Approaching LLM Alignment using Agents with RAG*

Vladyslav Fliahin[2,†], Olena Turuta[2] and Oleksii Turuta[1,†,*]

[1] V. N. Karazin Kharkiv National University, Svobody Square, 4, Kharkiv, 61022, Ukraine
[2] Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61000, Ukraine

Abstract
This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values. As AI models become more advanced, alignment becomes increasingly critical. This article explores a novel approach using agents and Retrieval-Augmented Generation (RAG) for alignment. We create a custom knowledge graph based on the Flickr30k subset. Leverage a Neo4j database to store predefined entities and their relationships, which serve as constraints for model outputs. We use RAG to guide the model's generation by focusing only on the relevant entities and relationships detected in an image, ensuring alignment with structured knowledge while ignoring irrelevant details.

Keywords
LLM, Alignment, RAG, Knowledge Graph, Multimodal Data

1. Introduction

1.1. Problem statement

As large language models (LLMs) become more capable [1], the challenge of aligning their outputs with human intent, ethical constraints, and factual accuracy becomes increasingly important. While traditional alignment methods focus on reinforcement learning with human feedback (RLHF) or prompt engineering, these approaches often lack fine-grained control over specific aspects of model behavior. In particular, ensuring that an LLM only considers predefined entities and relationships, especially when processing complex multimodal data like images, remains an open challenge.
This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values.

1.2. Overview of our method

In this work, we propose a novel approach to LLM alignment that leverages agents, Retrieval-Augmented Generation (RAG), and knowledge graphs to enforce controlled generation. We store a structured representation of allowed entities and their relationships in a Neo4j knowledge graph, which acts as a constraint system. When analyzing an image, our method first detects the entities present, retrieves only the corresponding allowed relationships from the database, and then guides the LLM's generation using RAG. This ensures that the model adheres strictly to the predefined knowledge constraints, avoiding irrelevant or undesired outputs.

Our contributions are as follows:

- A knowledge-constrained alignment framework using Neo4j and RAG to regulate model outputs.
- A multimodal alignment strategy that ensures only predefined entities and relations are described from an image.

*ProfIT AI'25: 5th International Workshop of IT-professionals on Artificial Intelligence, October 15–17, 2025, Liverpool, UK
1* Corresponding author.
† These authors contributed equally.
vladyslav.fliahin@gmail.com (V. Fliahin); olena.turuta@nure.ua (O. Turuta); oleksii.turuta@gmail.com (O. Turuta)
0009-0000-1520-0310 (V. Fliahin); 0000-0002-1089-3055 (O. Turuta); 0000-0002-0970-8617 (O. Turuta)

- Empirical validation demonstrates how this approach improves alignment precision while reducing hallucination.

The rest of this paper is structured as follows: Section 2 discusses related work on LLM alignment, knowledge-grounded generation systems, and agents. Section 3 describes our methodology, including the role of LLMs, agents, RAG, and Neo4j. Section 4 presents our experiments and results. Section 5 concludes with key insights and future research directions. Section 6 outlines the limitations of the developed framework.

2. Relevant work

Large Language Models (LLMs) have advanced in structured reasoning through techniques like Chain-of-Thought (the authors showed how such reasoning abilities emerge naturally in sufficiently large language models via a simple prompting [2]), Self-Consistency (the authors propose a new decoding strategy, self-consistency, to replace the naive greedy decoding used in chain-of-thought prompting; it first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths [3]), and Tree-of-Thought (authors introduced a new framework for language model inference, Tree of Thoughts (ToT), which generalizes over the popular Chain of Thought approach to prompting language models, and enables exploration over coherent units of text (thoughts) that serve as intermediate steps toward problem solving [4]), improving inference by generating intermediate steps rather than relying on greedy decoding [5, 6, 7, 8, 9, 10]. In case the amount of information is too big to fit into a prompt, prior works have used knowledge storage, such as knowledge graphs.

