https://doi.org/10.33271/nvngu/2025-1/090

A. V. Miakenkyi*,
orcid.org/0000-0002-4141-001X,
M. O. Aleksieiev,
orcid.org/0000-0001-8726-7469,
S. M. Matsiuk,
orcid.org/0000-0001-6798-5500

Dnipro University of Technology, Dnipro, Ukraine
* Corresponding author e-mail: miakenkyi.ar.v@nmu.one

RESEARCH ON THE EFFECTIVENESS OF USING LSTM ARCHITECTURE IN MODELING THE COGNITIVE PROCESS OF RECOGNITION

A person's ability to recognize and separate the meanings of words when working with textual information refers to the higher cognitive functions of the brain, in particular to the cognitive process of recognition. The solution to the problem of extracting the meaning of words in text is related to the tasks of natural language processing (NLP) and is called word sense disambiguation (WSD). There are many approaches to solving WSD, particularly using neural networks.

Purpose. Creation and analysis of the bidirectional LSTM neural network architecture for solving the WSD problem in the Ukrainian language.

Methodology. One of the modern approaches to solving the WSD problem is the use of LSTM models – a type of recurrent architecture of neural networks that allows you to capture long-term dependencies when modeling sequences. To determine the effectiveness of using this architecture, two neural networks were built during the study: the classic LSTM architecture and its improved version – Bi-LSTM. As part of the study, a data set based on the SUM dictionary of the Ukrainian language was also created. The implemented models were trained on the generated data set, after which a comparative analysis of the obtained data was performed.

Findings. The analysis of the results of the accuracy of the built models made it possible to determine the efficiency of the neural network built according to the Bi-LSTM architecture. The obtained accuracy results are equal to 73 % for the LSTM model and 83 % for Bi-LSTM, respectively, which is due to the presence of an additional layer in the Bi-LSTM model, which provides the opportunity to take into account the full context of the word in the given text.

Originality. The paper establishes the effectiveness of the neural network model built on the Bi-LSTM architecture for solving the WSD problem in texts in Ukrainian in comparison with the classical LSTM architecture.

Practical value. As a result of the work, a model is proposed that allows solving the problem of eliminating the ambiguity of words in the Ukrainian language, and which can be used in text processing tasks, in particular for modeling the cognitive process of understanding.

Keywords: *cognitive modelling, cognitive process, NLP, WSD, LSTM, Bi-LSTM, pymorphy2, stanza, tensorflow*

Introduction. The cognitive process of understanding belongs to the highest level of cognitive functions of the brain and is used by the human brain to identify objects that surround it. Identification of objects, words, or external stimuli occurs by retrieving stored information from memory and comparing it with information from sensory inputs [1]. For example, the ability of the human brain not only to recognize written words, but also to distinguish and understand the meanings of words when they can be used in different contexts is special, when processing textual information.

Solving the task of recognizing the meaning of words in the text belongs to the tasks of natural language processing (NLP) and is called word sense disambiguation (WSD). The task aims to automatically determine which of the possible meanings of a word is used in a given context.

Creating a model that is adapted to work with texts in Ukrainian is an actual problem, since most solutions for the WSD task are developed for English and other common languages. However, the Ukrainian language has its own specific features that require adapted approaches. For example, the grammatical structure, word formation, and semantic relations in Ukrainian may differ significantly from other languages, making the adaptation of existing models a difficult task.

Also, Ukrainian, like many other languages, is rich in polysemous words, i.e. words that have more than one meaning. For example, the word "head" can be used to mean a part of the body, as well as to define a person who is in charge of something. The ability to automatically distinguish between these meanings is important when creating high-quality systems for modeling the cognitive process of recognition in tasks related to textual information.

This paper presents an analysis of the effectiveness of models based on the LSTM architecture, and its improved version – Bi-LSTM, for solving the WSD problem, on a dataset formed with the help of the SUM Ukrainian language dictionary of the Ukrainian Language and Information Fund of the National Academy of Sciences of Ukraine.

Related works. Over the past decades, various approaches to solving the WSD problem have been proposed. Among the main approaches are supervised learning, knowledge-based approaches, and unsupervised learning.

Early studies used the support vector method (SVM) to classify word meanings using a set of features such as POS and surrounding words [2]. Another approach to solving the problem is a method based on graph data [3]. The authors propose a method for distinguishing word meanings using random walks on the knowledge graph. This method applies structural information from lexical databases such as WordNet and uses random walk algorithms to estimate the probability of different word meanings in context.

