P. V. Zdebskyi; A. Yu. Berko, Dr. Sc. (Eng.), Professor

VERIFICATION OF THE TEXT AFTER GENERATION BY MEANS OF LARGE LANGUAGE MODELS FOR FILTRATION OF THE INCORRECT ANSWERS

Nowadays the problem of matching large language models becomes relevant. Models are able to perform various tasks, using zero-shot approach. But as they became more intelligent, they find alternative routes for the solution of the tasks not as the researches expect. This is especially dangerous in the production environment because it is difficult to control the output of the model which was taught to be universal. In the given paper it is suggested to use one and the same model several times in different form in order to improve the quality of the generated text.

Method of accuracy increase of the models of the text content generation was further developed. In this case the user has not to provide tens of examples of the desired and non-desired behavior of the model because the model itself can do this automatically. That is, unlike the conventional methods of model accuracy enhancement, which require the training set of the models, the proposed approach includes the identification stage. As a result of identification we obtain the set of examples, on which the model learns and enhances its accuracy.

Two specific methods were proposed in the given research. The first method simply uses the model of the discriminator for the verification of the results of the generator model and repeats the request to create the text, if the results do not correspond the criteria of the user. By means of this method all the incorrect generations were removed but due to denotating the third of the correct generations as incorrect. The second approach is more complicated, besides the discriminator it uses the model of the simulator. The process required that the simulator model generated the samples of the data, entered by the user, after that the generator would generate the text of the answer for each sample and the discriminator would verify the generated results and add them to training data. This will increase the accuracy from 56 % to 66 % in the problem of logic conclusion.

Key words: *gpt-4, coordination problem, text generation, processing of natural language, problem of logic conclusion.*

Introduction

With the increase of the size of the models for natural language processing there appears the problem of their aligning. The problem of aligning is one of the subproblems of artificial intelligence security [1, 2]. Methods of multistage texts generation, using large language models for the aligning of the generated text with the user`s request are suggested in the given paper.

For the improvement of the accuracy of large language models various strategies are used. The developer of GPT-4, company Open AI, states that the division of complex problems into simple is one of the strategies of prompt engineering for the improvement of the results. For instance, one model classifies the user`s request and another suggests the solution of user`s problem. Other strategy is to give time to think. For example, the model took from the book all the abstracts, regarding certain subject, after that it can be asked to verify if any abstract is forgotten.

Nowadays active studies, connected with multistage text generation are carried out [3]. The example can be the work Large Language Modelsare Zero-Shot Reasoners, where it is suggested to request model to describe steps, needed for problem solution [4]. After it generates the steps, they together with the original task are transferred to the input of this model to get the final answer. In the work Bootstrapping Reasoning With Reasoning it is suggested to generate step-by-step solution of the problem and then add it to the training data set [5]. Only those step-by-step solutions which lead to the correct answer will be added, the correctness can be verified because we work with the labeled data. After supplementing of the training set with new data we verify one more time the examples for which step-by-step solution was generated incorrectly.

One more example can be the research Self-Consistency Improves Chain of Thought Reasoningin Language Models where the model generates many answers on one question and the answer which is common is considered to be final [6].

In the work Training Verifiers to Solve Math Word Problems model verifier which will verify the generator is trained [7]. Similar approach was used in this work but the suggested approach does not need labeled data.

Objective of the paper and problem set-up

In the given study multistage approach, including model discriminator for the verification of the generation results is used. That is, after main model generates the text, it is transferred to the input of another model which differs from the main by the prompt. This model discriminator determines if the result corresponds to the original request of the user.

Two ways of the discriminator usage to study if the retransmission to the input of large language model of the generated examples can improve the generation quality are considered. The first method is used for eliminating of incorrectly generated text and the second method is used for labeling the results of the generation to increase the training data of the basic model. In the second method the examples where the model made a mistake are transferred but it is indicated that these are the examples of negative behavior, they prevent model from making similar mistakes. The process is completely automated as the labeling is performed by the discriminator.

