UDC 004.93 **doi: 10.26906/SUNZ.2025.4.108-113**

Olena Peredrii

Simon Kuznets Kharkiv National University of Economics, Kharkiv, Ukraine

SHALLOW ANN MODELS TO CLASSIFY UKRAINIAN AI-GENERATED TEXT

Abstract. In this study, we address the task of detecting AI-generated fragments within Ukrainian-language texts. The objective is to develop a tool capable of identifying content produced with the assistance of artificial intelligence, particularly in PDF documents related to the IT domain. The research explores and analyzes existing solutions and approaches currently available in this area. Several commercial AI-content detectors were evaluated using our custom datasets. The dataset was constructed by segmenting bachelor's theses from IT-related fields into fragments of approximately 1,000 characters each. Five artificial neural network models were tested using the custom dataset combined with a traditional NLP pipeline, achieving an accuracy of 87–88%. Given the complexity of the problem and the ethical considerations within the educational context, the classification results should be further validated by human experts. The current implementation can serve as a foundation for future improvements.

Keywords: artificial neural networks (ANN); shallow ANN models; AI-generated content (AIGC); LLM; AI detector.

Introduction

Artificial Intelligence (AI) and Generative AI (GenAI) in the form of Large Language Models (LLMs) (often referred to as "AI" these days) are fundamentally changing not only life and work, but also the educational process. The widespread use of hundreds of LLMs in the form of chatbots in recent years has opened up many opportunities for both students and teachers. These include organizing personalized learning with the help of chatbots, automating routine tasks, creating tests and assignments (and even grading them), adapting course content, and more [1]. However, the use of GenAI tools also poses significant risks to academic integrity and ethical usage by both students and teachers [2]. A major challenge for teachers is understanding whether the solutions to problems proposed by students are simply artificially generated [1, 2]. When students use AI to perform cognitive tasks that require thinking, forming their own thoughts in the form of text, or generalizing, it undermines the fundamental goal of academic work, which is to develop critical thinking and master clear communication skills [1].

On the other hand, the introduction of technical tools that can create a basis for unfairly accusing students or teachers of violating academic integrity can increase the number of avoidable conflicts, as well as reduce the level of trust in the academic environment.

Despite this, the problems of violating academic integrity are relevant (checking texts for possible plagiarism has been performed automatically for many years), and tools for automatically identifying generated content in the form of texts or images continue to be created and become widespread both in the academic environment and when solving professional tasks related to working with content and requirements for it.

Review of existing solutions

Despite the flaws and drawbacks of AI content tools mentioned above, AI-created content detection problem generated huge scientific interest over the last few years [3–6]. Machine-generated content detectors are designed to identify text produced by AI (AI-generated content - AIGC), often asserting their effectiveness across various conditions and different language models. These detection methods are generally categorized into three primary approaches [3, 7, 8] described below.

Trained Detectors. These systems leverage labeled datasets comprising both human-written and AI-generated texts to train binary classification models. Through this training, they learn to recognize distinct patterns, structures, and stylistic characteristics unique to each type of text.

The frequent choice for this approach is to use BERT-based pretrained models as feature extractors for the text being analyzed. An example of this could be [9], where authors used RoBERTA models to classify AI-generated texts with 95% accuracy.

The representative of this family could also be GPTSentinel [10, 11], which includes RoBERTa-Sentinel and T5-Sentinel models. The first one uses RoBERTa pretrained model to extract features of the text with further classification of them using MLP. The second approach represents a sequence-to-sequence problem solved using the T5 model and producing "positive" or "negative" classification keywords in text form. Both models achieved over 97% accuracy.

RADAR [12–14] is the other example from the supervised classifier family. It addresses the issue with the failure of AI-text detections when facing LLM-paraphrased texts (available for the English language only).

The main drawbacks of the trained detectors family are obvious: building these models requires AI/ML/DL knowledge, the dataset according to the problem being solved, the hardware and time resources to train, deploy, and support the solution. But such models are effective in the particular domain, and could be modified or improved according to new requirements.

Zero-Shot Detectors. Unlike trained detectors described above, these methods do not require explicit training on AI-generated text from specific models. Instead, they infer the origin of AI text based on general linguistic properties, statistical features, and anomalies, or the inherent predictability of machine-generated content. The perplexity, burstiness, N-gram, and probability analysis are often mentioned in papers for the methods of this family.