The main disadvantages of the above-mentioned approaches are the orientation of their models to complex and language-specific functions and resources, as well as difficulties in determining the meaning when the same word is used in different senses within the same context.

Recent studies have shown the effectiveness of using the long-short term memory (LSTM) architecture in solving the WSD problem [4]. LSTM is a type of recurrent neural network architecture that allows capturing long-term dependencies when modeling sequences. In addition to the basic structure of a recurrent network, which includes input, output, and hidden layers, LSTM has a more complex structure with additional memory cells and gateways that allow it to selectively remember or forget information from previous time steps. This type of architecture is used in other NLP tasks such as machine translation [5], speech recognition, and other sequential modeling tasks.

Further research has led to the appearance of an improved version of the architecture – bi-directional LSTM or Bi-

LSTM [6]. An important difference from a traditional LSTM is that the state of a bi-directional network at each time step consists of the state of two LSTMs, one moving forward, and the other moving backward. For the WSD task, this means that the model can store long-term information not only about the previous but also about the next words around the target word in the context, which in many cases is absolutely necessary for correct sense classification.

Depending on the task for which the WSD model is built, both the classical LSTM model and its improved version are used, which makes it relevant to study the effectiveness of these models for recognizing the meaning of words in Ukrainian texts.

Purpose. The aim of this paper is to develop a model architecture for solving the word sense disambiguation problem in the context of the Ukrainian language using the bidirectional LSTM architecture, as well as to analyze the effectiveness of the developed architecture in comparison with the classical LSTM architecture.

LSTM architecture. As mentioned above, LSTM is a recurrent neural network that overcomes the problem of storing long-term dependencies faced by conventional RNNs. The LSTM architecture consists of a cell whose state determines the current long-term memory of the network, a hidden state, which is the output of the network from the previous step, and the current input of the network. The regulation of incoming information at each step of the network is regulated by three gateways: the forgetting gateway, the input gateway, and the output gateway. The network architecture is shown in Fig. 1.

The LSTM works sequentially starting from the forgetting gateway, which is provided with the current input vector as well as the hidden state from the previous step. The forget gate decides what information to discard from the cell state. It uses the input vector and the previous hidden state to generate a number between 0 and 1 for each number in cell state c_{t-1}.

$$f_t = (W_f[h_{t-1}, x_t] + b_f),$$

where W_f denotes the weighting matrix; h_{t-1} is the hidden state of the model from the previous step; x_t is the current input data of the model; b_f is the bias value.

In the next step, the input gate decides what new information to store in the cell state. It has two parts. A sigmoid level that defines the values to be updated, and a tanh level that creates a vector of new candidate values that can be added to the state.

$$i_t = (W_i[h_{t-1}, x_t] + b_i);$$

$$C_t^{\sim} = \tanh(W_C[h_{t-1}, x_t] + b_C),$$

where W_i and W_C denote the weight matrix of the input gate and cell state, respectively; h_{t-1} is the hidden state of the model from the previous step; x_t is the current input data of the model; b_i, b_C are the values of biases.

The previous state of the C_{t-1} cell is then updated to the new state of the cell by combining the two layers. The old state of the cell is multiplied by the forget gate to forget the data. The new candidate values are then added to the new cell state.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C_t^{\sim},$$

where f_t denotes the value of the forgetting gate; C_{t-1} is the previous state of the memory cell of the model; i_t is the value of the input gate of the model; $C_t^{\sim}$ is the vector of candidate values for the new state of the memory cell.

The last step is the output gate, which determines the new hidden state for the network.

$$o_t = (W_o[h_{t-1}, x_t] + b_o);$$

$$h_t = o_t \cdot \tanh(C_t),$$

where W_o denotes the weight matrix; h_{t-1} is the hidden state of the model from the previous step; x_t is the current input of the model; b_o is the offset value; C_t is the state value of the memory cell.

Bi-LSTM architecture. The Bi-LSTM model is a combination between a bi-directional recurrent network (BiRNN) and LSTM. Bi-LSTM is a sequence model that contains two levels of LSTMs, one for processing input data in the forward direction and the other for processing in the reverse direction. The architecture of the model is shown in Fig. 2.