Knowledge Graphs (KGs) are structured repositories of interconnected entities and relationships, offering efficient graph-based knowledge representation and retrieval [11, 12, 13].

Prior work combining KGs with LLMs has primarily focused on tasks such as knowledge-based question answering [14, 15, 16, 17], entity-centric retrieval [18, 19, 20], and fact-checking [21, 22, 23].

However, in the described applications, the obtained data was mainly used to extract the correct answer or infer the answer from it. Our work focuses on utilizing the extracted data as supporting information.

3. Agents with RAG (ARAG)

Our approach combines Neo4j knowledge graphs, Retrieval-Augmented Generation (RAG), and agent-based processing to enforce alignment constraints in large language models (LLMs) (see Fig. 1). This section details our framework, outlining how entities and relationships are stored, retrieved, and used to control LLM-generated descriptions of images.

3.1. Overview of our method

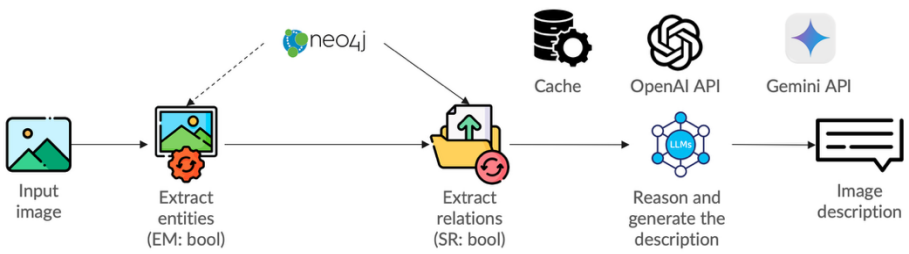

Figure 1: System overview diagram

The proposed system consists of three main components:

- Knowledge Graph: Stores predefined entities and relationships that outline the allowed knowledge constraints.
- Image Processing Module: Extracts entities from the image using the VLM.
- RAG-Enhanced LLM Agent: Retrieves relevant entities and relationships from Neo4j and conditions the model's output within those constraints.

3.2. Knowledge graph

Representation. To build the alignment graph, we incorporated the Flickr30k dataset [24], containing 29000 train, 1014 validation, and 1000 test samples. Each sample from the dataset has the respective image and 5 English captions. Due to resource limitations, we built the graph on top of the first 100 train samples from the dataset. We have checked two different setups:

- Extracting entities and relations from the image captions (this turned out to provide a small number of entities and relations and was not used for the full-size experiments)
- Extracting entities and relations directly from images (was used for the full-size experiments)

The first approach led to the 184 entities and 226 relations, while the second one led to the 231 entities and 404 relations present in the DB. In both cases, GPT-4o was used as a base LLM to generate the DB.

We structure our Neo4j database as a directed graph where:

- Nodes represent entities (e.g., "Person," "Vehicle").
- Edges define relationships between entities (e.g., "drives," "owns").

A sample knowledge graph structure:

(:Person)-[:OWNS]->(:Vehicle)
(:Vehicle)-[:IS_LOCATED_AT]→(:Building)

In the schema above, there are different relations: OWNS and IS_LOCATED_AT. We decided to store the relations under the single RELATES relation and to store the exact relation type as a type attribute. The same procedure was performed with the Person, Vehicle, and Building combined into a single Entity with the name attribute. The actual knowledge graph snapshot is depicted in Fig. 2.