It was shown that AI-generated text tends to occupy negative curvature regions of the model's log probability function, so this could be the only classification criterion [15]. DetectGPT detector [15] does not require training of a special classifier, but uses perturbation of the text to build a log probability function to make the decision.

Fast-DetectGPT [16] was proposed as an update of DetectGPT and included a performance boost and accuracy improvement.

The main limitation for this set of methods is the availability of the probabilities for the model's dictionary; for instance, a lot of popular LLM models don't reveal them.

Watermarking Techniques. This approach involves embedding a subtle, hidden pattern directly into the generated output, which can later be detected to confirm its AI origin [7]. While theoretically robust, this method is less commonly employed in black-box detection scenarios where the detector has no control over the content generation process. Probably, the continuous advances in LLM will make watermarking the only method to classify AIGC in the future.

There is also a separate field of commercial detectors, e.g. GPTZero, ZeroGPT, Originality.ai, and others. They use different methods and their combinations to classify AI-generated content, and often provide benchmarks to compare quality with competitors.

GPTZero [17] reported an accuracy over 99% and a false positive rate below 1% [18] for the languages it supports. There is no official support for Ukrainian, but results are available for this language with a "partially supported" system message.

ZeroGPT [19] claims 98% accuracy, based on training with 10 million AI-generated articles, and support of all available languages.

Originality.ai [20] claims 99% AI Content Detection Accuracy, positioning itself as a highly accurate detector for various AI writing tools, including GPT4.1, ChatGPT4o, Gemini 2.5, Claude 3.7, and DeepSeek V3, supports Ukrainian.

Copyleaks AI Detector [21] offers over 99% accuracy, supports 30+ languages, and detects popular LLM models, including ChatGPT, Gemini, and Claude, as well as newer models as they're released.

Isgen.ai supports over 80 languages, including Ukrainian, and claims to reach over 96% accuracy [22].

The consistent claims of very high accuracy (98-99%+) from commercial tools like Copyleaks, Originality.ai, and ZeroGPT contrast with scientific papers' research, which report on the various problems and accuracy issues with unseen data.

Finally, there are many enthusiastic projects (e.g., located on GitHub) with custom AI-text detectors, but a lot of them are already abandoned.

As a summary of this section, we can conclude that the problem of AIGC classification is complex both from a research and a practical point of view. A lot of LLMs exist, and they are evolving quickly, providing better and better quality of content, which requires constant updating of AI-detectors, which themselves are not very precise in practice initially. More control over quality and classification process is possible with the training of custom models in the domain of the problem, but this approach requires knowledge, data, time, and resources for the development, deployment, support, and continuous improvement. There are no good solutions to support the detection of AIGC in the Ukrainian language.

Ethical considerations

Modern AIGC detection tools are widely known for their unreliability and classification errors, especially in cases where the author of the text is not a native speaker of the verification language [1].

False accusations of academic misconduct arising from unreliable AI detectors can have serious consequences for authors, whether they are students or teachers. The widespread problem is not just the inaccuracy of these detectors, but also the underlying reason for this: they are designed to detect statistical patterns, but human writing is inherently varied and unpredictable, and artificial neural networks and quite good (and constantly improving) at detecting these statistical features. Additionally, authors have no tools to prove the authorship of the content when some AIGC detection tool mistakenly classifies the content to be AI-generated. There is an opinion [23] about what educational institutions should do instead of screening AI-generated content with AI-detectors. These actions include improving teaching and process-based assessment methods, oral exams and discussions, and educating students on the responsible use of AI tools.

The problem definition

The idea of the work is to build a research AI tool that is useful in the detection of the AIGC in the PDF documents in the IT domain. Taking into account all previous considerations, we want to focus on the principal axioms we are following in the research:

– this tool is not used for the automatic AI decision-making about AIGC;

– the developed tool is not used for the blind blaming of authors, it is much more a helpful assistant/recommender for the checker to find strange/inconsistent pieces of text for further improvements;

– we don't focus on the high-accuracy numbers a lot; we understand the complexity of the problem, and care more about reliable and honest results for this particular dataset, domain, and initial conditions.

Dataset

The dataset was created using the processing of 38 files with bachelor diplomas of students graduated in 2022-2024. The original text was split into segments of about 1000 symbols each, with 150 symbols overlapping between chunks without breaking words. The chunking process is important here as it defines the further usage and the evaluation of the models. This is very difficult to define if a short text (e.g., a sentence) was written by a human or generated from our point of view, so we created chunks of some valuable size. On the other hand, the results for each sentence classification are interesting, but future models should operate with bigger chunks. Finally, the dataset contains the chunks, even when by

1000-symbol pieces, it is impossible to detect whether it was generated or human-written. We left these examples deliberately to see how they would be classified, so the dataset is very challenging initially.