A feature of this architecture is the ability to process data in both directions, which makes it possible to better understand the relationship between sequences (for example, to take into account information about the next and previous words in a sentence relative to the target word). Hidden model states for the forward and backward layers are calculated as follows.

$$H_t^{Forw} = A \cdot (X_t \cdot W_{xh}^{Forw} + H_{t-1}^{Forw} \cdot W_{hh}^{Forw} + b_h^{Forw});$$

$$H_t^{Back} = A \cdot (X_t \cdot W_{xh}^{Back} + H_{t-1}^{Back} \cdot W_{hh}^{Back} + b_h^{Back}),$$

where W_{xh}^{Forw} and W_{xh}^{Back} denote the weight matrix for the forward and backward inputs, while W_{xh}^{Forw} and W_{xh}^{Back} denote the weight values for the forward and backward hidden states from the previous step H_{t-1}^{Forw} and H_{t-1}^{Back}. b_h^{Forw} and b_h^{Back} denote the offset values for the forward and backward states. A denotes the hidden layer activation function. The full hidden state H_t can be calculated by combining H_{t-1}^{Forw} and H_{t-1}^{Back}.

Finally, the initial state can be calculated using H_t.

$$O_t = H_t \cdot W_o + b_0,$$

where W_o and b_o are the values of the weight matrix and bias in the output layer [7].

Methodology. As mentioned above, the WSD task is to find the correct meaning of a given word in a given context. In this paper, we analyze the solution of the WSD task by supervised learning of two LSTM models – bidirectional and tradi-

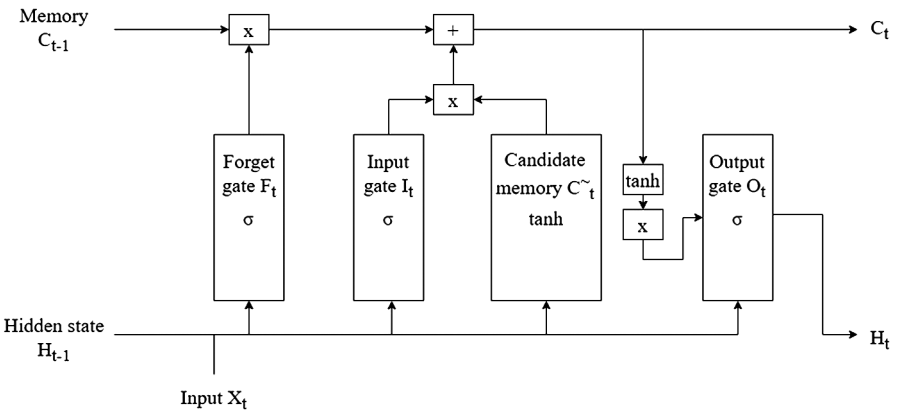

Fig. 1. LSTM Architecture

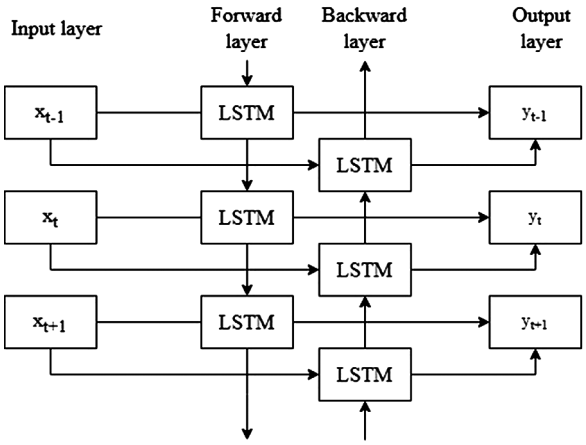

Fig. 2. Bi-LSTM architecture

tional ones. For this purpose, it is first necessary to obtain embeddings for the input context, its target word, and the target sense of this word.

After that, the models are fed with encoded words from the context surrounding the target word (for LSTM, these are the words that precede the target word, and for bidirectional LSTM, these are the words that both precede and follow the target word). Each input to the model is labeled with the value of the target word in the sentence. The output of the neural network is the predicted meaning of the target word in a given context.

A similarity calculation between the obtained predicted values and the real values from the test data set is necessary to evaluate the model's results. Depending on the type of task, methods for evaluating the obtained word embeddings are divided into two classes: intrinsic evaluation and extrinsic evaluation [8].