Program realization

User requests the model to generate the next which will correspond to certain rules and provide several examples. Model uses data, introduced by the user, for the response generation but prior its presentation the model verifies if the generated text corresponds to the rules, set by the user. If not, it is added as the example of the negative behavior for the additional training of the model. To verify this approach, the simple task was chosen to generate the implication from the sentence. Architecture of the system is presented in the Table 1.

Table 1

Components of the system

Name of the system architecture component	Designation
get_baseline_model_response	Obtaining the results of the basic model GPT-4.
get_classification_model_response	Classification of the results of basic model into true and false (false results are used for model improvement).
get_trained_model_response	Obtaining of the results of the improved model.

Algorithm of model functioning comprises such basic steps:

- Text generation on the basic model.
- Identification of the correspondence to user`s criteria.
- Training of the final model which takes into consideration negative examples of the behavior from the previous step.
- Text generation on the base of final model.

After carrying out of the initial experiment the decision was made to change the approach. In the previous approach the set of utterances was divided into two groups but it was decided to allocate the separate set of data for testing all the models as in this case we simulate the behavior of the model in real operation conditions. Now the model will independently generate the examples of user`s requests. As in the previous approach we gave additional set of model data as compared with the basic, then the quality could be improved due to greater amount of data [8].

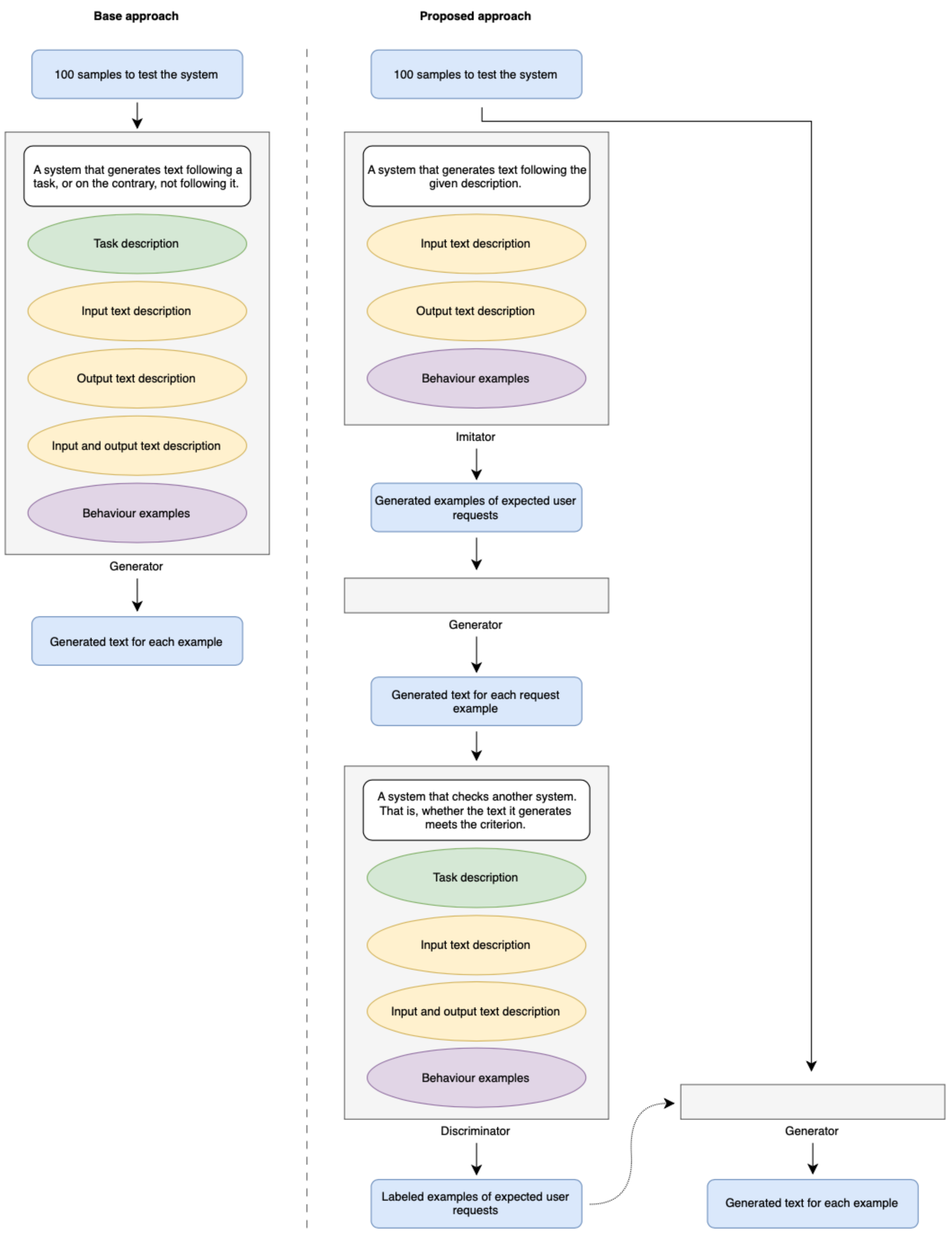

Base approach
100 samples to test the system
A system that generates text following a task, or on the contrary, not following it.