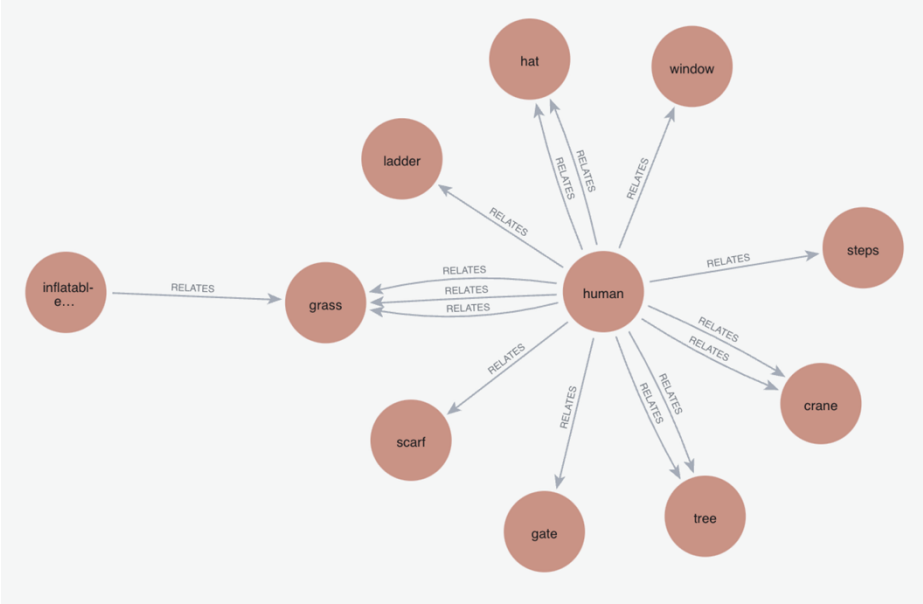

Figure 1: Knowledge graph snapshot

Such a structure ensures that if a "Person" and "Vehicle" appear in an image, only the predefined "OWNS" relationship is considered during LLM generation. There could also be introduced a lot of additional attributes for each of the entities and relations, but for the sake of simplicity, we did not incorporate much metadata.

Querying. The system queries Neo4j to retrieve only the relations that match the list of entities. We have implemented an optional strict mode to customize the retrieval process. The strict case is when we extract two entities, cat and dog, then we extract only relations between them, like dog->bark->cat. An unstrict case would also return all the relations that match only one entity, like dog->eat->food, dog->is playing->ball.

Extracted entities matching. This step is optional and is used only during inference. By utilizing the vector embeddings, we match the extracted entities for a particular image with the list of entities present in the DB. This step is required because some entities may not be consistently extracted every time (e.g., windows/window, man/human, building/house). Without handling such situations, the results may be much worse than expected. We utilize the vectors obtained using the OpenAI Embeddings API. It provides us with 64-dimensional normalized embeddings that are further compared using a cosine similarity score and a 0.7 threshold.

In addition, we incorporated a caching procedure that pre-calculates embeddings for all the new strings and then uses them in case we face the same entity. Caching significantly reduces the actual costs for this operation. This is a local version of the semantic search that could be used on the Neo4j side, but was simplified to the local version. In the production scenario, it should be performed by a single operation in Neo4j.

3.3. Image Processing

The image processing module detects objects and extracts their textual representations (e.g., "car," "building"). Right now, we utilize the VLM for this task, including the input image and the prompt. This works fine for now, but can be substituted by any Zero-Shot model like Grounding DINO or something similar in the future, in case cost reduction is needed for production applications.

3.4. Vanilla LLM

In our setup, Vanilla LLM calls do not use DB or any other tools to perform the task.

3.5. Agent Processing

There are two different types of agents that we could use: tool-calling agents and code agents. After preliminary experiments, we found out that code agents perform better in planning and aligning with the step-by-step nature of the instructions.

During the implementations, we utilized the smolagents framework as a core of the agent backend. ReAct was used as the agent's planning strategy.

RAG-Enhanced Generation. The agent conditions the LLM using retrieved knowledge, ensuring it describes only the allowed entities and relationships. Furthermore, we prompted the model to identify the exact positions of the entities in the image. The are nine available values for position: top-left, top-center, top-right, center-left, center, center-right, bottom-left, bottom-center, and bottom-right. After the initial description generation, the agent defines the positions for each entity and verifies its answer with the DB to minimize the possible hallucinations.