Intrinsic evaluation methods are designed to test word embeddings on specific, isolated tasks, calculating their qualities, such as semantic or syntactic relationships, regardless of their performance when used in subsequent tasks. Such methods provide a quick, task-independent assessment of the ability of a vector space to capture the meaning and similarity of words, with the greatest attention being paid to the semantic similarity of simple lexical units, such as words and their meanings.

Extrinsic evaluation methods, on the other hand, are designed to evaluate the quality of word representations in complicated natural language processing tasks in which embeddings are part of a larger model. Unlike intrinsic methods, extrinsic evaluation methods allow one to determine the effectiveness of word embeddings in solving tasks, such as text categorization or sentiment analysis, taking into account the characteristics in the context of the components of one model or between several models. Although external evaluation methods are important for understanding the effectiveness of embedding integration, they have greater variability in terms of tasks and benchmarks compared to simpler internal methods.

This study employs a method for evaluating the semantic similarity of words, which belongs to the class of intrinsic evaluation methods. This attribute is the simplest to evaluate for vector representations. The idea of the method is that the distances between word vectors in the embedded space reflect the actual semantic similarity between these words. Depending on the task, there are two most popular strategies: finding the maximum similarity for a pair of words by semantic value, or MaxSim, and averaging the similarities between possible pairs of words, or AvgSim [9].

MaxSim strategy is used to identify the most similar words to a certain word and also to identify the most representative words in the context of a certain embedding. This approach is used in such tasks as forming semantic groups of words, identifying synonyms, or classifying words by context.

In turn, AvgSim evaluates the average similarity of embeddings of words in tasks where the general representation of the semantic space is important. For example, if you want to calculate how well embeddings represent a certain category of words. This strategy is used to categorize word representations to evaluate how evenly they are distributed in terms of semantic similarity in classification and clustering tasks.

To calculate the similarity of word embeddings, the cosine similarity metric is used [9]. The cosine similarity is a measure used to determine the similarity between two vectors in space, regardless of their dimension. This method is widely used in machine learning tasks, in particular in text analysis, for comparing documents, assessing the similarity between queries and search results, and clustering data. The similarity is measured between the predicted meaning of the target word and its true meaning. The semantic value with the highest cosine similarity is considered the predicted meaning of the word in a given context.

Word embedding. In the context of NLP, word embedding is a technique for representing words as vectors in a multidimensional space, where the distance and direction between vectors reflect the similarity and relationships between the corresponding words. Thanks to this representation technique, words can retain their semantic and syntactic information based on the context in which the word is used [11].

For the first time, the idea of using vector representations of words was applied in order to generalize and eliminate the dimensionality problem in large-scale language models that predicted the next word in the text [12]. A feature of the proposed approach was the projection of unprocessed word vectors onto the embedding layer before being sent to other layers of the network. Such embedding models obtained from language models of neural networks [13] are called prediction-based models.

Another type of word embedding models are models that rely on using word context matrices to get vector representations. Such models are called count-based models, because the creation of word embeddings is not done by training algorithms that predict the next word given its context, but by globally counting occurrences of the word-context in the corpus.

Examples of prediction-based models are CBOW and Skip-Gram [14]. These models are logarithmic with a two-stage training procedure. The main goal of CBOW is to predict a target word based on its context. SG, in turn, aims to predict each word in context using the target word. The algorithms are shown in Figs. 3 and 4 respectively.

The most popular count-based model is Global Vectors for Word Representation or Glove [15]. The basic idea of word representation is that the actual semantic information about a pair of words is encoded by the co-occurrence ratio in the entire word corpus. A word representation is created by maximizing