The chunks obtained after splitting were processed manually in order to remove unnecessary spaces, break lines, and special symbols, which could lead to classification based on them. After this cleaning, we left 2533 chunks.

For each cleaned human-written chunk, we generated its AI-written version using "gpt-4o-mini" model and this prompt: "Analyze the topic of the text below and create similar text in Ukrainian. Don't provide your summary, just use it for generation. Don't use Markdown, just plain text. Don't rewrite the text, generate your own:". After that, we received more than 15000 chunks (a few chunks appeared instead of a single one because LLM added a lot of break lines), which were processed manually as well. Some chunks contained summary, markdown symbols (so, our model may fail for the obvious case when markdown symbols show that the text is definitely AI-generated), a lot of lines, or just bad text. After the cleaning, we obtained 2634 chunks, and this quantity is comparable with 2533 human-written chunks, which makes this dataset balanced.

Evaluation with different existing AI-detectors

We tested how known commercial detectors work for our custom dataset. We used evaluation versions of the software for all of them for classification, but followed our main idea – to classify text chunks grabbed from the documents in an automatic manner, exactly like we are going to do. The quantity of experiments differs between commercial tools because they provide different quantities of text tokens available for free (and no upload of big documents), but we tested the same text chunks for fair comparison.

For the five human-written text chunks classified by GPTZero we received five uncertain decisions, only one of them showed that the text was created by a human (74% human and 26% AI), all others were shown as AI-generated. Two text chunks from the five AI-generated texts were classified as human texts (all of them with uncertain decisions again).

ZeroGPT tool detected nine out of ten human-written pieces as AI-generated (the correct sample contains 44% of AI, all others – at least 87%), and all ten AI-generated samples were classified correctly (with a 98% minimum quantity of AI-content).

Originality.ai classified 2 out of 10 human-written text chunks as original text (with high confidence – 98%) and 8 others as AI-generated (also with high – at least 97% – confidence). One AI-generated chunk was classified as human-written, the other 9 were classified correctly (all of them with at least 97% confidence).

We were able to test 20 text chunks with Copyleaks detector, 4 human-written chunks were identified as AI-generated (100% of AI-generated content), and 16 chunks were classified correctly (0% AIGC). The situation is similar for AI-generated pieces of text: 5 of them were misclassified as human-written (0% AIGC), and 15 were classified as AI-written (100% AIGC).

All five human-written chunks were correctly classified by the Isgen.ai tool, but 2 AI-generated pieces out of 5 were classified also as human text (notifying that 100% content was human-created).

Grid search for the shallow architecture of the model

We used a traditional NLP approach to solve text classification problems: build an ANN model with one embedding layer, a single dense layer, and an output layer containing just one neuron. One of our goals is to create such shallow models that could be trained on a single CPU/GPU using regular hardware available for everyone. 80% of the dataset were used as training/validation, and the remaining 20% were used as a test part.

The preprocessing of the text chunks includes the calculation of the vocabulary size for the training part of the dataset, training of a tokenizer for all training chunks according to the vocabulary size, and pre-padding of vectors to align all their lengths. The lengths of vectors in the train data vary from about 230 symbols (appearing as text chunks at the end of documents) to 1913 symbols (appearing because AI-generated chunks sometimes are longer compared to the initial ones), but about 90% of lengths are in [850;1150] range. This tokenizer was serialized and used during testing of the models.

The effective architecture of such shallow models could be found with grid search, exploring a lot of possible combinations of hyperparameters, revealing also the opportunity to create ensembles of the models to make a final classification decision.

Input vectors are fed into the first embedding layer of the model, which converts integer labels of the words according to the dictionary to meaningful continuous values during training. The input parameters for this layer are known already: size of vocabulary, size of the input vector, and the size of the output vector – the hyperparameter for this layer that should be defined via grid search. Models tend to overfit during training, so we tested values in [1;10] range as the size of the output vector. The quantity of neurons for the dense layer we tested was the powers of 2 in the range [2;128]. There were no benefits in using longer embedding vectors, pretrained embeddings, or an additional dense layer for this task.