4. Experiments

To accept or reject the alignment improvements, we compared 10 head-to-head approaches utilizing 50 samples and three different metrics.

4.1. Data

To accept or reject the alignment improvements, we compared 10 head-to-head approaches utilizing 50 samples and three different metrics.

4.2. Models

We have utilized two different API providers: OpenAI and Google. GPT-4o-mini and Gemini-2.0-flash-lite were selected as candidates. On top of vanilla LLMs, we introduced different modifications for them. ARAG suffix means that the models utilize Agentic flow with RAG. The EM suffix means this setup uses the entities matching option, and the SR means that we included the strict relations option.

Regular LLM calls (without agentic flow) did not have access to the tools/DB. All the agentic flow setups were equipped with three tools (load_initial_image, extract_entities, and get_data_from_neo4j), a code interpreter, and a maximum of 7 steps to accomplish the task.

For the LLM-as-a-judge purpose, we utilize the same model we used during the knowledge graph creation – gpt-4o. This will mitigate the bias of inference and evaluation using the same model. To calculate the BERTScore, we incorporate bert-base-uncased.

4.3. Metrics

To evaluate the experiment results, we need to be able to understand how good the model performs at describing the image and to what extent it aligns with the DB reference. We focused on the following metrics:

- Answer Relevance [0 - 10] – Measures how well the generated response aligns with the expected content of the image. It is assessed based on the LLM judgment of the generated descriptions of the image. A higher score indicates that the response remains applicable and contextually appropriate despite alignment constraints.
- Groundedness [0 – 10] – Evaluates the extent to which the model's output is based on retrieved knowledge rather than hallucinated information. A higher groundedness score indicates better alignment with structured knowledge and reduced model hallucination.

- BERTScore [0 – 1] – A widely used text similarity metric based on contextual embeddings from a pre-trained BERT model. It compares the generated description with reference captions by computing cosine similarity between token embeddings.

5. Results

The overall prediction time took ~4.5 hours to accomplish. The evaluation took ~14.5 hours due to the GPT-4o's degraded performance stated by OpenAI in the last few days. We have highlighted the best results per model in bold because we have to evaluate the models separately compared to the vanilla LLM setup. The final results are depicted in Table 1 below.

Table 1.
ARAG vs vanilla LLM performance

Approach	Relevance	Groundedness	BERTScore
GPT-4o-mini	**8.72**	5.0	**0.560**
GPT-4o-mini-ARAG	7.06	6.0	0.538
GPT-4o-mini-ARAG-EM	6.74	**6.34**	0.542
GPT-4o-mini-ARAG-SR	6.44	5.88	0.536
GPT-4o-mini-ARAG-EM-SR	6.14	6.18	0.536
Gemini-2.0-flash-lite	**8.52**	5.92	**0.607**
Gemini-2.0-flash-lite-ARAG	6.40	6.82	0.592
Gemini-2.0-flash-lite-ARAG-EM	5.82	**7.68**	0.587
Gemini-2.0-flash-lite-ARAG-SR	6.20	6.60	0.593
Gemini-2.0-flash-lite-ARAG-EM-SR	5.94	6.92	0.589

As we can see, all the somehow aligned setups obtained higher groundedness scores compared to the vanilla LLM approach. The final groundedness score also significantly depends on the base model capabilities. In our results, we can state the significant difference between the Gemini-2.0-flash-lite and the gpt-4o-mini alignment capabilities.

Meanwhile, answer relevance dropped significantly during alignment, the answers provided by the aligned models are still valid (but, of course, less detailed). The most important thing is that we see a correlation between the extent of alignment and the answer relevance.

BERTScore is almost the same, meaning both models provide captions that partially align with the labels. This can be explained by low-detail captions describing the Flickr30k dataset. They are right to the point and were initially focused on much less capable models.

From the above experiments, we can say that for a more optimal performance entities matching should be set to True while the strict relations should be set to False.