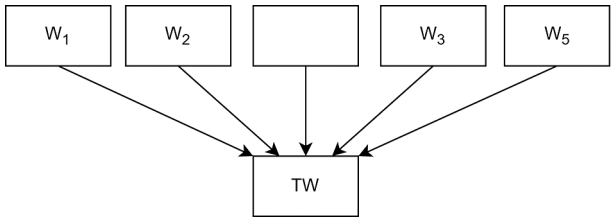

Fig. 3. CBOW algorithm

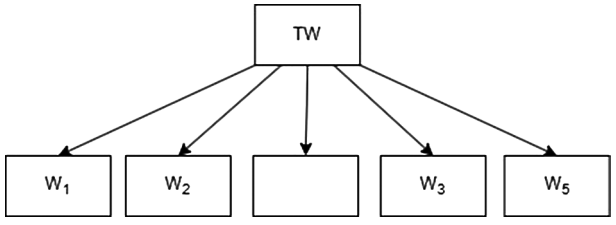

Fig. 4. Skip-Gram algorithm

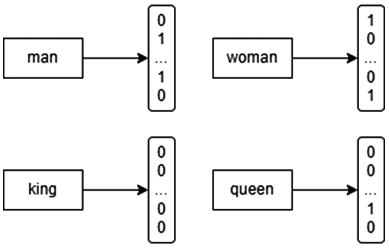

Fig. 5. Glove example of word vectors in three-dimensional space

the similarity of each pair of words, which is measured by the proportion of co-occurrences of this pair. In general, this model performs better than prediction-based models because it takes into account global statistical relationships between words in the entire corpus rather than in the local context [16]. This allows it to better understand the semantic relations between words and more accurately determine their meanings in different contexts.

In the context of this work, the initial embeddings of context, target word, and lexical meaning of the target word were obtained with the Glove model trained using the Ukrainian language corpus containing texts from fiction [17]. The obtained vectors were used to initialize the input layer of the LSTM neural network.

Dataset. To compare the work of LSTM models, a data set was created based on the "SUM" – the dictionary of the Ukrainian language, which is freely available through the Ukrainian Language and Information Fund of the National Academy of Sciences of Ukraine. The examples presented in the dictionary are formed using a wide list of sources, which includes fiction, folklore, journalism, popular science works, mass media articles, and Internet resources. At the time of the research, the version of the dictionary contains words, exclusively for the word "ПРЕФЕРЕНЦІЯ" (preference). An example of word with multiple meanings is shown in Table 1.

The dataset was constructed by retrieving each word from the dictionary, as well as its corresponding definition and examples. The resulting dataset contains data on 138,044 words. Further operations on the obtained data set include normalization of examples and their labeling.

Normalization of the examples consists in removing all punctuation from it, as well as reducing each word in the examples to its canonical form. For nouns, for example, it will be a word in the singular, nominative case. For this task, the morphological analyzer pymorphy2 was used, which is written in Python and provides models for the analysis of Ukrainian words [18].

After obtaining the normalized form of the word examples, the marking operation was carried out, which consists in replacing the target word in the example with a special symbol.

The target word's part of speech was also determined using a part-of-speech tagging (POS) algorithm. POS is a grammatical classification that typically includes verbs, adjectives, adverbs, nouns, etc. POS tags are an important natural language processing algorithm in tasks like machine translation, resolving word ambiguity, analyzing responses to questions, and more.

Traditional approaches to part-of-speech tagging include rule-based methods and statistical methods. The rule-based approach for POS tagging uses manually created rules to assign tags to words in a sentence. Linguistic features of the language, such as lexical, morphological, and syntactic information, are used to create these rules. The disadvantage of this approach is the complexity of creating a rule set with the involvement of experts, as well as tagging words that may have multiple meanings in the text. Statistical methods, on the other hand, use models trained on large annotated corpora to predict POS tags. An example of such a model is the hidden Markov model (HMM). This model is based on the Markov model, where the data structure is examined by analyzing transitions between hidden states. Unlike the classical Markov model, in HMM, the state is hidden by the observer, but the outcome, which depends on the state, is visible.

Another approach to part-of-speech tagging is using machine learning models. This approach utilizes machine learning algorithms like decision trees, support vector machines, or neural networks to learn patterns from the data. Special attention is paid to contextual information. The most popular ML algorithms used for POS taggers are naive Bayes, conditional random fields (CRF), Brill, and TnT.

In this work, the Stanza library, developed at Stanford University for the Python programming language, was used for tagging the parts of speech of target words in the dataset. It provides a wide range of tools for analyzing texts in various languages, including POS tagging for Ukrainian [19]. An example of the obtained dataset is shown in Table 2.