Training of nets was performed using Adam optimizer, L2 regularizer, binary cross-entropy loss function, RELU activation for the middle dense layer, sigmoid for the last one, dropout layer, and early stopping procedure, saving the best state of the model before it starts overfitting.

Fig. 1 shows the entire picture of the grid with the test accuracies of the models (better accuracies are highlighted with green color).

Based on these results, we choose such five model architectures: model 1 contained 2 embedding neurons and 256 dense neurons, model 2 – 3 embeddings and 64 dense ones, model 3 – 4 embeddings and 64 dense neurons, model 4 included 6 embedding neurons and 64 dense neurons, finally, model 5 was built with 10 embedding and 128 dense neurons.

dense layer neurons	embeddings out quantity									
	1	2	3	4	5	6	7	8	9	10
2	0,51	0,51	0,75	0,51	0,51	0,51	0,51	0,51	0,51	0,51
4	0,72	0,51	0,51	0,51	0,51	0,51	0,51	0,51	0,51	0,51
8	0,75	0,79	0,72	0,71	0,51	0,51	0,51	0,51	0,51	0,51
16	0,83	0,84	0,87	0,51	0,51	0,72	0,81	0,51	0,51	0,51
32	0,8	0,87	0,85	0,82	0,74	0,86	0,87	0,73	0,71	0,87
64	0,54	0,83	0,89	0,86	0,87	0,88	0,84	0,87	0,87	0,87
128	0,76	0,85	0,84	0,86	0,88	0,87	0,87	0,88	0,88	0,89
256	0,88	0,84	0,87	0,88	0,86	0,87	0,86	0,86	0,87	0,85

Fig. 1. Results of grid search
for the selection of ANN architecture

Test accuracies for them were 88.00 (F1 score for human class was 0.88, AI – 0.88), 87.62 (0.87, 0.88), 87.91 (0.87, 0.88), 87.75 (0.87, 0.87), and 87.13 (0.87, 0.87), respectively.

Ensembles

Using ensembles of ANN instead of making the decision by just a single ANN can achieve better results for the problems with insufficient data (multiple models can learn different features of the data), or when a single model does not provide good accuracy (a few models can compensate the classification mistakes made by others).

There are a lot of ways to make a joint decision based on the outputs from different models. They include averaging of outputs, prediction the majority of models after voting, different weighted schemes of models' outputs, boosting or even stacking of the models: training another classification model based on the outputs of models in an ensemble. In the context of our problem, we considered the ensemble as a way to improve classification accuracy and provide a broader picture of the decisions about the chunk of text.

Referring to the accuracy of five models as the probabilities of correct classification, it is possible to calculate the corresponding probability for the ensemble according to the aggregating scheme. We tested the majority voting approach, where the decision is made in favor of the class that receives the most votes from separate models. This was not successful, as all models failed synchronously for the same text chunks from the test part of the dataset. We also tested AdaBoost and GradientBoosting classifiers without success.

Chunks and sentences classification

The usage of a trained model for text chunks processing is straightforward and presented in Fig. 2. The initial PDF document is divided into non-overlapping text chunks of about 1000 symbols each, and each of these chunks is processed using five models M1 – M5. The result is averaged per chunk, and the statistics are gathered to show the overall probability that a random text chunk from the document is AI-generated, with the histogram showing the summary of classification results for chunks.

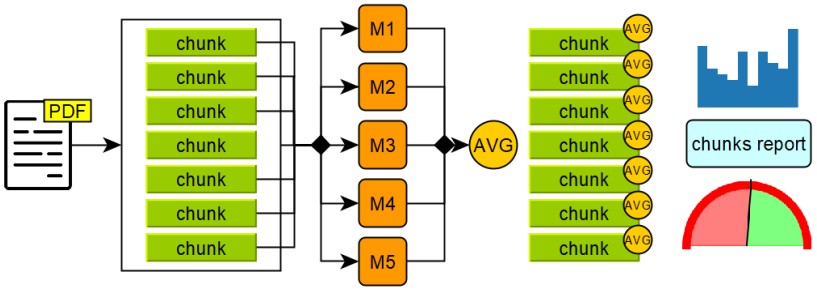

Fig. 2. The processing pipeline to create chunks report

This is known [24] that the classification of longer text chunks is more reliable compared to the classification of sentences. The models we have trained were designed to work with chunks of about 1000 symbols, and using them to classify short sentences may lead to incorrect results. On the other hand, the beautiful visualization of the AI-generation probability for the particular sentence is interesting.