6. Conclusion

The scientific novelty of the presented system lies in contributing scalable LLM alignment approaches to generate outputs close to human goals and values. We introduced an approach for improving LLM alignment using Neo4j knowledge graphs, Retrieval-Augmented Generation

(RAG), and agent-based processing. Our method ensures that an LLM describes only predefined entities and relationships extracted from an image, enforcing structured alignment constraints. Using vector-based entity matching, strict relationship retrieval, and agentic execution, we successfully constrained the model's output while maintaining relevant and coherent responses.

Our experimental results demonstrate that the aligned agent-based approach significantly improves groundedness compared to a vanilla LLM with prompting. The trade-off is a slight drop in answer relevance, but overall, the method remains effective for structured and controlled generation. Importantly, BERTScore results suggest that despite these constraints, the aligned model's responses still align with human-annotated captions at approximately the same level.

While our approach has proven effective, there are several avenues for future work:

- Scaling predictions to larger models such as GPT-4o or fine-tuned open-weight LLMs to reduce hallucinations.
- Expanding the dataset beyond the first 100 samples of Flickr30k to increase generalization.
- Verifying the performance using the Ukrainian multi30k dataset [25].
- Verify the framework performance on the specific domains, like Autonomous Driving.
- Integrating Neo4j-side semantic search for more efficient and scalable entity matching.
- Exploring techniques to enhance the alignment of vision-language models with structured knowledge.
- Researching the capabilities of image-based alignment instead of the entities and relations DB [26, 27, 28].

By combining custom constraints from knowledge graphs with retrieval-enhanced generation, our work demonstrates a promising pathway for more controllable, aligned, and factually grounded LLMs. This methodology can be extended to other alignment-sensitive applications, such as medical AI, legal document analysis, and explainable AI systems.

Source code is available on GitHub [29].

6.1. Limitations

This research was conducted as part of the master's thesis at the Kharkiv National University of Radio Electronics. That is the reason why the KG sizes and the evaluation sizes are not that big. The overall experiment budget was around 30$. We tested only the GPT-4o, GPT-4o-mini, and the Gemini-2.0-flash-lite models in our tests because of their affordability. We expect our approach to scale even better using the more prominent models like Gemini-2.0-pro and Claude-sonnet-3.7.

The obtained framework sometimes faces the output token limits during the evaluations. These are just being retried for now.

We have conducted full-size experiments only using the Code Agent. After the preliminary tests, the Tool Calling Agent almost always showed worse results.

We did not evaluate our framework on the new data samples that were not used to build the KG.

Image size is limited to 20MB (current OpenAI limitation).

Acknowledgements

This publication is based upon work from COST Action GOBLIN - Global Network on Large-Scale, Cross-domain and Multilingual Open Knowledge Graphs (CA23147), supported by COST (European Cooperation in Science and Technology).

Declaration on Generative AI

During the preparation of this work, the author(s) used Gemini-2.5 in order to: Grammar and spelling check.

References

[1] Erkut Erdem, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii, Oleksii Turuta, Aykut Erdem, Iacer Calixto, Elena Lloret, Elena-Simona Apostol, Ciprian-Octavian Truică, Branislava Šandrih, Sanda Martinčić-Ipšić, Gábor Berend, Albert Gatt, and Grăzina Korvel, Neural Natural Language Generation: A Survey on Multilinguality, Multimodality, Controllability and Learning. J. Artif. Int. Res. 73 (2022).

[2] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou et al., Chain-of-thought prompting elicits reasoning in large language models. In: Advances of the 36th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 24824-24837 (2022).

[3] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, Self-consistency improves chain of thought reasoning in language models. In: Proceedings of the 11th International Conference on Learning Representations (ICLR) (2023).

[4] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, Tree of thoughts: Deliberate problem solving with large language models. In: Advances of the 37th Neural Information Processing Systems (NeurIPS), pp. 11809-11822 (2023).