Task description
Input text description
Output text description
Input and output text description
Behaviour examples
Generator
Generated text for each example
Proposed approach
100 samples to test the system
A system that generates text following the given description.
Input text description
Output text description
Behaviour examples
Imitator
Generated examples of expected user requests
Generator
Generated text for each request example
A system that checks another system. That is, whether the text it generates meets the criterion.
Task description
Input text description
Input and output text description
Behaviour examples
Discriminator
Labeled examples of expected user requests
Generator
Generated text for each example

Fig. 1. Usage of the simulator and the discriminator for increasing the training data

Two methods, aimed at improvement of text generative models are proposed. The first approach is when after generation model-discriminator rejects the text if it does not correspond to the original request of the user. Text is generated several times until the discriminator confirms that the text is correct. The second approach is that model simulator generates certain number of potential requests of the user. Then, the response is generated for each request. Each response is verified by the model discriminator and with the corresponding label transfers it into the training data set. Model with this expanded training data set generates the text by original user`s request.

To use the suggested approach it is necessary to formulate the problem, describe the requirements to the input and output text, provide several examples of the correct and incorrect behavior of the system. The first method gives considerable improvement of the results and can be used for any problem, connected with text generation. The second method has narrower application. Especially where there are no resources to prepare more than several examples for the model and if it is easier to formulate the task, showing incorrect examples.