To gather information about separate sentences, we perform the chunking of the text into overlapping pieces with the step length that is equal to the sentence length: first 1000 symbols of the document form the first chunk, the next chunk is formed with such a shift that removes the first sentence from the previous, then the second sentence, and so on.

Thus, each sentence is a part of multiple chunks, and we accumulate prediction results for it as a representative of these chunks. Each prediction is already an averaged value over all models M1 – M5, and a set of these predictions could be averaged again (or just visualized without aggregation for better analysis) into a single output value for each sentence (Fig. 3).

Results

The evaluation of the developed system was tested for the initial drafts of student bachelor's diplomas, theses graduated in 2025, in the review mode without any actions being taken based on the results. The automatic evaluation is complicated and requires a lot of human work, especially for the documents' ground-truth, which is unknown, e.g., we don't know ourselves whether the text was generated or not.

The only available tool we can use for the analysis of entire documents is the nice Plagramme software [25] that is capable of plagiarism and AI detection in the text files, and provides classification results for each separate sentence in percentages of confidence. Quick tests based on the text chunks from our dataset (the first 100 pieces of human-written and AI-generated pieces were used as a single document) showed a good level of AIGC detection for human-written texts (below 5%), but only about 50% for text created from AI-generated pieces. So, based on these results, we can conclude that this tool tends to miss some AIGC content for our domain and,

probably, this format (separate chunks already obtained) of the problem. We perform a manual two-sided comparison of the text analysis reports with our tool and Plagramme. We roughly compared reports paragraph-by-paragraph, focused only on paragraphs with AIGC found, and ignored short sentences classified as AI-generated in between human-written paragraphs (this seems not to be very reliable). So, 31 text pieces identified by Plagramme as AIGC have the output classification value 0.5 or above obtained with our models, and 12 pieces below 0.5.

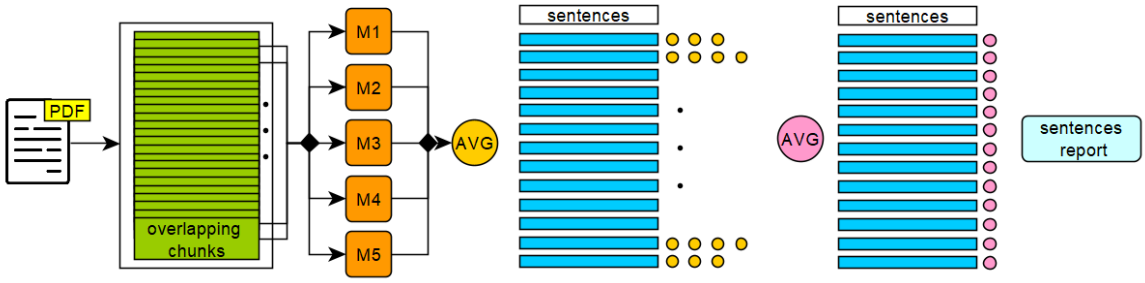

Fig. 3. The processing pipeline to create a sentence report

On the other hand, 26 text pieces classified by our approach as AIGC have a high (about 100%) level of confidence that it was AI-generated by Plagramme, and 15 pieces were classified as human-written. Commonly, there is a strong agreement on all decisions about significant AIGC pieces of text in both reports, but both reports contain errors when definitely human-written text was mistakenly classified as AIGC and vice versa.

The example of a pair of reports is shown in Fig. 4. The left part contains the screenshot from the Plagramme software analysis report, where all 3 sentences were classified as AIGC, and the last one as human-written.

The right part represents the result of highlighting a PDF file based on the proposed approach, where the first sentence was not marked successfully (this happened sometimes because of the complexity of mapping text in PDF), the next two sentences were classified as AIGC with 0.92 and 0.8 confidence, and the last sentence is uncertain with a confidence of 0.54.