Model training. Training and testing data were generated from the dataset for model training. For this purpose, the dataset

Table 1

Multiple meanings of word "замок" ("castle") from "SUM" dictionary

Word	Meaning	Example
Замок (castle)	Укріплене житло феодала доби Середньовіччя з оборонними, господарськими, культовими і т. ін. будівлями, звичайно оточене високим кам›яним муром із кількома вежами. (Fortified housing of a feudal lord of the Middle Ages with defensive, economic, religious, etc. buildings, usually surrounded by a high stone wall with several towers.)	Пишний замок у пана Данила, нащадка древнього Жмайла!... Міцні, білі стіни, високі вежі, дубові двері, залізом куті, вузькі вікна ...; обнесений він мурами з баштами по кутках, з гарматами і гаківницями. (Magnificent castle of Mr. Danylo, a descendant of the ancient Zhmail!... Strong, white walls, high towers, oak doors, iron corners, narrow windows...; it is surrounded by walls with towers at the corners, with cannons and hook guns)
Замок (prison)	Тюремна будівля; тюрма (Prison building; prison)	Хан відправив його в Туреччину в подарунок султанові. Там деякий час він сидів в одиночці Семивежного замку – тюрмі для політичних злочинців та суперників султана. (Khan sent him to Turkey as a gift to the Sultan. There for some time he sat in solitary confinement in the Castle of the Seven Towers – a prison for political criminals and rivals of the Sultan)
Замок (lock)	Пристрій для замикання дверей у приміщеннях, дверцят шафи, а також скринь, шухляд (A device for locking doors in rooms, cabinet doors, as well as chests and drawers)	Вона закривала і відкривала чемодан, було чути, як стукає кришка і клацають замки. (She was closing and opening the suitcase, you could hear the lid banging and the locks clicking)
Замок (lock)	У деяких видах вогнепальної зброї – пристрій, признач. для здійснення пострілу. (In some types of firearms – a device designed to fire a shot.)	Але ти все-таки замислився? – сказав товариш Вовчик, провіряючи замки в своїй рушниці. (But you still thought about it? Comrade Vovchyk said, checking the locks in his gun.)

Table 2

Example of word “абажур” (“lampshade”) from dataset

Attribute name	Attribute value
ID	e36e8898-dc2d-4a02-9c9a-3ff9f4831391
Target word	АБАЖУР (lampshade)
Part of speech	NOUN
Context	Моя лампа під широким картоновим абажуром ділить хату на два поверхи – вгорі темний, похмурий, важкий; під ним – залитий світлом. (My lamp under a wide cardboard lampshade divides the house into two floors – upstairs is dark, gloomy, heavy; under it – filled with light.)
Normalized context	[мій, лампа, під, широкий, картоновий, абажур, ділити, хата, на, два, поверх, вгорі, темний, похмурий, важкий, під, він, залитий, світло] ([my, lamp, under, wide, cardboard, lampshade, divide, house, into, two, floor, upstairs, dark, gloomy, heavy, under, it, fill, light])
Tagged context	[мій, лампа, під, широкий, картоновий, <target>, ділити, хата, на, два, поверх, вгорі, темний, похмурий, важкий, під, він, залитий, світло] ([my, lamp, under, wide, cardboard, <target>, divide, house, into, two, floor, upstairs, dark, gloomy, heavy, under, it, fill, light])
Meaning ID	абажур_1
Meaning of word	Частина світильника, звичайно у вигляді ковпака, признач. для зосередження і відбиття світла та захисту очей від його впливу. (Part of the lamp, usually in the form of a cap, is intended for focusing and reflecting light and protecting the eyes from its influence.)

was filtered to remove all words with less than two examples. After that, the first instance of each word was selected for the training set, and the remaining instances were selected for the test set.

The obtained training set was used to train two LSTM and Bi-LSTM models, built with the difference that the Bi-LSTM implemented two layers of input data – forward input and backward input, while the conventional LSTM implemented only forward input [20]. The models were implemented using the tensorflow library in the Python programming language [21]. To optimize the learning process, an early stopping mechanism based on minimizing the loss function was implemented, along with Nadam (Nesterov Adam), an optimizer that combines the advantages of Adam and Nesterov Momentum and is used to accelerate convergence and improve the performance of deep learning models [22]. The parameters of the LSTM and Bi-LSTM models are shown in Table 3.

Results. The results of the models are presented in Table 4. For validation, a test sample was used, which was obtained from the generated dataset.