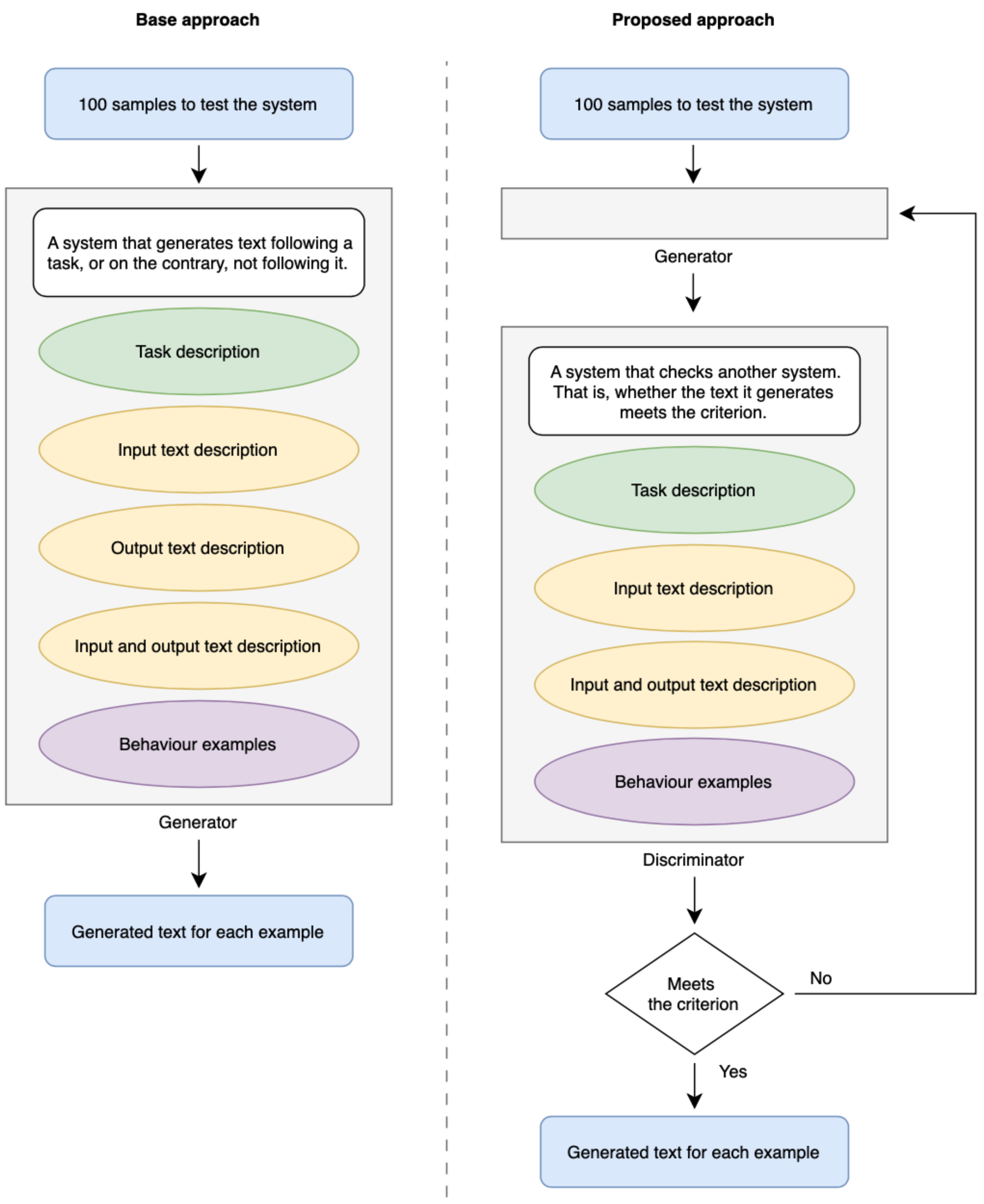

Base approach
100 samples to test the system
A system that generates text following a task, or on the contrary, not following it.
Task description
Input text description
Output text description
Input and output text description
Behaviour examples
Generator
Generated text for each example
Proposed approach
100 samples to test the system
Generator
A system that checks another system. That is, whether the text it generates meets the criterion.
Task description
Input text description
Input and output text description
Behaviour examples
Discriminator
Meets the criterion
No
Yes
Generated text for each example

Fig. 2. Method, using the discriminator for the verification of generation

Method of text generation using the verifier for the assessment of the results when the basic model generates the text is suggested. This method enabled to improve the accuracy of generation on the example of logic conclusion problem [9].

General description of the method:

1. User makes a request x.

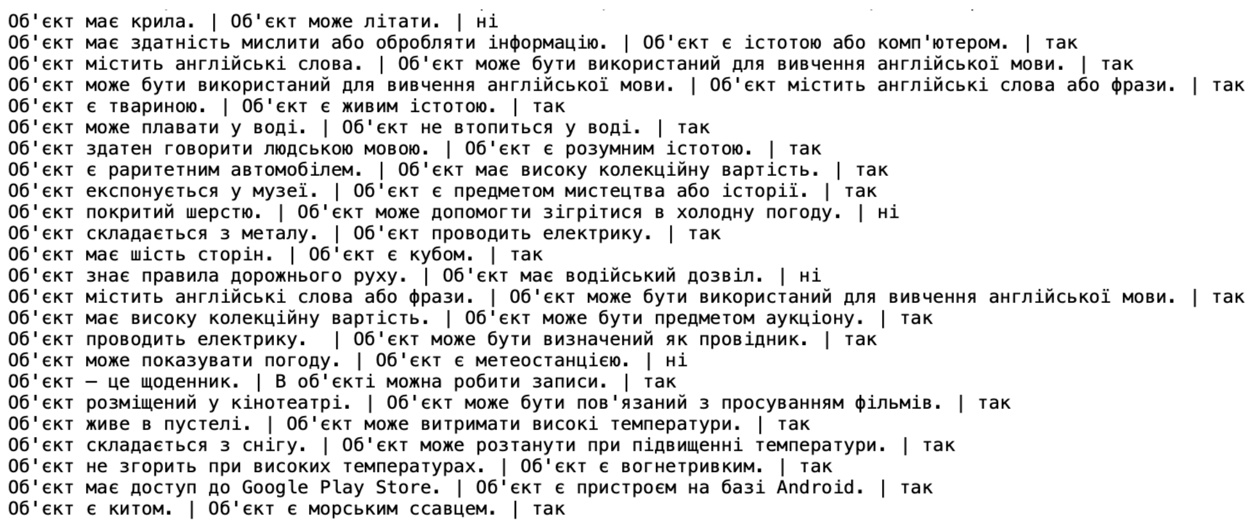
2. Model G generates N examples. By default $N = 40$.
3. Model I classifies each of N examples on the correspondence to user`s request x.
4. Examples z, which do not correspond to the user`s request x, are stored.
5. Copy of the model $G\ is\ created$, the examples z are transferred to it . Copy G with the examples of negative behavior z creates model T, which takes into consideration these examples for generation unlike the original model G.
6. Model T is used to answer the original request of the user x.

Analysis of the results

When the approach, using the additional training of the model on new examples is applied the accuracy increased from 56 % to 66 %. Accuracy matrix was calculated as the ratio of the correct generations to all generations on 100 examples. There are two reasons, limiting the accuracy increase. They are: multiplication of the error and insufficient growth of the number of examples. But more examples, greater is the impact of the errors in training data and greater cost, that requires fine tuning, which is limited in GPT-4. It should be noted that the suggested approaches were tested on the problem of logical conclusion (NLI). The example is the request “Object has all the angles equal 90^0”, basic model responded “Object is the square”, and the suggested model replied “Object is a rectangular”.

Approach, using the model-discriminator for the verification of the generated text is simpler, it was also tested on 100 examples with the same matrix. The result of testing is 60 correct classifications “yes”, 27 correct classifications “No”, 0 incorrect classifications “yes”, and 13 incorrect classification “No”. That is, the model-discriminator correctly rejects approximately 40 % of the responses, generated by the system, because it classifies it as those which do not correspond to the user`s criterion. About two thirds of the responses the model rejected were rejected correctly. The model-discriminator did not pass any incorrect generation but one third of those which were labeled as incorrect, were correct generations. The example of the pair of sentences, rejected by the model-discriminator is the request “Object studies physics” and the response of the basic model “Object is a student”, because not only students study physics.

The example of non-processed results of the basic model is shown below. Each row – is separate pair of statements and label which shows if there is a connection between them. Horizontal line «|» is a divider. The first statement in each row is determined by the user, the second is the statement, which, in model opinion, follows from the first. After that, the request is made to the model-discriminator to determine if the connection between the pair of sentences is logic, that is why, the label is also generated by the model. Pairs of statements then are added to training data set of the final model.



```
Об'єкт має крила. | Об'єкт може літати. | ні
Об'єкт має здатність мислити або обробляти інформацію. | Об'єкт є істотою або комп'ютером. | так
Об'єкт містить англійські слова. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт може бути використаний для вивчення англійської мови. | Об'єкт містить англійські слова або фрази. | так
Об'єкт є твариною. | Об'єкт є живим істотою. | так
Об'єкт може плавати у воді. | Об'єкт не втопиться у воді. | так
Об'єкт здатен говорити людською мовою. | Об'єкт є розумним істотою. | так
Об'єкт є раритетним автомобілем. | Об'єкт має високу колекційну вартість. | так
Об'єкт експонується у музеї. | Об'єкт є предметом мистецтва або історії. | так
Об'єкт покритий шерстю. | Об'єкт може допомогти зігрітися в холодну погоду. | ні
Об'єкт складається з металу. | Об'єкт проводить електрику. | так
Об'єкт має шість сторін. | Об'єкт є кубом. | так
Об'єкт знає правила дорожнього руху. | Об'єкт має водійський дозвіл. | ні
Об'єкт містить англійські слова або фрази. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт має високу колекційну вартість. | Об'єкт може бути предметом аукціону. | так
Об'єкт проводить електрику.  | Об'єкт може бути визначений як провідник. | так
Об'єкт може показувати погоду. | Об'єкт є метеостанцією. | ні
Об'єкт – це щоденник. | В об'єкті можна робити записи. | так
Об'єкт розміщений у кінотеатрі. | Об'єкт може бути пов'язаний з просуванням фільмів. | так
Об'єкт живе в пустелі. | Об'єкт може витримати високі температури. | так
Об'єкт складається з снігу. | Об'єкт може розтанути при підвищенні температури. | так
Об'єкт не згорить при високих температурах. | Об'єкт є вогнетривким. | так
Об'єкт має доступ до Google Play Store. | Об'єкт є пристроєм на базі Android. | так
Об'єкт є китом. | Об'єкт є морським ссавцем. | так
```

