processing, with the help of which good results were obtained.

A trained neural network can demonstrate good results:

- On sentences that she once met ("saw").
- Sentences that are similar in context to those that she already knows how to process.
- Sentences with "damaged" (for example, incompletely spelled or misspelled) words.

The work included: various methods of neural network optimization that prevent cases of gradient explosion, gradient damping, overtraining, etc. are considered and analyzed.

Various natural language processing methods are considered.

References

[1] O. Romanovskyi, et al., Prototyping Methodology of End-to-End Speech Analytics Software, in: 4th International Workshop on Modern Machine Learning Technologies and Data Science, vol. 3312 (2022) 76–86.

[2] I. Iosifov, et al., Transferability Evaluation of Speech Emotion Recognition Between Different Languages, Advances in Computer Science for Engineering and Education 134 (2022) 413–426. doi:10.1007/978-3-031-04812-8_35

[3] I. Iosifov, O. Iosifova, V. Sokolov, Sentence Segmentation from Unformatted Text using Language Modeling and Sequence Labeling Approaches, in: 7th International Scientific and Practical Conference Problems of Infocommunications. Science and Technology (2020) 335–337. doi: 10.1109/PICST51311.2020.9468084.

[4] I. Iosifov, et al., Natural Language Technology to Ensure the Safety of Speech Information, in: Workshop on Cybersecurity Providing in In-formation and Telecommunication Systems, vol. 3187, no. 1 (2022) 216–226.

[5] O. Iosifova, et al., Analysis of Automatic Speech Recognition Methods, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 2923 (2021) 252–257.

[6] O. Romanovskyi, et al., Automated Pipeline for Training Dataset Creation from Unlabeled Audios for Automatic Speech Recognition, Advances in Computer Science for Engineering and Education IV, vol. 83 (2021) 25–36. doi:10.1007/978-3-030-80472-5_3

[7] O. Iosifova, et al., Techniques Comparison for Natural Language Processing, in: 2nd International Workshop on Modern Machine Learning Technologies and Data Science, vol. 2631, no. I (2020) 57–67.

[8] An Introduction to Transformer Models in Neural Networks and Machine Learning (2023). URL: https://www.linkedin.com/pulse/intro

duction-transformer-models-neural-networks-machine-learning

[9] OpenAI ChatGPT. URL: https://chat.openai.com/

[10] OpenAI DALL-E. Generating Digital Images from Natural Language Descriptions. URL: https://openai.com/dall-e-2

[11] How DALL-E 2 Actually Works. URL: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

[12] Grammarly. Generative AI assistance for Writing. URL: https://www.grammarly.com/

[13] Rytr. AI writing assistant. URL: https://rytr.me

[14] How ChatGPT actually works. URL: https://www.assemblyai.com/blog/how-chatgpt-actually-works/

[15] Supervised Fine-tuning: customizing LLMs. URL: https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3

[16] S. Pitis, Failure Modes of Learning Reward Models for LLMs and other Sequence Models, in: The Many Facets of Preference-Based Learning (2023).

[17] Data Right: Building a Successful Dataset From the Ground Up. URL: https://www.grammarly.com/blog/engineering/high-quality-nlp-datasets/

[18] Experimenting with GECToR: Research into Ensembling and Knowledge Distillation for Large Sequence Taggers. URL: https://www.grammarly.com/blog/engineering/experimenting-with-gector/

[19] CLIP: Connecting text and images. URL: https://openai.com/research/clip

[20] ChatGPT Architecture: Will ChatGPT Replace Search Engine? URL: https://opchatgpt.com/chatgpt-architecture-will-chatgpt-replace-search-engine/

[21] S. Ghannay, et al., Word Embedding Evaluation and Combination, Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA) (2016) 300–305.

[22] One-Hot Encoding in NLP. URL: https://www.geeksforgeeks.org/one-hot-encoding-in-nlp/

[23] S. Bagui, et al., Machine Learning and Deep Learning for Phishing Email Classification using One-Hot Encoding, J. Comput. Sci. 7(17) (2021) 610–623. doi: 10.3844/JCSSP.2021.610.623.

[24] K. Church, Word2Vec, Nat. Lang. Eng. 1(23). (2016) 155-162, doi: 10.1017/S1351324916000334.

[25] T. Pickard, Comparing word2vec and GloVe for Automatic Measurement of MWE Compositionality, in: Joint Workshop on Multiword Expressions and Electronic Lexicon (2020) 95–100.

[26] Continuous Bag of Words (CBOW) in NLP (2023). URL: https://www.geeksforgeeks.org/continuous-bag-of-words-cbow-in-nlp/

[27] Skip-Gram: NLP Context Words Prediction Algorithm (2019). URL: https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c

[28] Negative Sampling vs Hierarchical Softmax (2021). URL: https://flavien-vidal.medium.com/negative-sampling-vs-hierarchical-softmax-462d063dfca4

[29] J. Pennington, R. Socher, C Manning, GloVe: Global Vectors for Word Representation, Conference on Empirical Methods Natural Language Processing, Association for Computational Linguistics (2014) 1532–1543. doi: 10.3115/v1/D14-1162.

[30] Long Short-Term Memory Network (2021). URL: https://www.sciencedirect.com/topics/computer-science/long-short-term-memory-network

[31] J. Cheng, L. Dong, M. Lapata, Long Short-Term Memory-Networks for Machine Reading, Conference on Empirical Methods In Natural Language Processing, Association for Computational Linguistics (2016) 551–561. doi: 10.18653/v1/D16-1053.

[32] L. Pasa, A. Sperduti, Pre-training of Recurrent Neural Networks via Linear. Autoencoders, 27th International Conference Neural Information Processing System (2015) 3572–3580.

[33] A. Takezawa, How to Implement Seq2Seq LSTM Model in Keras (2019). URL: https://towardsdatascience.com/

how-to-implement-seq2seq-lstm-model -in-keras-shortcutnlp-6f355f3e5639

[34] A. Vaswani, et al., Attention is All you Need, 31st International Conference Neural Information Processing System (2017) 6000–6010.

[35] D. Kingma, J. Ba, Adam: a Method for Stochastic Optimization, 3rd International Conference Learning Representations (2015). doi: 10.48550/arXiv.1412.6980.

[36] K. He, et al., Deep Residual Learning for Image Recognition, 29th IEEE Conference Computer Vision and Pattern Recognition (2016) 770–778. doi: 10.1109/cvpr.2016.90.

[37] N. Srivastava, et al., Dropout: a Simple Way to Prevent Neural Networks from Overfitting, J. Mach. Learning Res. 1(15) (2014) 1929–1958.

[38] Named Entity Recognition: The Mechanism, Methods, Use Cases, and Implementation Tips (2023). URL: https://www.altexsoft.com/blog/name d-entity-recognition/

[39] J. Eisenstein, Natural Language Processing. (2018). URL: https://cseweb.ucsd.edu/~nnakashole/ teaching/eisenstein-nov18.pdf

[40] M. Tarwani, S. Edem, Survey on Recurrent Neural Network in Natural Language Processing, Int. J. Eng. Trends Technol. 6(48) (2017 301–304. doi: 10.14445/22315381/IJETT-V48P253.

[41] A. Gillis, Natural Language Understanding (NLU) (2023). URL: https://www.techtarget.com/searchent erpriseai/definition/natural-language-understanding-NLU