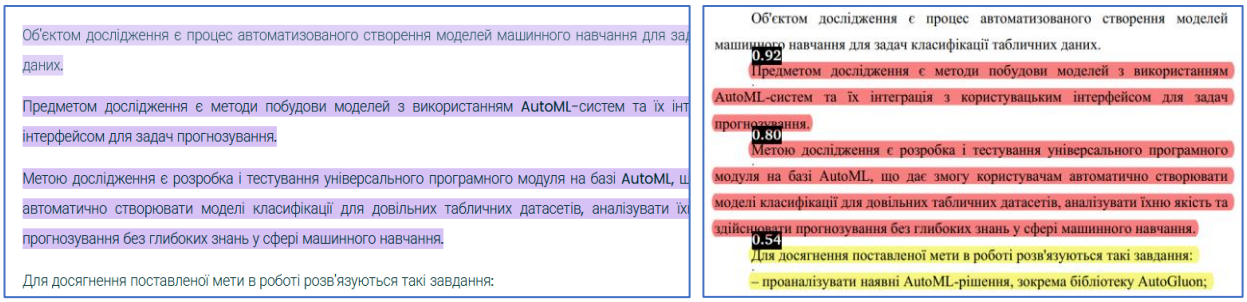

Fig. 4. Example of the report after analysis of the document

Conclusion

In the scope of this research, the AI tool was implemented to classify whether the piece of text in Ukrainian language in the IT domain was AI-generated or human-written.

Five different shallow artificial neural network models were trained using the custom dataset and traditional NLP pipeline, and an accuracy of about 87-88% was achieved. Taking into account the complexity of the problem as well as the ethical background in the educational domain, the result of classification should be analyzed by humans; no automatic AI decision is considered here.

The current implementation could be used as a core for future improvements, as most time was spent on data preparation and cleaning. The training of a shallow ANN is fast and doesn't require special hardware.

There are a lot of possible ways to improve the current initial result. The output of the created AI tool is natural from a black-box system: there is no explanation about why the model makes the decision in favor of a specific class. The dataset created does not have enough size to apply the ensemble of networks successfully and obtain better classification accuracy. Another interesting question to research may include a better choice of chunk size, the usage of deeper models, and the application of LLM itself for the classification.

Additionally, only ChatGPT tool was used to generate text for the dataset; it is interesting to test another popular chatbots as well and extend possible usage cases of the tool. Finally, we focused here on the classification of directly generated text without editing/humanizing, as this problem is already challenging enough (including our goal to solve it using typical widespread hardware available for everyone).

References

1. From Tool to Temptation: AI's Impact on Academic Integrity, UMass Amherst. [Online]. Available: https://www.umass.edu/ideas/news/tool-temptation-ais-impact-academic-integrity.