Fig. 3. Example of the discriminator results

Conclusions

The results show that the suggested approaches improve the quality of text generation. Simpler approach with the verification of the generated text and repeated generation gives considerable improvements of the generation quality. The second approach shows better results than the basic model and will be especially useful in case when the user can not provide the model with the sufficient amount of examples to describe the desired behavior of the model for text generation.

REFERENCES

1. The alignment problem from a deep learning perspective [Electronic resource] / R. Ngo, L. Chan, S. Mindermann // arXiv. – 2022. – Access mode: https://arxiv.org/abs/2209.00626.

2. Goal Misgeneralization in Deep Reinforcement Learning / L. Langosco; J. Koch, L. D. Sharkey [et al.] // Proceedings of the 39th International Conference on Machine Learning. International Conference on Machine Learning.PMLR. – 2022. – P. 12004–12019.

3. Training language models to follow instructions with human feedback [Electronic resource] / L. Ouyang, J. Wu, X. Jiang [et al.] // arXiv. – 2022. – Access mode: https://arxiv.org/abs/2203.02155.

4. Large Language Models are Zero-Shot Reasoners [Electronic resource] / T. Kojima, S. S. Gu, M. Reid [et al.] // arXiv. – 2022. – Access mode: https://arxiv.org/abs/2205.11916.

5. STaR: Bootstrapping reasoning with reasoning [Electronic resource] / E. Zelikman, Y. Wu, J. Mu, N. D. Goodman [et al.] // arXiv. – 2022. – Access mode: https://arxiv.org/abs/2203.14465.

6. Self-consistency improves chain of thought reasoning in language models [Electronic resource] / X. Wang, J. Wei, D. Schuurmans, [et al.] // arXiv. – 2022. – Access mode: https://arxiv.org/abs/2203.11171.

7. Training verifiers to solve math word problems[Electronic resource] / K. Cobbe, V. Kosaraju, B. Mohammad [et al.] // arXiv. – 2021. – Access mode: https://arxiv.org/abs/2110.14168.

8. Intelligent System for Semantically Similar Sentences Identification and Generation Based on Machine Learning Methods / P. Zdebskyi, V. Lytvyn, Y. Burov [et al.] // CEUR workshop proceedings – 2023. – Vol. 2604. – P. 317–346.

9. Explaining simple natural language inference / A.-L. Kalouli, A. Buis, L. Real, M. Palmer, V. de Paiva [et al.] // Proceedings of the 13th Linguistic Annotation Workshop. – 2019. – P. 132–143.

Editorial office received the paper 01.03.2024.
The paper was reviewed 18.03.2024.

Zdebskyi Petro – Post Graduate with the Department of Information Systems and Networks, e-mail: petrozd@gmail.com.

Berko Andriyi – Doctor of Science (Engineering), Professor with the Department of Information systems and networks.

National University “Lvivska Polytechnica”.