The obtained results showed weighted accuracy values of 73 and 83 % on the test data set for the LSTM and Bi-LSTM models, respectively. Optimization of accuracy was achieved with the help of dropout technology, which helps to reduce retraining of the model. The idea of the algorithm is to randomly “turn off” neurons at each stage of training. The tables show the corresponding dropout values and their effect on the accuracy of the model. Tables 5 and 6 show the corresponding dropout values and their effect on model accuracy.

Thus, the correct setting of the dropout parameter can improve the forecasting accuracy. If you do not use this technology at all, the results obtained may not be optimal.

The cosine similarity between the predicted meaning vector for a given word and the true meaning of the word in the context was chosen as the metric for calculating accuracy. The greater accuracy of the Bi-LSTM model is due to the use of the second backward layer of the model, which allows taking into account not only the context preceding the target word in the sentence, but also the context following it.

Conclusion. In this paper, we have built and compared models for solving the WSD problem, as one of the approaches to modeling the cognitive process of recognition, for recognizing the meaning of given words in Ukrainian texts. For this purpose, two neural networks based on the LSTM and Bi-LSTM architectures were built. To train the neural networks, a dataset was generated based on the SUM dictionary of the Ukrainian language. The results showed that the bidirectional LSTM has a higher accuracy than the unidirectional one, which is due to the use of a backward input layer that allows considering the full con-

Table 3

LSTM model parameters

Attribute	Value
Embedding size	300
LSTM units	300
Learning rate	0.002
Optimization algorithm	Nadam
Momentum	1

Table 4

Results of model training

Model name	Accuracy, %
LSTM	73
Bi-LSTM	83

Table 5

Results of LSTM model training at different values of dropout

Model	Dropout	Accuracy, %
LSTM	0	70
LSTM	0.1	71
LSTM	0.2	**73**
LSTM	0.5	72

Table 6

Results of Bi-LSTM model training at different values of dropout

Model	Dropout	Accuracy, %
Bi-LSTM	0	79
Bi-LSTM	0.1	82
Bi-LSTM	0.2	**83**
Bi-LSTM	0.5	81

text surrounding the word in the sentence. Thus, the architecture of the bidirectional LSTM allows us to effectively solve the WSD problem in the context of modeling the cognitive process of human comprehension in Ukrainian text processing. Further research on this topic may include improving the context labeling algorithm for the dataset, as well as experimenting with training data customization to improve the model's accuracy.

References.

1. Metzler, T., & Shea, K. (2011). Taxonomy of cognitive functions. *Proceedings of the 18th International Conference on Engineering Design*, 330-341. Retrieved from https://mediatum.ub.tum.de/1167203.

2. Pal, A. R., & Saha, D. (2015). Word Sense Disambiguation: A Survey. *International Journal of Control Theory and Computer Modeling*, *5*(3), 1-16. https://doi.org/10.5121/ijctcm.2015.5301.

3. Agirre, E., De Lacalle, O. L., & Soroa, A. (2014). Random Walks for Knowledge-Based Word Sense Disambiguation. *Computational Linguistics*, *40*(1), 57-84. https://doi.org/10.1162/coli_a_00164.

4. Popov, A. (2017). Word Sense Disambiguation with Recurrent Neural Networks. *RANLP 2017 – Student Research Workshop*. Shoumen, Bulgaria: Incoma Ltd. https://doi.org/10.26615/issn.1314-9156.2017_004.

5. Sundermeyer, M., Alkhouli, T., Wuebker, J., & Ney, H. (2014). Translation Modeling with Bidirectional Recurrent Neural Networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 14-25. https://doi.org/10.3115/v1/D14-1003.

6. Murugesan, R., Mishra, E., & Krishnan, A. H. (2021). Deep Learning Based Models: Basic LSTM, Bi LSTM, Stacked LSTM, CNN LSTM and Conv LSTM to Forecast Agricultural Commodities Prices. *Research Square*. https://doi.org/10.21203/rs.3.rs-740568/v1.

7. Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into Deep Learning*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2106.11342.

8. Shi, Y., Zheng, Y., Guo, K., Zhu, L., & Qu, Y. (2018). Intrinsic or Extrinsic Evaluation: An Overview of Word Embedding Evaluation. *2018 IEEE International Conference on Data Mining Workshops, 1*, 1255-1262. https://doi.org/10.1109/icdmw.2018.00179.