[5] S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig, Docprompting: Generating code by retrieving the docs. In: Proceedings of the 11th Conference on Learning Representations (ICLR) (2023).

[6] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, Large language models are zero-shot reasoners. In: Advances of the 36th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 22199–22213 (2022).

[7] A. Creswell, M. Shanahan, and I. Higgins, Selection-inference: Exploiting large language models for interpretable logical reasoning. In: Proceedings of the 11th Conference on Learning Representations (ICLR) (2023).

[8] N. Shinn, B. Labash, and A. Gopinath, Reflexion: an autonomous agent with dynamic memory and self-reflection. In: Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc., Red Hook, NY, USA, Article 377, pp. 8634–8652 (2023).

[9] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk et al., Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2024).

[10] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, STaR: Bootstrapping Reasoning With Reasoning. In: Advances of the 36th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 15476–15488 (2022).

[11] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods. In: 17th international semantic web conference (ISWC), IOS Press, NLD, pp. 489–508 (2017).

[12] Q. Wang, Z. Mao, B. Wang, and L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724–2743 (2017).

[13] Y. Jing, Y. Yang, X. Wang, M. Song, and D. Tao, Meta-aggregator: Learning to aggregate for 1-bit graph neural networks. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp. 5281–5290 (2021).

[14] D. Sanmartin, Kg-rag: Bridging the gap between knowledge and creativity, arXiv preprint (2024).

[15] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, Knowledge graph prompting for multi-document question answering. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI), pp. 19206–19214 (2024).
[16] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In: Proceedings of the Advances in 37th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 132876-132907 (2024).
[17] X. Li, R. Zhao, Y. K. Chia, B. Ding, S. Joty, S. Poria, and L. Bing, Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
[18] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, Reasoning on graphs: Faithful and interpretable large language model reasoning. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
[19] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, H.-Y. Shum, and J. Guo, Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
[20] H. Liu, S. Wang, Y. Zhu, Y. Dong, and J. Li, Knowledge graph-enhanced large language models via path selection. In: Findings of the 62nd Association for Computational Linguistics (ACL), Bangkok, Thailand, pp. 6311-6321 (2024).
[21] R.-C. Chang and J. Zhang, CommunityKG-RAG: Leveraging Community Structures in Knowledge Graphs for Advanced Retrieval-Augmented Generation in Fact-Checking, arXiv preprint (2024).
[22] Y. Mu, P. Niu, K. Bontcheva, and N. Aletras, Predicting and analyzing the popularity of false rumors in weibo, Expert Systems with Applications, vol. 243, p. 122791 (2024).
[23] A. Kau, X. He, A. Nambissan, A. Astudillo, H. Yin, and A. Aryani, Combining knowledge graphs and large language models, arXiv preprint (2024).
[24] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier, From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions, Transactions of the Association for Computational Linguistics, pp. 67–78 (2014).
[25] Nataliia Saichyshyna, Daniil Maksymenko, Oleksii Turuta, Andriy Yerokhin, Andrii Babii, and Olena Turuta, Extension Multi30K: Multimodal Dataset for Integrated Vision and Language Research in Ukrainian. In: Proceedings of the 2nd Ukrainian Natural Language Processing Workshop (UNLP), pp. 54–61 (2023).
[26] Kyrychenko, I., Tereshchenko, G., & Smelyakov, K., Optimized Indexing Method in a Hybrid Image Storage Model for Efficient Storage and Access in Big Data Environments. In: Proceedings of the 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), 1-4 (2024).
[27] Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I. and Kobylin, O., Reducing computational costs by compressing the structural description in image classification methods, Advanced Information Systems 9, 5-12 (2025).
[28] Gorokhovatskyi, Volodymyr, et al. "Search for visual objects by request in the form of a cluster representation for the structural image description." Advances in Electrical and Electronic Engineering 21.1 (2023).
[29] Implementation of the paper, https://github.com/Vlad-Fliahin/LLM-alignment-with-ARAG, last accessed 2025/04/20.