2. Bittle K., & El-Gayar O. "Generative AI and Academic Integrity in Higher Education: A Systematic Review and Research Agenda" Information 2025, 16(4), 296. Apr. 2025 doi: https://doi.org/10.3390/info16040296.
3. Fraser K. C., Dawkins H., Kiritchenko S. "Detecting AI-Generated Text: Factors Influencing Detectability with Current Methods". Journal of Artificial Intelligence Research 82 (2025) pp. 2233-2278. doi: https://doi.org/10.1613/jair.1.16665.
4. Abdali S., Anarfi R., Barberan CJ, He J. "Decoding the AI Pen: Techniques and Challenges in Detecting AI-Generated Text". KDD '24: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 6428-6436, Aug. 2024. doi: https://doi.org/10.1145/3637528.3671463.
5. Li Y., Li Q., Cui L., Bi W., Wang L., Yang L., Shi S., & Zhang Y., "Deepfake Text Detection in the Wild". 2023. URL: https://ar5iv.labs.arxiv.org/html/2305.13242.
6. Li Y., Li Q., Cui L., Bi W., Wang Z., Wang L., Yang L., Shi S., & Zhang Y., "MAGE: Machine-generated Text Detection in the Wild. " May 2024. doi: https://doi.org/10.48550/arXiv.2305.13242.
7. Tufts B., Xuandong Zhao, Lei Li. A Practical Examination of AI-Generated Text Detectors for Large Language Models URL: https://arxiv.org/html/2412.05139v4.
8. Balla E. (2025, May 22). How NLP Powers AI-Generated Text Detection. The AI Journal URL: https://aijourn.com/how-nlp-powers-ai-generated-text-detection/
9. Solaiman I., et al., "Release strategies and the social impacts of language models". OpenAI Report, Nov. 2019, doi: https://doi.org/10.48550/arXiv.1908.09203.
10. Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, Bhiksha Raj. "Token Prediction as Implicit Classification to Identify LLM-Generated Text". In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 2023, pp.13112–13120. doi: https://doi.org/10.48550/arXiv.2311.08723
11. Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, Bhiksha Raj. "GPT-Sentinel: Distinguishing Human and ChatGPT Generated Content". doi: https://doi.org/10.48550/arXiv.2305.07969
12. Xiaomeng Hu, Pin-Yu Chen, Tsung-Yi Ho. RADAR: Robust AI-Text Detection via Adversarial Learning. doi: https://doi.org/10.48550/arXiv.2307.03838.
13. Radar tester: robust ai-text detection via adversarial learning. URL: https://radar-app.vizhub.ai/
14. RADAR-Vicuna-7B Model, Hugging Face. URL: https://huggingface.co/TrustSafeAI/RADAR-Vicuna-7B
15. Mitchell E., Lee Y., Khazatsky A., Manning C. D., & Finn C., "DetectGPT: Zero-shot machine-generated text detection using probability curvature," Proceedings of the 40th International Conference on Machine Learning (ICML'23), Honolulu, HI, USA, 2023, Article No.: 1038, pp. 24950–24962. doi: https://doi.org/10.48550/arXiv.2301.11305
16. Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, Yue Zhang. "Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature". doi: https://doi.org/10.48550/arXiv.2310.05130.
17. GPTZero. URL: https://gptzero.me/
18. Napier E. "GPTZero. Behind the Scenes: Multilingual Detection Update". May 2025. URL: https://gptzero.me/news/behind-the-scenes-multilingual-detection-update/
19. ZeroGPT. URL: https://www.zerogpt.com/
20. AI Checker – Most Accurate AI Detector. URL: https://originality.ai/ai-checker.
21. AI Detector – Free AI Checker for ChatGPT, GPT-4, Gemini & More. URL: https://copyleaks.com/ai-content-detector.
22. Забезпечення автентичності: найточніший український ШІ-детектор. Isgen.ai detector. URL: https://isgen.ai/uk.
23. Markley T. (2025, February 27). The Problem with AI Detectors: Why Professors Should Reconsider Their Use, URL: https://www.kaltmanlaw.com/post/problem-with-ai-detectors-professors-should-rethink.
24. Tian Y., Chen H., Wang X., Bai Z., Zhang Q., Li R., Xu C., & Wang Y. "Multiscale positive-unlabeled detection of AI-generated texts". In Proceedings of the Twelfth International Conference on Learning Representations (ICLR), 2024. doi: https://doi.org/10.48550/arXiv.2305.18149.
25. Plagramme – Plagiarism checker & AI detector. URL: https://plagramme.com.

Received (Надійшла) 07.09.2025
Accepted for publication (Прийнята до друку) 05.11.2025

ABOUT THE AUTHORS / ВІДОМОСТІ ПРО АВТОРІВ

Передрій Олена Олегівна – кандидат технічних наук, доцент кафедри інформатики та комп'ютерної техніки, Харківський національний економічний університет імені Семена Кузнеця, Харків, Україна;
Olena Peredrii – PhD, Associate Professor, Department of Informatics and Computer Engineering, Simon Kuznets Kharkiv National University of Economics, Kharkiv, Ukraine;
e-mail: olena.peredrii@hneu.net; ORCID: https://orcid.org/0000-0003-0390-1931;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57202751577.

Неглибокі ANN-моделі для класифікації україномовного тексту, згенерованого штучним інтелектом

О. О. Передрій

Анотація. У цьому дослідженні розглядається задача виявлення фрагментів, згенерованих штучним інтелектом, в україномовному тексті. Мета роботи – розробити інструмент, який допомагатиме визначати контент, створений за допомогою ШІ, зокрема у PDF-документах, що стосуються ІТ-сфери. В роботі було розглянуто та проаналізовано які рішення та підходи існують на даний момент. Створено власний тестовий набір даних за допомогою розбиття дипломних робіт бакалаврів ІТ спеціальностей на сегменти приблизно по 1000 символів кожен. Виконано тестування роботи різних комерційних детекторів на цьому наборі даних. Побудовано та навчено п'ять моделей штучних нейронних мереж із використанням цього набору та традиційних NLP процедур обробки тексту, було досягнуто точності 87–88%. Зважаючи на складність проблеми та етичні аспекти застосування інструменту у сфері освіти, результати класифікації повинні бути додатково перевірені експертами. Поточна реалізація може слугувати основою для подальших удосконалень.

Ключові слова: штучні нейронні мережі (ШНМ); неглибокі моделі ШНМ; контент, створений штучним інтелектом (AIGC); LLM; детектор штучного інтелекту.