9. Reisinger, J., & Mooney, R. J. (2010). Multi-Prototype Vector-Space Models of Word Meaning. *North American Chapter of the Association for Computational Linguistics,* 1173-1182. Retrieved from https://aclanthology.org/N10-1013.

10. Gunawan, D., Sembiring, C. A., & Budiman, M. A. (2018). The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents. *Journal of Physics Conference Series*, *978*, 012120. https://doi.org/10.1088/1742-6596/978/1/012120.

11. Almeida, F., & Xexéo, G. (2019). *Word Embeddings: A Survey*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1901.09069.

12. Sun, S., & Iyyer, M. (2021). Revisiting Simple Neural Probabilistic Language Models. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5181-5188. https://doi.org/10.18653/v1/2021.naacl-main.407.

13. Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, *1*, 238-247. https://doi.org/10.3115/v1/p14-1023.

14. Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1301.3781.

15. Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 1532-1543. https://doi.org/10.3115/v1/d14-1162.

16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1310.4546.

17. *NER–models for MITIE. lang-uk*. Retrieved from https://lang.org.ua/en/models/.

18. Tmienova, N., & Sus, B. (2019). System of Intellectual Ukrainian Language Processing. *Selected Papers of the XIX International Scientific and Practical Conference "Information Technologies and Security"*, 199-209. Retrieved from https://ceur-ws.org/Vol-2577/.

19. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). *Stanza: A Python Natural Language Processing Toolkit for Many Human Languages*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2003.07082.

20. Kågebäck, M., & Salomonsson, H. (2016). *Word Sense Disambiguation using a Bidirectional LSTM*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1606.03568.

21. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, ..., & Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1603.04467.

22. Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. *Proceedings of the 4th International Conference on Learning Representations*, 1-4. Retrieved from https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ.

Дослідження ефективності використання архітектури LSTM при моделюванні когнітивного процесу розуміння

А. В. М'якенький, М. О. Алексєєв, С. М. Мацюк*

Національний технічний університет «Дніпровська політехніка», м. Дніпро, Україна

* Автор-кореспондент e-mail: miakenkyi.ar.v@nmu.one

Здатність людини розпізнавати та виокремлювати сенси слів при роботі з текстовою інформацією відноситься до вищих когнітивних функцій мозку, зокрема до когнітивного процесу розуміння. Розв'язання задачі виокремлення сенсу слів у тексті належить до задач обробки природних мов або natural language procesing (NLP) та має назву «усунення неоднозначності слів» або word sense disambiguation (WSD), для вирішення якої існують багато підходів, зокрема з використанням нейронних мереж.

Мета. Створення та аналіз архітектури нейронної мережі двонаправленої LSTM для розв'язання задачі WSD в українській мові.

Методика. Одним із сучасних підходів для розв'язання задачі WSD є використання моделей LSTM – типом рекурентної архітектури нейронних мереж, що дозволяє фіксувати довгострокові залежності при моделюванні послідовностей. Для визначення ефективності використання даної архітектури під час дослідження були побудовані дві нейронних мережі: за класичною архітектурою LSTM та її вдосконаленою версією – Bi-LSTM. У рамках дослідження також був сформований набір даних, оснований на словнику української мови SUM. Отримані моделі були навчені на сформованому наборі даних, після чого був проведений порівняльний аналіз отриманих даних.

Результати. Аналіз результатів точності роботи побудованих моделей дозволив визначити ефективність нейронної мережі, побудованої за архітектурою Bi-LSTM. Отримані результати точності дорівнюють відповідно 73 % для LSTM моделі та 83 % для Bi-LSTM, що обумовлено наявністю у моделі Bi-LSTM додаткового шару, який надає можливість для врахування повного контексту слова у поданому тексті.

Наукова новизна. У роботі встановлена ефективність моделі нейронної мережі, побудованої за архітектурою Bi-LSTM, для розв'язання задачі усунення неоднозначності слів у текстах українською мовою у порівнянні з класичною архітектурою LSTM.

Практична значимість. У результаті роботи запропонована модель, що дозволяє розв'язувати задачу усунення неоднозначності слів в українській мові, яку можна використовувати у задачах обробки текстів, зокрема для моделювання когнітивного процесу розуміння.

Ключові слова: *когнітивне моделювання, когнітивний процес, NLP, WSD, LSTM, Bi-LSTM, pymorphy2, stanza, tensorflow*

The manuscript was submitted 11.08.24.